# PROVABILITY-BASED SEMANTIC INTEROPERABILITY BETWEEN KNOWLEDGEBASES AND DATABASES VIA TRANSLATION GRAPHS

By

Joshua Taylor

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF COMPUTER SCIENCE

Approved:

_____

Selmer Bringsjord
Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

July 2007
(For Graduation August 2007)

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

This thesis presents *provability-based semantic interoperability* (PBSI), a type of semantic interoperability characterized by the ability to express complex relationships between ontologies, and to share information even in situations where information cannot be directly *translated* from one ontology to another. Relevant research in interoperability is reviewed, including languages and ontologies that have been designed to facilitate the exchange of information, as well as techniques for relating ontologies and automating information exchange between them. Work in the Rensselaer Artificial Intelligence and Reasoning (RAIR) Laboratory during a number of interoperability experiments is discussed, with particular respect to a new technique for enabling interoperability. Finally, *Translation Graphs* are introduced as a new formal structure for automatically extracting axiomatic relationships that govern the sharing of information between multiple ontologies, even in cases where information cannot be directly translated. The structure of Translation Graphs is described, and examples and a sample implementation is given.

# CHAPTER 1
# Introduction

Herein I review several motivations for achieving semantic interoperability that have arisen with the advent of pervasive computing and knowledge bases that store vast amounts of useful information. I give a high-level overview of *semantic interoperability*, a concept further developed in later sections. The work described herein lies in the overlap of artificial intelligence (AI), knowledge engineering, and ontology management.

## 1.1 The Need for Semantic Interoperability

### 1.1.1 In a Truly Useful Semantic Web

One of the promises of the information age is unfettered and uniform access to myriad information sources through the Internet. Unfortunately, this promise is yet unfulfilled except in movies and television:

*A detective opens his laptop computer and, with but a few hurried keystrokes, accesses 'the database'. He scans police records, property deeds, criminal histories, financial and employment background, genealogical records and more. These are cross-referenced at the click of a mouse, or the press of a key, and the detective speeds off in pursuit of his next target.* What organization keeps and cross-references *all* of that information?

*A vacationer sits down at her desktop PC and opens a web browser to book travel reservations, and moments later is comparing the cheapest five-star hotels near the best beaches, and sorting them based on their proximity to inexpensive Thai restaurants and historic train stations. Then, without leaving the website, she finds a next train leaving from the local station; she can catch it if bags are packed and the house left within the hour.* What travel agency researched all of this and indexes it so efficiently?

The answer in both cases is that *no one* can collect, store, organize, and maintain such a large quantity of information. It would be wasteful to do so; the

information is already available and organized and stored elsewhere and would so quickly become out of date. It would be difficult to do so; local copies of the information must be maintained and updated on a continuous basis.

There are now websites which advertise their ability to find personal information and perform background checks, and sites which can find and compare travel packages, though they fall short of Hollywood's visions. The reason that these websites are popular is because they have devoted a great deal of time and effort to tackle *very specific* instances of a very difficult problem. They have collected information from a variety of sources, studied the just as numerous formats in which it is represented, determined workable methods to organize and cross-reference this information, and built an interface to access it.

Problems of information exchange are not new with the advent of pervasive computing technology, but their presence is much more pronounced as engineers and programmers try to tackle the problem with software. There is much more information available now, but rarely is it in the form a particular application needs.

The early documents of the World Wide Web had only hyperlinks and basic markup. The innovation of hyperlinking and the use of the Common Gateway Interface (CGI) was enough to cause the explosion of the World Wide Web, devoid of meaningful *semantic* content that could be easily and *mechanically* extracted.

To achieve a useful Semantic Web, information on the web must (1) be structured in a meaningful way and (2) information from different systems must be able to be combined easily and *meaningfully*. (1) is being addressed as more and more information is stored in databases, and by the adoption of regular markup languages such as XML and XHTML. (2) is only happening partially. Service-oriented architectures *are* sharing information meaningfully, but need complete knowledge of the ontologies employed by the systems involved. When systems with web presence can share information without having to have extensive knowledge of their peers' ontologies or schemata, web-based agents can be built that are capable of deep reasoning and planning [40].

More and more emphasis is being placed on semantic markup, and extensible markup languages like XML and XHTML allow authors to *extend* their markup

languages with new tags corresponding to the terms in their ontologies.

This new and flexible mechanism for marking semantic knowledge is useful and makes document and text retrieval easier and more effective than ever before, but a caveat remains: a seeker must be aware and familiar with the *author's choice* of markup and ontology.

### 1.1.2 Intelligence Analysis

The defense and intelligence communities have, for some time, recognized the need for semantic interoperability, and have sponsored research and development of tools and languages to address this need. Some results of this research are the DARPA Agent Markup Language [22], DAML, and DAML+OIL, for the markup of information and for the description of ontologies. Though languages such as KIF (§2.1.1.3) and Common Logic (§2.1.1.4) have been developed for describing the *relationships* between ontologies and for the exchange of information between ontologies, in 2005 the Disruptive Technology Office sponsored the Interoperable Knowledge Representation for Intelligence Support (IKRIS) Workshop (§3.1), which resulted in the IKRIS Knowledge Language (IKL), an extension to Common Logic that addresses specific needs of the intelligence community.

Intelligence analysts (IAs) use many different software packages with different featuresets, but interoperability with other packages has, until recently, usually been an afterthought.

Programmers studying the process of intelligence analysis have written software to aid analysts in many ways, including evidence collection, data visualization, and collaboration with other analysts. These systems represent many different kinds of information, and, understandably, do not employ the same knowledge representation formats. As a result, there are many good systems now available to IAs, but there is great difficulty in using these tools together.

Slate (§3.1.1.1) and Solomon [13] are two applications under development at the Rensselaer Artificial Intelligence and Reasoning (RAIR) Laboratory to aid IAs in their day to day work. Slate offers IAs automated hypothesis generation, a workspace for argument construction and verification, and tools for bias identifi-

cation. Solomon is an intelligent question and answer (QA) system which aims to produce rational, justified answers for conceptual, hypothetical, and open-ended questions. Both Slate and Solomon have been designed and implemented taking interoperability into account. The techniques described in this thesis have been developed with RAIR Lab systems as testbeds, and these systems are the first to integrate the lab's interoperability technology.

# CHAPTER 2
# Preliminaries

In this chapter, I first summarize major developments in languages for information representation, and then examine several large ontologies. With these in mind I examine past approaches to enabling semantic interoperability. Finally, I define *provability-based semantic interoperability* (PBSI) and argue why it is the only technique that can fully realize the potential of semantic interoperability.

## 2.1 Heritage of Information Exchange

### 2.1.1 Languages and Representations

A number of languages and representations have been developed to facilitate the use and interchange of symbolic information. These range from formal logics to graph notations and vary greatly with their intended applications. Several of the most prominent representations and languages are discussed here.

#### 2.1.1.1 Semantic Networks and Conceptual Graphs

The term *semantic network* refers to a number of structures (which are usually expressed graphically) that have been used for knowledge representation. Sowa describes six commons types of semantic networks [72]. These include *learning networks*, or neural networks, *implicational networks*, or Bayesian networks, and executable networks, but those which are of the most interest to knowledge representation are *definitional networks* and *assertional networks*.

The foundation of definitional networks is found in Aristotle's *Categories*. Therein, Aristotle describes his system for categorizing individual objects as well as their classes [4]. Porphyry depicted Aristotle's work graphically in the third century; such depictions became known as an *arbor porphyriana* or Pophyrian tree. With only a surprisingly small amount of modification, these definitional networks have become one of the primary methods of describing object oriented programming systems, database schemata, as well as the foundation for description logics.

The first assertional networks were Peirce's existential graphs, which graphically encoded sentences of first-order logic.

*Conceptual graphs* [71] combine some of the properties of Peirce's existential graphs with those of Shapiro's propositional networks [68].

#### 2.1.1.2 FOL

First-order logic was developed by Frege, Peirce and others to aid in the disambiguation of sentences expressed in natural languages, and to provide a sound and valid mechanism for manipulating these sentences. First-order logic is appealing as interchange language because of its well-defined structure, flexibility, and ease of use. Though there are shortcomings in the expressiveness of first-order logic, much of mathematics can and has been expressed in first-order logic and corresponding theorems proved [67].

Though first-order logic is used as a singular term, properly speaking, there are a set of *first-order logics*, or a family of *first-order languages*, each of which is defined by a set of relation and function symbols. While academic texts tend to describe a first-order logic using a particular syntax, it is understood that there are many syntaxes one can choose from. E.g, $\forall x\ Happy(x)$ and `(forall (?x) (Happy ?x))` are simply syntactic variants of the same sentence. The term first-order logic, then, can be understood as a description of the types of structures an actual language must support, and the set of first-order languages as the set of actual languages which do support them.

#### 2.1.1.3 KIF

The *Knowledge Interchange Format* (KIF) [31] is one product of the ARPA Knowledge Sharing Effort (KSE) [36]. The KSE also encouraged development of the related tools, such as parsers, common knowledge bases, and the formalization of certain common features of knowledge representation languages [56].

KIF is language with first-order semantics that was developed as an interchange format that would enable the reuse and sharing of information from many different knowledge bases. It was intended to be an expressive logical language for the description of knowledge bases and ontologies, including the SUO (§2.1.2.2).

KIF's design also took into consideration some of the lower-level interoperability issues of its day, such as character encodings and file formats [76].

KIF included constructs specifically for ontology construction, such as forms for defining functions and relations axiomatically in terms of other existing forms. As an example of such definitions, in standard first-order logic, given the functions *FatherOf* and *PaternalGrandfatherOf*, the latter can be defined by the axiom

$$\forall x \left[\forall y \left[(y = \mathit{PaternalGrandfatherOf}(x)) \leftrightarrow (y = \mathit{FatherOf}(\mathit{FatherOf}(x)))\right]\right] \quad (2.1)$$

or, perhaps, more concisely,

$$\forall x \left[\mathit{PaternalGrandfatherOf}(x) = \mathit{FatherOf}(\mathit{FatherOf}(x))\right] \quad (2.2)$$

Experience has shown that human authors are prone to commit errors while writing axiomatic definitions. For instance, consider the following example, which aims to axiomatically relate a binary predicate *LocatedAt* and a unary function *LocationOf* from events to locations:

$$\forall l \left[\forall e \left[(l = \mathit{LocationOf}(e)) \leftrightarrow (\mathit{LocatedAt}(e, l) \wedge \mathit{Location}(l)) \wedge \mathit{Event}(e))\right]\right]$$

This states that $l$ is the location of $e$ if and only if $e$ is located at $l$, $l$ is a location, and $e$ is an event.

However, mathematical and logical functions, such as *LocationOf* are understood to map *every* element in the universe to some value. Then *LocationOf* must map every element in the universe to some value. However, if it is not the case that, for every element $x$ in the universe, $\mathit{Event}(x)$, then there is a contradiction. A better formalization would be:

$$\forall l \left[\mathit{Location}(l) \rightarrow \forall e \left[\mathit{Event}(e) \rightarrow (l = \mathit{LocationOf}(e)) \leftrightarrow \mathit{LocatedAt}(e, l)\right]\right]$$

This latter formalization is correct; for every event $e$, the location $l$ of $e$ is a location, and it must be the case that $e$ is located at $l$.

Prompted by the difficulty associated with writing *correct* defining axioms,

```
(deffunction paternal-grandfather-of (?x) :=
  (father-of (father-of ?x)))
```

**Figure 2.1:** `paternal-grandfather-of` **specified with KIF's** `deffunction`**.**

```
(= paternal-grandfather-of
   (lambda (?x)
     (father-of (father-of ?x))))
```

**Figure 2.2: The defining axiom for** `paternal-grandfather-of`**.**

KIF's designers included special syntactic forms for specifying such axioms. For instance, Equation. 2.1's representation in KIF is given in Figure 2.1. The definition generates a *defining axiom*, shown in Figure 2.2.

#### 2.1.1.4 Common Logic

After KIF and Conceptual Graphs (CG) had been adopted by various researchers, a *Common Logic* (CL) Standardization group formed to formalize the relation between KIF and Conceptual Graphs [37]. Common Logic standardized the underlying semantics of the two representation formats and developed an abstract syntax of which KIF and CG would be particular dialects.

CL's development was concurrent to the explosion of the World Wide Web, and the use of XML as an interchange format. Provisions for namespaces, and special treatment of string datatypes and URIs were added to the language.The final ISO draft [20] specified three concrete syntaxes (i.e., dialects) of Common Logic, viz., Common Logic Interchange Format (CLIF), Conceptual Graph Interchange Format (CGIF), and eXtended Common Logic Markup (XCL).

### 2.1.2 Ontologies

In philosophy, *ontology* is the study of existence, of what is, the things that are (not only physical things, but also concepts and classes, and so on), and of the categorization or organization of these things. Ontology is used in computer science with a similar, but distinct meaning. Within computer science, an ontology is a model of data. Data here may refer to the data structures and objects within a

program, or can refer to that which is stored in a knowledge representation system.

In a knowledge representation system based on first-order logic, a simple family-based ontology might specify functions in the language such as $\mathsf{MotherOf}(x)$, $\mathsf{FatherOf}(x)$, that the relations are $\mathsf{Parent}(x, y)$, and $\mathsf{ChildOf}(x, y)$. The ontology would also specify constraints on the models of these relationships; e.g., that

$$\mathsf{Parent}(x, y) \rightarrow (\mathsf{MotherOf}(y) = x \wedge \mathsf{FatherOf}(y) = x)\,,$$

that $\mathsf{MotherOf}(x) \neq \mathsf{Fatherof}(x)$, and so on.

The languages described so far have been designed for the interchange of knowledge and information. There are two reasons that information cannot be shared directly between knowledge representation systems:

1. The *syntax* of representation formats are not identical.

2. The ontologies to which the representation systems subscribe are not the same.

The first would be easily solved by syntactic manipulations (when the underlying formalisms of the systems are sufficiently similar) if the second were not the case. There are a number of causes for the second:

1. Many applications to which knowledge representation is applied have a small domain and it is more convenient to build a small, special-purpose ontology for the application.

2. Even when the domains of ontologies overlap, many domains can be formalized in many (possibly mutually exclusive) ways, and no one formalization is clearly better than the other. No single ontology can contain all the formalizations that a knowledge engineer might like to use.

3. Many domains can be formalized to various levels of granularity. What may be an appropriate level of abstraction for one application may be far too abstract, or far too in-depth for another.

Nonetheless, there have been some valiant efforts toward building ontologies that capture enough information to be adopted by knowledge engineers. These

ontologies, to various degrees, *have* been integrated, at least in part, with some applications.

The difficulties of interoperability stem from the surface-level incompatibilities of ontologies, and the promise of interoperability from the belief that the same incompatibilities of ontologies are, in fact, reconcilable.

Several large ontologies which have been built to provide information for a large number of systems are reviewed here. These ontologies are prominent examples of the approach to knowledge representation that attempts to built an all-encompassing ontology capable of encoding all the information that might arise.[1]

#### 2.1.2.1 Cyc

The Cyc project is one of the long-standing players in symbolic knowledge-based AI. Launched in 1984, the Cyc project's aim was to build a knowledge base that would contain all the commonsense information (or at least enough) that an intelligent agent would need to behave reasonably in the real world. Properly speaking, Cyc includes not only an *ontology*, but also specialized inference procedures, and knowledge management tools [49].

The Cyc knowledge base is represented in the Cyc Representation Language, *CycL*, a language with declarative first-order semantics, but a syntax which makes some higher-level notions easy to express [48].

A comparison of the possible uses of Cyc's KB and inference technologies gave as the "better" and "best" possibilities for Cyc the following (taken verbatim from [49, p. 34]):

- Better: Cyc's KB is used by the next generation of AI research programs and its size and breadth help make them more than theoretic exercises. No one doing research in symbolic AI in 1999 wants to be without a copy of Cyc, any more than today's researchers want to be without EVAL and ASSOC. Eventually, it empowers the first full-fledged natural language understanding systems, non-brittle expert systems, and machine learning systems.

[1] This is of course, a slight exaggeration. The ontologies presented are intended to encompass a wide domain, and to provide a foundation on which more specialized ontologies can be built.

- Best: Cyc, or something similar, serves as the foundation for the first true artificial intelligent agent [sic]. Application programs routinely tie into it, in effect letting it look over their shoulder. No one in the early twenty-first century even considers buying a machine without common sense, any more than anyone today even considers buying a PC that cannot run spreadsheets, word processing, and networking software.

The "better" case is only partially realized; *some* research institutions *do* make productive use of Cyc's technology, and the the OpenCyc [1] project, an open source version of the Cyc technology, includes the Cyc ontology and inference engine (though the inference engine is not made open source).

In the "best" case scenario, "application programs routinely tie into it". The majority of application programs are designed without knowledge representation as their primary purpose, and do not use CycL as a data representation. Though it could not be reliably predicted in 1990, many application programs do use certain formats of structured data representation; XML is the chief example today.

#### 2.1.2.2 SUMO and SUO

An upper-level ontology aims to codify general-purpose terms, and to provide a foundation on top of which more specialized ontologies can be built [59]. An upper-level ontology will provide top-down descriptions, such as `Physical` and `Abstract`, whereas a domain-specific ontology probably would eschew these types of categorizations.

The Standard Upper Merged Ontology (SUMO) is an upper-level ontology that was formed by combining a number of existing upper-level ontologies [61]. The merge occurred in two phases. The first was a syntactic joining of the ontologies, which rewrote all of the information in SUO-KIF, a version of KIF designed for the Standard Upper Ontology. The second was a semantic merge in which researchers examined the contents of the merged ontologies in great detail, searching for contradictions, or ontological mismatches.

The Standard Upper Ontology working group [2, 57] is working toward building the Standard Upper Ontology (SUO) from which other ontologies can be built.

SUMO has been proposed as a starting document to be taken as the SUO. Further advantages to using SUMO as SUO come from the work that has already been done to make use of SUMO, including SUMO to aid in natural language processing [62], in processing *controlled* English [60], and mapping terms in WordNet to corresponding terms in SUMO [58].

## 2.2 Relevant Past Approaches

I review a number of past approaches to the problem of semantic interoperability, noting how these approaches fare with respect to the aforementioned desiderata; in particular, whether: an approach is logically based, asymmetry of translation is preserved, information in a foreign ontology can influence a query in a native ontology even when the foreign information cannot be directly translated, and the quality of available justifications is sufficiently high.

### 2.2.1 Schema Matching

When the subject domains and vocabularies of the ontologies to be related are similar and the information represented within them is not too complex, *schema matching* can be effective in translating information from one ontology to another. With schema matching, corresponding terms from the ontologies are selected, and information from one is recast in another. There are automated tools that aid in schema matching [75]. Evaluating whether a schema matching is correct can be difficult, particularly if the matching has been generated (even partially) automatically. A schema matching can be provided as primitive justification for results. It seems difficult, with schema matchings, to capture semantic influence when information translation is not possible.

### 2.2.2 Institutions and Schema Morphisms

The use of *schema morphisms* to map the sentences of one ontology to sentences of another allows for more complex transformations between ontologies. This approach can be used when ontologies are treated as *institutions* [35]. Within this framework, it is possible to determine whether a schema morphism is correct [33],

and to impose constraints that capture some of the asymmetry of translation and semantic influence. Morphisms are not trivial to construct, but can capture relationships between ontologies using different logics [34]. Signature morphisms are expressed with a different formalism and notation than the ontologies themselves, however, and so the justification for a particular translation requires human intervention or specialized reasoning outside of the ontologies themselves.

### 2.2.3 Axiomatic Translation

Simple syntactic manipulation of sentences does not afford the meaningful translations that are desired. In fact, to answer queries expressed in a query ontology using information from various source ontologies often requires making use of information from many source ontologies. Unfortunately, sentences in a source ontology that have semantic consequences in a target ontology cannot always be translated into the target ontology [24]. Semantic interoperability is still attainable, however, by relating the ontologies logically, and evaluating queries with respect to provability. Ontologies can be related axiomatically using *lifting axioms* [16] or by merging the ontologies to be related and expressing *bridging axioms* in the new merged ontology [25].

### 2.2.4 Summary

The techniques reviewed above have been used in real applications and have successfully enabled varying levels of interoperability. No individual system, however, possesses all of the necessary qualities for top-notch semantic interoperability. Building on these excellent foundations, we believe that our system of translation graphs takes a step closer to the ideal.

## 2.3 What is Semantic Interoperability?

Many systems today achieve various levels of interoperability and information exchange using ontology mapping [18] and schema matching [75]. These techniques are useful and have achieved high levels of information sharing, but cannot capture all the relationships that semantic interoperability requires.

In the general tradition of logicist AI and cognitive science [8, 9, 10], and specifically in the tradition of logic-based semantic interoperability [16, 25], we maintain that semantic interoperability can be evaluated only with respect to provability-based queries. This stems from the fact that ontology mapping and schema matching cannot always capture asymmetry of translation [24], nor can information from a source ontology always be translated into a corresponding form in a target ontology, even if the information has semantic consequences in the target ontology.

Ontologies contain complex relationships among their own terms, and any approach to semantic interoperability must be able to capture not only these, but also the relationships between between multiple ontologies. A system which does not use a sufficiently expressive formalism or language to describe these relationships is inherently specialized and cannot be used for general applications.

Furthermore, consumers of the products of semantic interoperability should have access to the *justifications* that bring about those products. Consumers should have, then, in a schema-mapping approach, access to the mapping itself, in an axiomatic approach, access to the axioms, and in a *provability*-based approach, access to the *proofs*. Ideally, the proofs would be couched in a format that is readily understood by non-specialists; e.g., proofs in natural language are far superior to resolution-based proofs. Herein we describe a new brand of PBSI that meets the desiderata just enumerated.

# CHAPTER 3
# Experiments

Through a number of interoperability experiments we have been exposed to the kinds of information that today's software systems must be able to share and exchange. The amount of attention paid to knowledge representation by the systems varies greatly; some are primarily knowledge-based, while others only incidentally make use of knowledge representation techniques. Several experiments stand out for the amount of information exchanged, the complexity of the information shared, or the important abilities gained through interoperability.

## 3.1 IKRIS

Teams working in the ARDA-funded Novel Intelligence from Massive Data (NIMD) program developed a number of software systems to aid IAs and other members of the intelligence community (IC) in day-to-day work. Various projects focused on collecting, filtering, and categorizing massive amounts of information from many sources. Other projects aimed to help analysts organize and retrieve information once it had become available. Yet other teams built systems to help the analyst document their reasoning and arguments, and to generate reports.

While NIMD was initiated to address the sheer volume of information now becoming available to IAs as well as the heterogeneity of these data, and though each team had varying success in handling and processing some amount of information, analysts to whom NIMD tools became available were left with a new problem. More tools were available to analysts than any individual analyst could use, certain types of tools were specialized for certain types of analysts, and the output of these tools was, itself, heterogeneous; the results produced by one tool could not easily be used in another tool.

In April of 2005, the Disruptive Technology Office, DTO (formerly ARDA), sponsored the 18-month IKRIS workshop [54] which sought to enable interoperability between not only NIMD tools but many knowledge representation and reasoning

systems originating in multiple DTO programs.

The three specific goals of the workshop were given [54]:

**Workshop Goal(s)**

- Specify a knowledge interchange formalism that enables interoperability of KR&R systems across DTO and IC programs;
- Evaluate the interchange formalism by testing whether it can be used to effectively interchange the knowledge bases developed for sample analysis tasks between the knowledge representation and reasoning modules of prominent analyst support systems being developed in ongoing DTO programs;
- Actively seek opportunities to transfer IKRIS-derived specifications and technologies to operational users.

The resulting interchange formalism described a framework of knowledge representation systems employing their own ontologies, a dedicated intertheory represented in a new language (IKL) designed for information exchange, and for each system, a pair of translators, one translating from the the particular system into the intertheory, the other translating from the intertheory into the system.

The knowledge bases developed for sample tasks were based upon Case Studies used at the Joint Military Intelligence College, and were encoded in the systems participating (§3.1.1) in the IKRIS workshop.

The third goal is an ongoing task, but the realization of one such opportunity is the continuing collaboration (§3.2) between Slate (§3.1.1.1) and Oculus' GeoTime (§3.2.1).

### 3.1.1 Participating Systems

Three primary knowledge representation and reasoning systems were involved in the Capstone Demo. The particular systems involved were a function of the workshop participants; many of those involved in the workshop were also associated with research or development of software with significant knowledge representation

| Professional "Laic" Reasoners | Formal Reasoners |
|---|---|
| Intelligence Analysts, item writers, ... | Mathematicians, logicians, technical philosophers, ... |
| Students of Intelligence Analysis, item writing, ... | Students of math, logic, philosophy |

**Table 3.1: Slate's Target Audience**

or reasoning components. The systems first equipped with and able to employ IKRIS technology are Slate, KANI, and Noöscape.

#### 3.1.1.1 Slate

Slate [14] is a system under development at the RAIR Lab. Work on Slate began in 2003 under ARDA's NIMD program. Under NIMD, Slate was intended to aid IAs throughout the analytic process, but with particular emphasis on the subfields of hypothesis generation, argument construction and refinement, and report generation.

Though Slate is built with IAs in mind, the IA is only one occupation whose product is enhanced by Slate. In fact, the workflow of the IA is remarkably similar to that of the mathematician or logician; indeed, any profession characterized by analytic and creative reasoning can benefit from Slate. Slate's target audience [15], is given in Table 3.1.

Beginning in the Fall semester of 2005, Slate (in conjunction with NDL [5]) has been used by students in the Introduction to Logic course at RPI as an intelligent assistant for designing, specifying, and validating or invalidating proofs and arguments.

The processes that the IA and mathematician follow are very similar, and are compared in Table 3.2.

| | |
|---|---|
| IA tasked. | Mathematician given problem. |
| Reads, gathers data, etc. | Reads, gather info on prior work, etc. |
| Develops an argument-sketch in support of an hypothesis or recommendation. | Develops a proof-sketch in support of a theorem. |
| Fills in gaps and refines the argument. | Fills in gaps in the sketch. |
| Issues a written report expressing and defending the argument. | Releases an "informal" proof to the community. |

**Table 3.2: The Parallel Processes**

#### 3.1.1.2 KANI

*Knowledge Associates for Novel Intelligence* (KANI) [28] is technology under joint development by the Stanford Knowledge Systems AI Lab, IBM's T.J. Watson Research Center, and the Battelle Memorial Institute. KANI supports the analytic process by helping analysts process and organize information, manual construction of hypothesis and alternative models.

The associates are modular components to be part of the analytic process. Great detail is not necessary here, but a brief summary of the individual associates and their capabilities is included [29]:

1. **Hypothesis Generation and Tracking Associate** for knowledge representation and reasoning. This associate provides a framework in which hypotheses can be constructed and challenged by alternative explanations.

2. **Massive Data Extraction and Structuring Associate** for processing and extracting information from text documents. This associate, also known as the **Knowledge Extraction Associate**, uses IBM Research's Unstructured Information Management Architecture (UIMA) [27] to extract information from *unstructured* text documents according to domain ontologies, bringing the information to the other associates in a structured encoding.

3. **Background Knowledge Identification and Assembly Associate** for collecting and organizing background information pertaining to particular documents. This associate is able to process specially structured resources on the WWW that contain domain-specific knowledge on various topics.

4. **Information Interaction Associate** to ease analyst interaction with the other associates. This associate is also responsible for managing the graphical workspace (Figure 3.1.1.2). Records of these visual workspaces over time, and from multiple analysts are records of analyst work, intended to be used for reflection and meta-analysis.

Though many of KANI's actions are automatic, e.g., automated inferencing, automated information extraction from text, KANI was designed so as to make

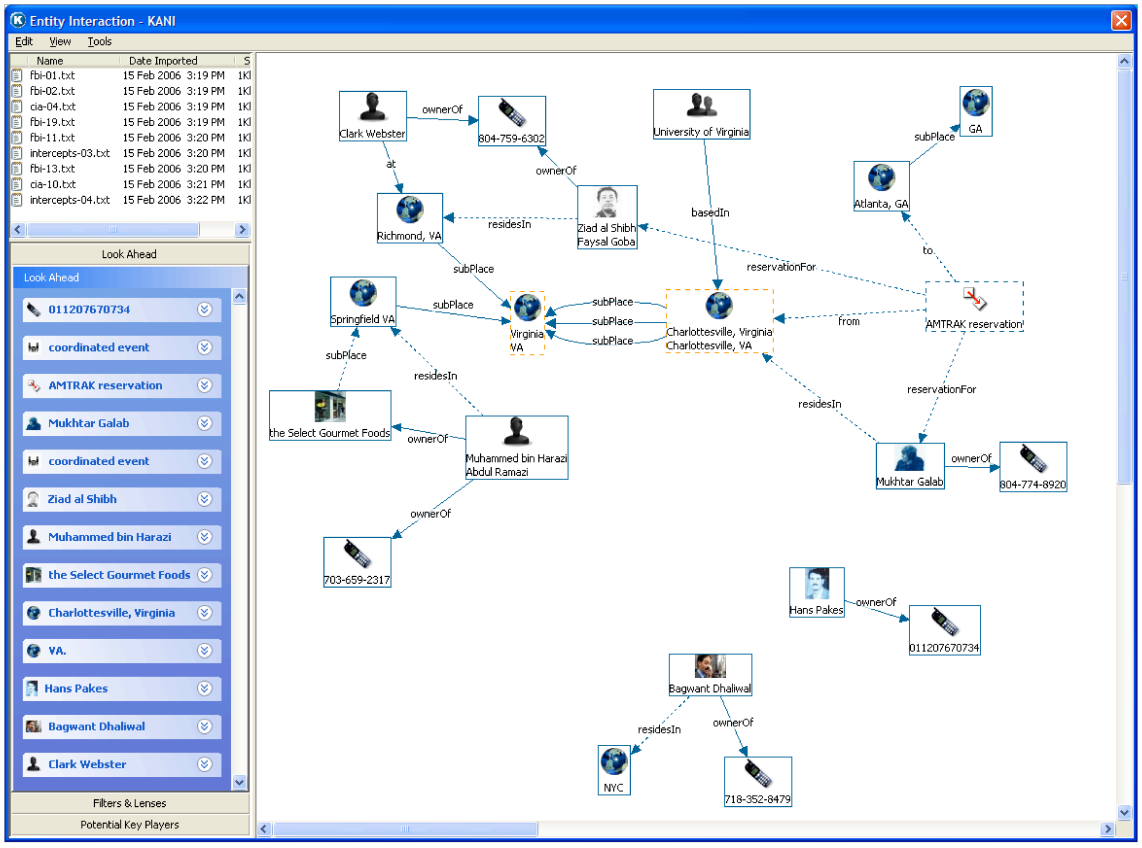


**Figure 3.1: An hypothesis in KANI. Relations and connections between objects are depicted graphically, similarly to Conceptual Graphs.**

the justifications for any automatic actions, particularly inferences, available to the analyst. Treating the information from the text extraction process as a series of inferences allows the this process to be described to analysts [26, 51].

#### 3.1.1.3 Noöscape

Cycorp's Noöscape [69] is designed to provide information retrieval and question answering technology to IAs. Built upon Cyc's knowledge base and inference engine, Noöscape receives queries from analysts in natural English, and attempts to answer the questions deductively. When a deductive answer can be constructed, the corresponding proof, or argument structure, is rendered in English and given as a response. Noöscape, using Cyc's information provenance, is able to cite the sources and authors of each piece of information in the argument.

When Noöscape is unable to answer a query deductively, it attempts to respond with an argument generated *abductively*. A partial argument for an answer is generated, and Noöscape *abduces* the missing information. When the final argument is presented to the analyst, the abduced information is marked as such and the analyst can choose to reject the abduced premises, mark them as accepted, or launch a targeted investigation into them.

During the development of Noöscape, Cycorp was also working on the *Terrorist Knowledge Base*™(TKB) [23], a comprehensive knowledge base containing information about terrorism, terrorist groups, terrorists, etc. The TKB provides the Cyc knowledge base with the information needed to answer difficult questions about terrorism.

### 3.1.2 Interchange Formalism

The interchange formalism for the IKRIS program is an architecture in which a number of software systems (knowledge representation and reasoning tools, information gathering tools; henceforth *end systems*) exchange information by *translating* information from the ontology of one system into the ontology of another. The process of translation is governed by bridging axioms which, at the very least, place limits on the outputs of a translator based on its input.

To achieve interoperability amongst a set of systems sufficiently expressive, between any two systems $i$ and $j$, there is a translation function $\mathsf{T}_{i,j}$ which takes as input sentences in $i$ and produces as output sentences in $j$. (We will also use the notation $\mathsf{T}_{i,j}(\Phi)$ where $\Phi$ is a *set* of formulae in $i$. This is defined as $\{\mathsf{T}_{i,j}(\phi) : \phi \in \Phi\}$.)

However, building a translator between each pair of the systems is not practical, and the number of translators needed grows quadratically with the number of systems. This is discouraging, particularly when the goal is to increase the ease and the speed, and to decrease the effort, with which a new system can be integrated into the set of already-interoperating systems. However, by adding a single dedicated ontology as an intertheory with which each of the component systems can interact, the amount of work required to integrate a new system can be made constant; the

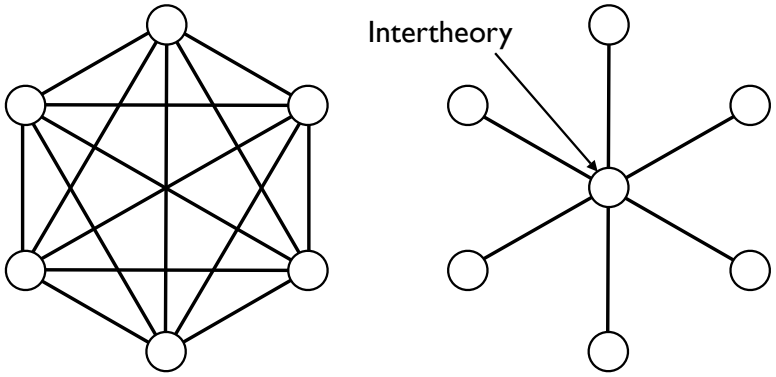


**Figure 3.2: Solid lines denote bidirectional translators. When no intertheory is used, complete interoperability necessitates $O(n^2)$ translators. When a common intertheory is used between $n$ systems, the number of translators needed is $O(n)$.**

| | | Source | | |
|---|---|---|---|---|
| | | Slate | Noöscape | KANI |
| Target | Slate | $\mathsf{T}_{\mathsf{I},\mathsf{S}} \circ \mathsf{T}_{\mathsf{S},\mathsf{I}}$ | $\mathsf{T}_{\mathsf{I},\mathsf{S}} \circ \mathsf{T}_{\mathsf{N},\mathsf{I}}$ | $\mathsf{T}_{\mathsf{I},\mathsf{S}} \circ \mathsf{T}_{\mathsf{K},\mathsf{I}}$ |
| | Noöscape | $\mathsf{T}_{\mathsf{I},\mathsf{N}} \circ \mathsf{T}_{\mathsf{S},\mathsf{I}}$ | $\mathsf{T}_{\mathsf{I},\mathsf{N}} \circ \mathsf{T}_{\mathsf{N},\mathsf{I}}$ | $\mathsf{T}_{\mathsf{I},\mathsf{N}} \circ \mathsf{T}_{\mathsf{K},\mathsf{I}}$ |
| | KANI | $\mathsf{T}_{\mathsf{I},\mathsf{K}} \circ \mathsf{T}_{\mathsf{S},\mathsf{I}}$ | $\mathsf{T}_{\mathsf{I},\mathsf{K}} \circ \mathsf{T}_{\mathsf{N},\mathsf{I}}$ | $\mathsf{T}_{\mathsf{I},\mathsf{K}} \circ \mathsf{T}_{\mathsf{K},\mathsf{I}}$ |

**Table 3.3: The compound translators produced by the composition of translators between the intertheory and end systems.**

number of translators required overall is $O(n)$ [53]. This is depicted in Figure 3.2.

As a result, the decision was made to use a dedicated IKRIS Intertheory, and build axiomatically governed translators that would translate from an end system to the intertheory, and from the intertheory to the end system. In general, a translator from system $x$ to system $y$ will be denoted $\mathsf{T}_{x,y}$, and so the requirement is that for each system $\sigma$ in the workshop, the two translators $\mathsf{T}_{\sigma,\mathsf{I}}$ and $\mathsf{T}_{\mathsf{I},\sigma}$ would be built.

Because they are functions, translators can be composed. For every participating system in the workshop, two translators had been built, and so end-to-end translators are simply composed translators. An end-to-end translator for systems $x$ and $y$ is the composition $\mathsf{T}_{\mathsf{I},y} \circ \mathsf{T}_{x,\mathsf{I}}$. Table 3.3 gives all the definitions of end to end translators for the systems in the IKRIS workshop. To denote end systems, we use the subscripts $\mathsf{S}$, $\mathsf{K}$, $\mathsf{N}$, and $\mathsf{I}$, to denote, respectively, Slate, KANI, Noöscape, and the Intertheory.

The appearance of translators of the form $\mathsf{T}_{x,x}$ may at first be surprising, but

they were critical in the evaluation of their component translators. Several important constraints on translators can be expressed using these types of translators. These and other constraints are discussed with the evaluations (§3.1.4).

### 3.1.3 IKL and the Intertheory

#### 3.1.3.1 IKL

While languages such as KIF (§2.1.1.3) and Common Logic (§2.1.1.4) had been developed to facilitate interoperability and knowledge exchange, these were designed for systems working in more traditional academic and AI domains; more flexibility and greater expressivity was desired. Building on the work that shaped KIF and CL, the IKRIS Workshop produced *IKL*, the IKRIS Knowledge Language. IKL is an extension to CL which adds the "ability to talk about the proposition that its own sentences express" [39]. Though any first-order language theoretically has this capability (through formalizing the language within itself), IKL adds this capability at a useful and accessible level of abstraction, and in a way that facilitates expressing the *context* of sentences. IKL would also serve as the language in which the intertheory would be written.

#### 3.1.3.2 The IKRIS Intertheory

Within the IKRIS Workshop, a number of working groups were formed, each working on some facet of the workshop's goals. The IKRIS Scenarios Working Group, approached the task of developing an intertheory general and expressive enough to capture the notions that software for the IC would need to represent, and yet also so concise that application programmers would be able, and willing, to use it [41].

The intertheory, then, is an ontology targeted at knowledge representation for the intelligence community. The intertheory incorporated existing ontologies, such as Process Specification Language (PSL), and [65, 66] and OWL-Time [43, 42], introduced new terms for scenario descriptions, and related these terms axiomatically (in IKL). Two examples, Figures 3.3 and 3.4 are reproduced from [41].

The scenarios groups also participated in the creation of the bridging axioms that related the ontologies of the systems in IKRIS to the new intertheory. Most

```
(forall (e f)
  (if (fluentFor f e)
      (and (psl:fluent f)
           (State e))))

(forall (e)
  (if (State e)
      (exists (f)
        (fluentFor f e))))
```

**Figure 3.3: PSL's `fluent` is related to the intertheory's `State` and `FluentFor`. PSL terms are prefixed with `psl:`.**

```
(forall (f t t1)
  (if (t:begins t1 t)
      (iff (holdsFor f t)
           (exists (o1)
             (and (psl:activity_occurrence o1)
                  (psl:holds f o1)
                  (t:before/= (psl:end_of o1) t1)
                  (not (exists (o2)
                         (and (psl:activity_occurrence o2)
                              (psl:falsifies o2 f)
                              (t:before/= (psl:end_of o1)
                                          (psl:end_of o2))
                              (forall (t2)
                                (if (t:ends t2 t)
                                    (t:before (psl:end_of o2) t2))))))))))
```

**Figure 3.4: A complex axiom relating terms from PSL and OWL-Time, which are prefixed with `psl:` and `t:`, respectively.**

bridging axioms were rather concise, as exemplified by Figure 3.5, also directly from [41].

### 3.1.4 Evaluation

The IKRIS program was evaluated with respect to several round trip information exchange challenges, and in exchanging information pertinent to a Joint Military Intelligence College Case Study. The evaluations aimed to determine whether or not:

1. The ontologies were sufficiently expressive to capture the semantic constructs of their peers'.

```
(forall (x)
  (iff (cyc:isa x cyc:StaticSituation)
       (State x)))

(forall (x)
  (iff (cyc:genls x cyc:StaticSituation)
       (StateType x)))
```

**Figure 3.5: Two bridging axioms relating the intertheory's `State` and `StateType` to terms in the Cyc KB. Cyc terms are prefixed with `cyc:`.**

2. The translators could be built to satisfy Equations 3.1, 3.2, and 3.3.

3. There was a language expressive enough to capture the relationships between the ontologies of the systems in the IKRIS program (and, it was hoped, other ontologies used by applications used within the intelligence community).

Several evaluations, called *round trips* for their general structure, focused on technical aspects of the translators, and determined whether or not the translators met specific constraints on the quality of translation. One large evaluation, the Capstone Demo, was less formal in nature, and demonstrated that the translators and the interchange formalism could be implemented and applied to a real Case Study in use by the Joint Military Intelligence College.

#### 3.1.4.1 Round Trips

Given a system $\sigma$, the two translation functions $\mathsf{T}_{\sigma,\mathsf{I}}$ and $\mathsf{T}_{\mathsf{I},\sigma}$ are not required to be inverses, and so the composed translation function $\mathsf{T}_{\sigma,\sigma} = \mathsf{T}_{\mathsf{I},\sigma} \circ \mathsf{T}_{\sigma,\mathsf{I}}$ is not an identity function on the sentences in $\sigma$. As a result, there are several constraints that translation functions should satisfy.

**Preservation of Deductive Implication.** When some formula, $\psi$ is deducible from a set of formulae, $\Phi$, deductive implication should be preserved when both $\Phi$ and $\psi$ are translated into the intertheory and back into $\sigma$. This is formally expressed as

$$\forall_{\psi \in \sigma, \Phi \subseteq \sigma} \; \Phi \vdash \psi \text{ iff } \mathsf{T}_{\mathsf{I},\sigma} \circ \mathsf{T}_{\sigma,\mathsf{I}}(\Phi) \vdash \mathsf{T}_{\mathsf{I},\sigma} \circ \mathsf{T}_{\sigma,\mathsf{I}}(\psi) \tag{3.1}$$

**Preservation of Semantic Domain and Co-Domain.** Equation 3.1 does not address the issue that $\mathsf{T}_{\sigma,\sigma}$ might perform a decrease the size of the projection into $\sigma$'s ontology; that is, the co-domain of $\mathsf{T}_{\sigma,\sigma}$ might use a strict subset of the vocabulary of $\sigma$'s ontology. (E.g., perhaps $\sigma$'s ontology describes integer and rational arithmetic, but $\mathsf{I}$ describes only rational arithmetic, and $\mathsf{T}_{I,\sigma}$ generates formulae which use only the vocabulary for the rationals.) Two constraints are necessary to guarantee that the semantic domains and co-domains are preserved,

$$\forall_{\psi\in\sigma,\Phi\subseteq\sigma}\ \Phi\vdash\psi \quad \text{iff} \quad \mathsf{T}_{\mathsf{I},\sigma}\circ\mathsf{T}_{\sigma,\mathsf{I}}(\Phi)\vdash\psi \tag{3.2}$$

$$\forall_{\psi\in\sigma,\Phi\subseteq\sigma}\ \Phi\vdash\psi \quad \text{iff} \quad \Phi\vdash\mathsf{T}_{\mathsf{I},\sigma}\circ\mathsf{T}_{\sigma,\mathsf{I}}(\psi) \tag{3.3}$$

Equation 3.2 ensures that consequences of a knowledge base are maintained when the knowledge base is translated. Equation 3.3 ensures that consequences of a knowledge base are maintained when the consequences are translated.

The issue did not arise in the IKRIS workshop, but it is conceivable that in dealing with certain types of systems, Equations 3.2 and 3.3 might be relaxed:

$$\forall_{\psi\in\sigma,\Phi\subseteq\sigma}\ \Phi\vdash\psi \quad \text{only if} \quad \mathsf{T}_{\mathsf{I},\sigma}\circ\mathsf{T}_{\sigma,\mathsf{I}}(\Phi)\vdash\psi \tag{3.4}$$

$$\forall_{\psi\in\sigma,\Phi\subseteq\sigma}\ \Phi\vdash\psi \quad \text{only if} \quad \Phi\vdash\mathsf{T}_{\mathsf{I},\sigma}\circ\mathsf{T}_{\sigma,\mathsf{I}}(\psi). \tag{3.5}$$

These relaxed forms would allow for the translation process to effectively *add* information. This may be an analogue to the manner in which human reasoners seem to *gain* information through interchange, perhaps by recasting information or reasoning about the representation schemes of their peers [12].

The first round trip illustrated the compatibility between Slate and the IKRIS intertheory, and determines the quality of the two translators, $\mathsf{T}_{\mathsf{S},\mathsf{I}}$ and $\mathsf{T}_{\mathsf{I},\mathsf{S}}$. The structure of this round trip is illustrated in Figure 3.6. This round trip showed that the translators satisfy Equation 3.1.

Experience with the output of the translators strongly suggested that they also satisfied Equations 3.2 and 3.3. In general, proof that a translator, not provability-based, satisfies these constraints would be infeasible for all but the simplest translators.

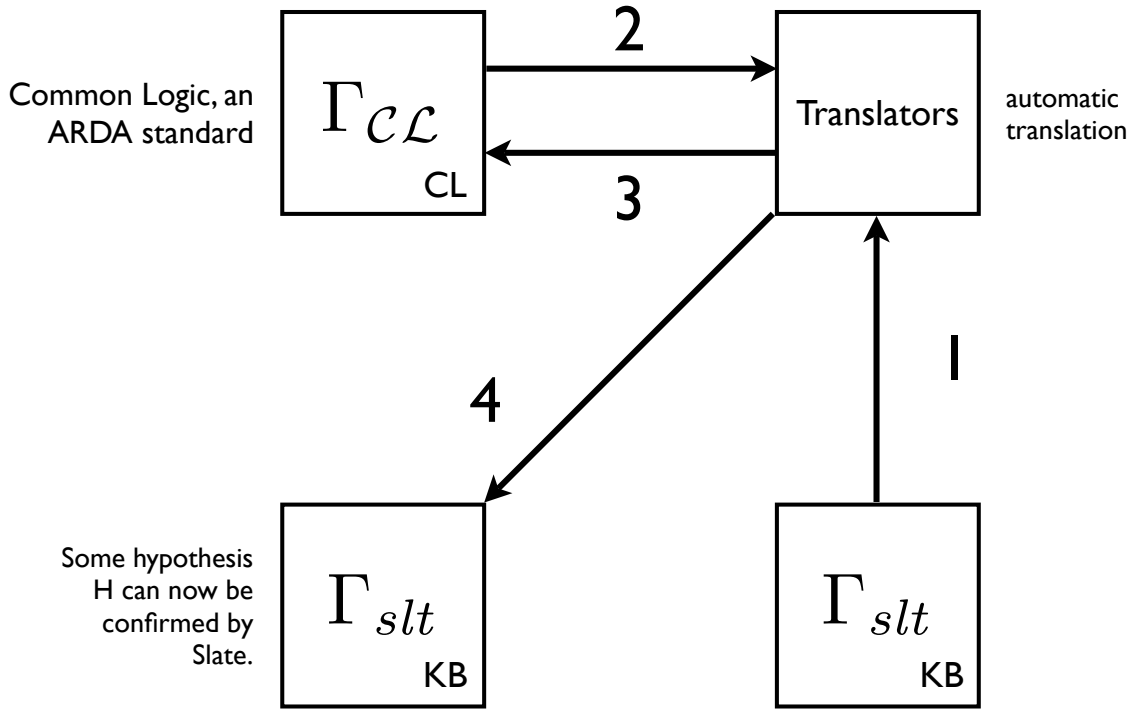


**Figure 3.6: In the first round trip evaluation, knowledge from a knowledge base $S$ in Slate was automatically translated to Common Logic. The resulting Common Logic knowledge base was automatically translated back into the Slate format. Slate successfully answered the queries described by Equations 3.1, 3.2, and 3.3.**

When working with multiple systems, however, the constraints of Equations 3.2 and 3.3 cannot be applied directly, for they reference entailment relationships between formulae in the same system. As a result, Equations 3.2 and 3.3 become

$$\forall_{\psi,\Phi}\ \Phi_\sigma \vdash \psi_\sigma \quad \text{iff} \quad \mathsf{T}_{\mathsf{I},\rho} \circ \mathsf{T}_{\sigma,\mathsf{I}}(\Phi_\sigma) \vdash \psi_\rho \tag{3.6}$$

$$\forall_{\psi,\Phi}\ \Phi_\sigma \vdash \psi_\sigma \quad \text{iff} \quad \Phi_\rho \vdash \mathsf{T}_{\mathsf{I},\rho} \circ \mathsf{T}_{\sigma,\mathsf{I}}(\psi_\sigma). \tag{3.7}$$

where $\Phi$ is a set of propositions and $\psi$ is a proposition, and subscripts on $\Phi$ and $\psi$ denote their encodings in the subscripted system. Such propositions might be the result of encoding an English report in an end-system.

With an initial, but tentative, confirmation that the notion of axiomatic translation was feasible, other translation exercises could begin. These were variants of the first round trip, but were, properly speaking, one way trips between systems in

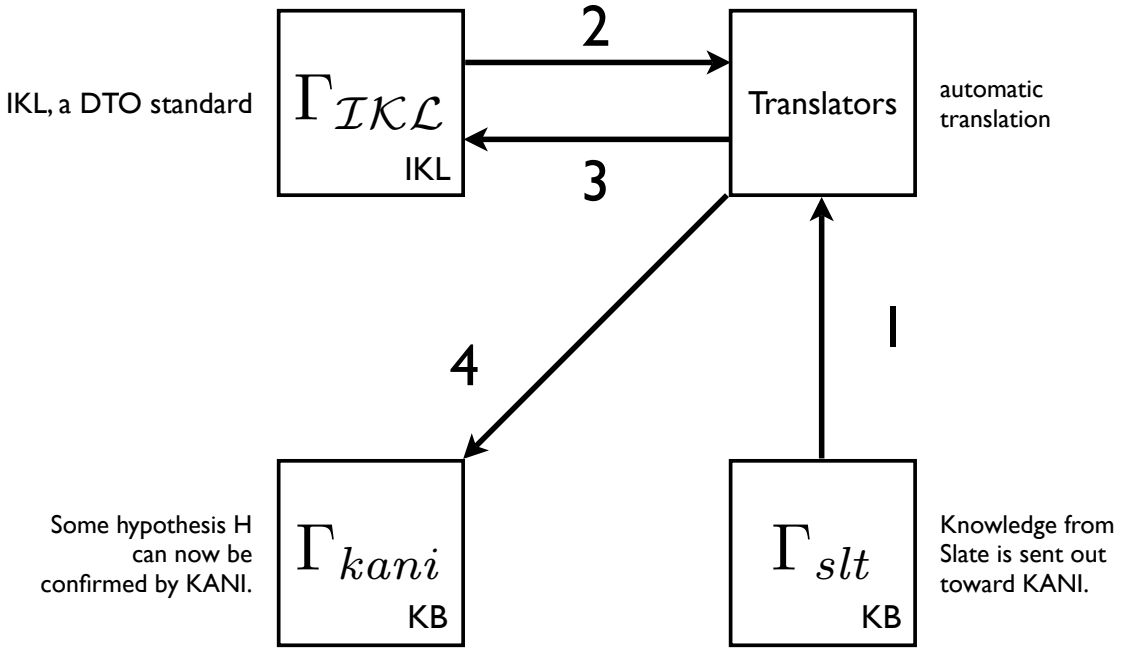


**Figure 3.7: The second round trip demonstration illustrated that knowledge translation between systems using different ontologies is feasible. KANI was able to use an automatically translated Slate knowledge base to confirm an hypothesis.**

the IKRIS program. One such exercise is illustrated in Figure 3.7.

After more translations between end systems, the translators were determined suitable for more serious use, and work began (or rather, continued in the *IKRIS Capstone Demo*).

#### 3.1.4.2 IKRIS Capstone Demo

The IKRIS Workshop was sponsored because of the promise offered by information exchange. The interchange formalism described how, and provided a framework in which, information could be exchanged. The Capstone Demo served as an illustration of the benefits offered analysts by interoperability and information exchange. The Capstone Demo tracks the progress of an analyst working on *Chart H*, a portion of Hughes' *Case Study Four: Sign of the Crescent*. The analyst is able to solve Chart H when the knowledge bases of the participating systems can be transparently shared, and the features of the systems can be applied to information in the other systems. A high-level choreography of the demo is provided in Figure

3.8.

The IKRIS Capstone Demo begins as an analyst using KANI has received information regarding phone calls between five individuals, viz. Ramazi, Pakes, Goba, Dhaliwal, and Galab, who are suspected of nefarious behavior. The phone calls seem to refer to an event, henceforth $E$, a meeting of some sort, and the analyst considers the most straightforward hypothesis, $H1$, that the phone calls do, in fact, refer to a meeting of the four suspects. The analyst sends the hypothesis to Noöscape asking whether a meeting between the suspects at the proposed time and meeting place is actually plausible.

Noöscape considers the hypothesis but abduces, using some knowledge privy to its own knowledge base as well as common sense reasoning, $H2$, that one of the participants, Pakes, of the meeting actually will be ship-bound in an harbor in Boston, lacking the paperwork and credentials to enter the country. Noöscape constructs the argument explaining its reasoning and sends this argument back to KANI, and also to Slate.

Slate examines Noöscape's argument for $H2$, to look for any weak points or other potential problems with the claim that Pakes cannot leave his vessel. Using some of its own background knowledge, Slate presents a *countermodel* in which Pakes has *forged* documents and so might be able to, albeit illegally, leave the boat. Now, Noöscape's argument contained abductive steps, and so Slate's countermodel does not conclusively strike down the possibility described by Noöscape, but does give some cause for the analyst to reconsider, so Slate sends the countermodel to the analyst using KANI.

The analyst using KANI examines Noöscape's argument as well as the alternative countermodel produced by Slate, but decides that Slate's countermodel, while feasible, is probably implausible, and that Pakes probably cannot leave the ship. The analysts then asks Noöscape for an entire hypothesis based on $H2$, that is, to elaborate on the hypothetical that Pakes cannot leave the ship.

Noöscape, expanding on the $H2$ generates the hypothesis $H4$ that the event discussed in the phone communications is not actually a meeting, as Pakes seems to be a participant of $E$, and that the participants of events which are meetings are

usually able to attend the meeting. Noöscape sends this argument to the analyst as well as Slate.

Rather than generating another countermodel, Slate consults some information in its knowledge base and realizes that not only (according to Noöscape's argument) can Pakes not attend the meeting, but that neither can Goba or Galab.

Attempting to flesh out the hypothesis, the analysts refines his hypothesis and decides to focus some attention on the individual suspects. Noöscape happens to know that Ramazi is actually an alias for a character named Ramzi who also happens to be mentioned in KANI's knowledge base.

The now enlightened analyst consults KANI concerning Ramzi and learns that he maintains a warehouse about which little is known. Slate just happens to have on hand the information that explosives were recently uncovered in Ramzi's warehouse.

The now quite concerned analyst is coming to the realization that, with very high likelihood, references to $E$ are actually indirect talk about a planned bombing.

## 3.2 With Oculus' GeoTime

As we began work in the DTO-sponsored *Advanced Representation for Interactive Visualization* (ARIVA), now *ASpace-X*, program with the RAIR Lab's *Advanced Knowledge Representation and Reasoning for Interactive Visualization* (AKRRIV) technology, and IKL's fame grew in the R&D segment of the intelligence community, we investigated the possibility of interoperability work with Oculus' GeoTime [63, 44, 17]. This section describes the IKL-based ongoing work to enable interoperability between Slate and GeoTime.

### 3.2.1 GeoTime

Oculus's GeoTime system presents geospatial and temporal information with an animated three-dimensional visualization. This method of visualization has cognitive benefits that enables users to more quickly grasp trends in the presented data, and to efficiently receive larger quantities of information with a smaller cognitive workload. GeoTime's visualizations aid analysts in uncovering behavioral patterns of entities and and in predicting their future actions [45].

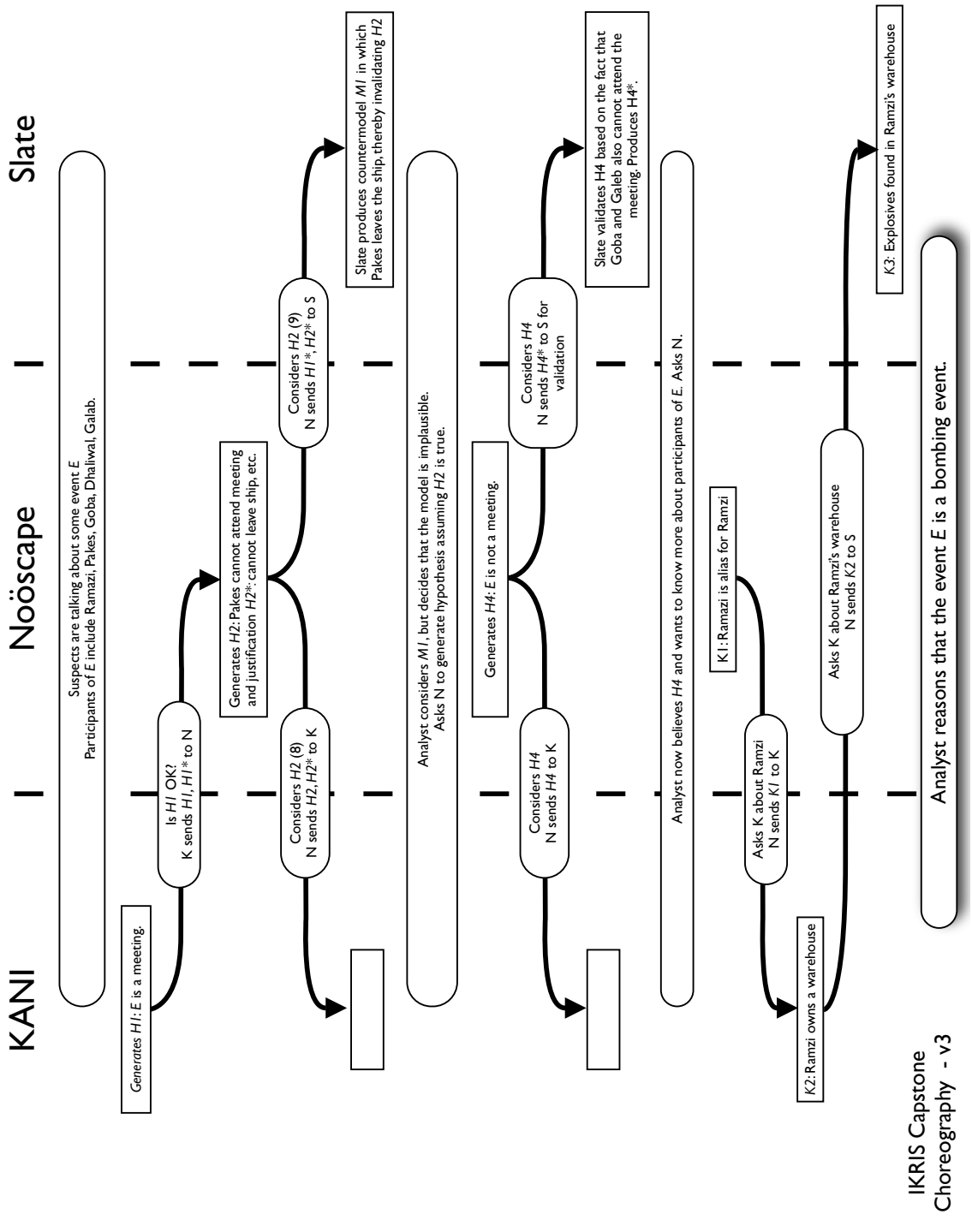


**Figure 3.8: The IKRIS Capstone Demo high level choreography.**

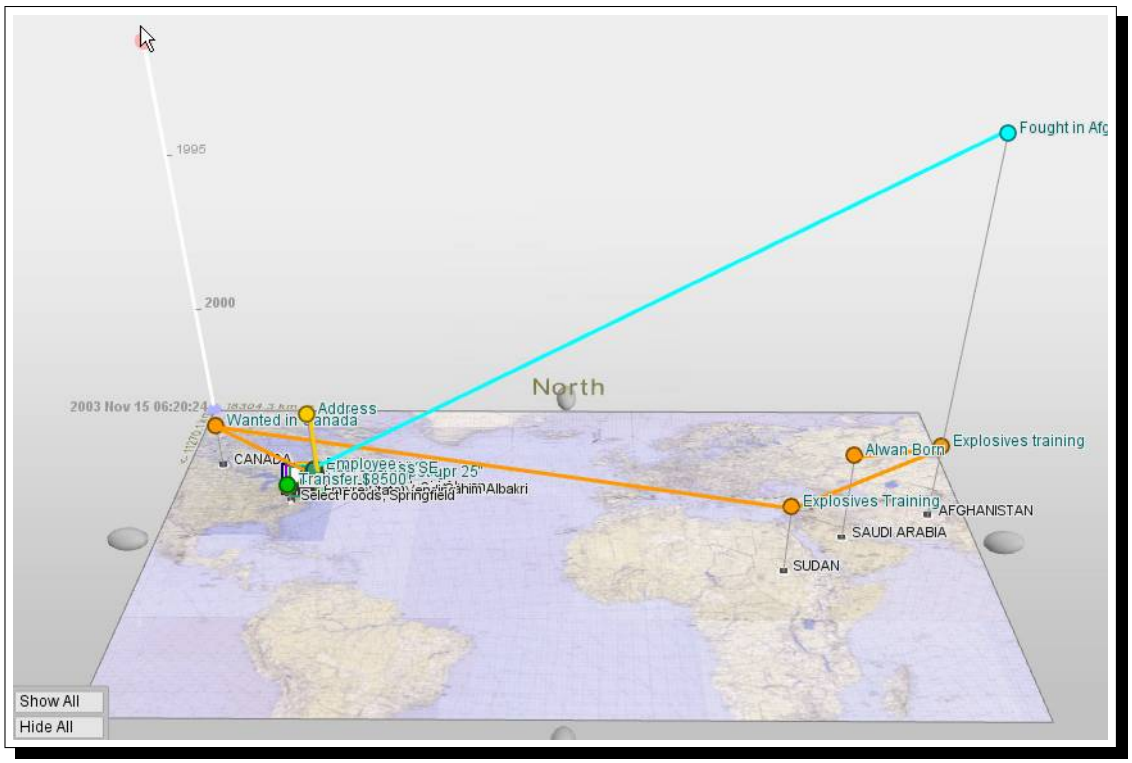


**Figure 3.9: Display of information in GeoTime.**

GeoTime has a relatively simple, but effective data model in which every object is either an **entity**, **location**, **event**, or **association**. The **entity** type corresponds, roughly, to actors. An **association** associates other kinds of objects; e.g., an entity participating in an event would be connected to the event with an association [47, 46].

### 3.2.2 Interoperability

Translators were built by researchers at Pacific Northwest National Laboratories (PNNL) working on *Visualization and Interoperable Knowledge Representation Services* (VIKRS), another project in ASpace-X. GeoTime's native storage format, OGT, is XML, specified by a custom DTD. The PNNL translators converted the XML into "OGT-flavored IKL" which was then related to an intertheory designed for this interoperability. Specifically, this intertheory was not the IKRIS Intertheory. PNNL's translators took XML of the type shown in Figure 3.10, and produced IKL of the form shown in Figure 3.11.

```
<Scenario>
  <Target
    name="John Doe"
    id="1234"
    location="3456"
    display="Person"/>
  <Location
    name="Washington, DC"
    id="3456"
    display="City"/>
</Scenario>
```

**Figure 3.10: GeoTime's XML, OGT. For confidentiality reasons, actual OGT is not shown. This captures the general structure of OGT without using its vocabulary.**

```
(ogt:type "1234" "Target")
(ogt:name "1234" "John Doe")
(ogt:location "1234" "3456")
(ogt:display "1234" "Person")
(ogt:type "3456" "Location")
(ogt:name "3456" "Washington, DC")
(ogt:display "3456" "City")
```

**Figure 3.11: "OGT-flavored" IKL. Individual attributes of the XML elements have been specified using binary relations. Bridging axioms specify how the IDs correspond to objects in the intertheory with particular attributes.**

Appendix A contains materials relating to the GeoTime interoperability effort. In particular, `axioms.ikl` (§A.1) contains the axioms that governed the behavior of the provability-based translators constructed at RPI that converted formulae from the OGT-flavored IKL into the intertheory. `hypothesis.ikl` (§A.2) is an hypothesis generated by Slate (Figure 3.12) working on a sample scenario imported from GeoTime. `preserved-information.ikl` (§A.3) contains assertions about objects with symbolic names in Slate which were identified by global unique identifiers in GeoTime.

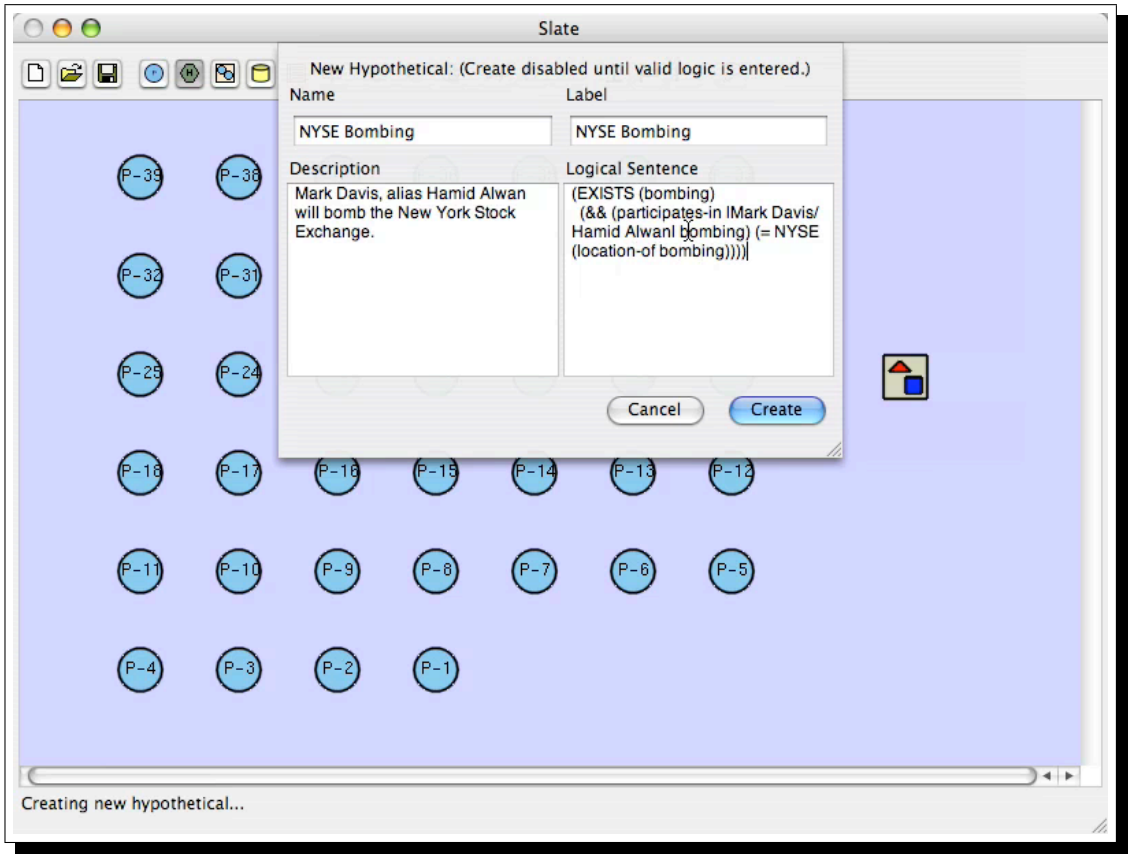


**Figure 3.12: A hypothesis is generated in Slate using the information imported from GeoTime.**

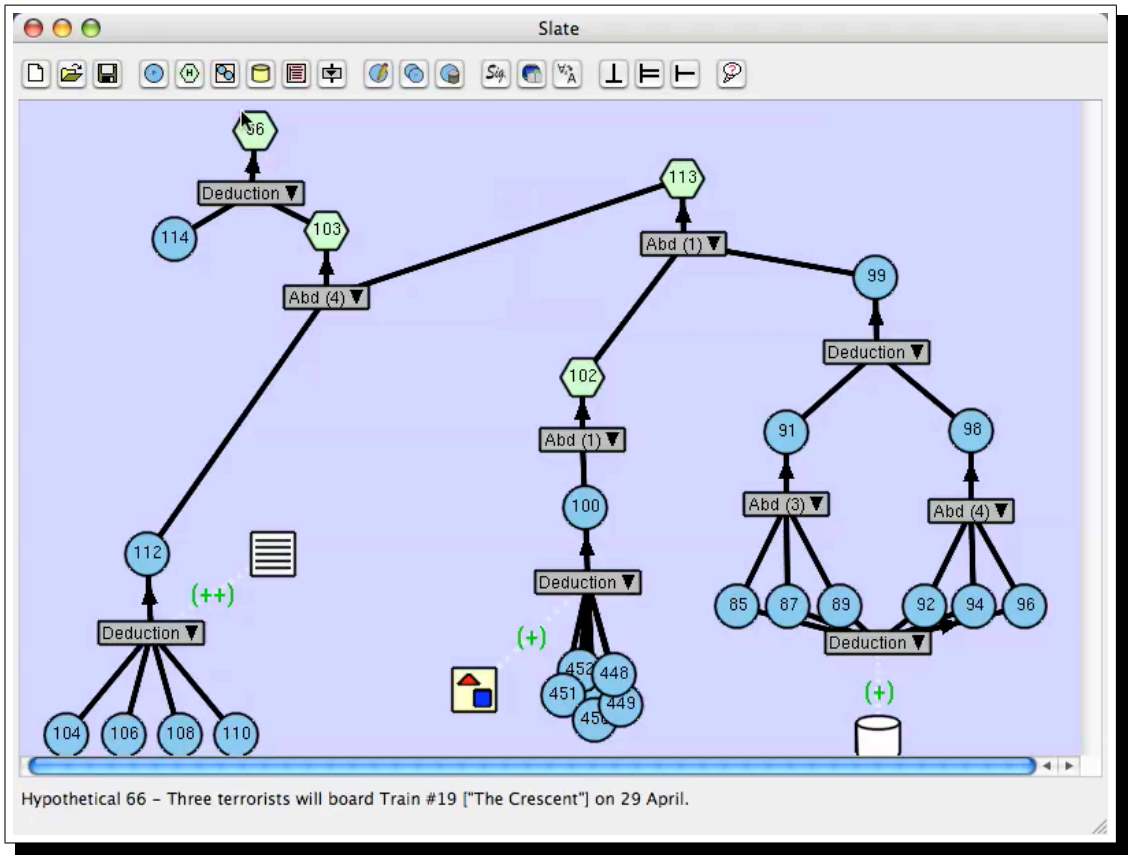


**Figure 3.13: GeoTime provided information from Hughes' *Case Study Four: Sign of the Crescent*, allowing a user of Slate to crack Chart E. The cluster of propositions at the bottom center of the screen corresponds to the imported GeoTime scenario.**

# CHAPTER 4
# Translation Graphs

During the various interoperability experiments (§3), we refined the techniques with which axiomatic interoperability was described and implemented.

In the IKRIS workshop (§3.1), bridging axioms expressed in IKL were the primary enabler of interoperability. The bridging axioms, constructed *manually* by workshop participants, related the end system ontologies to the intertheory. Translators were also built manually, but their behavior was governed by the bridging axioms.

In the GeoTime work, we decided to use the same general approach, but started to look for ways to generate bridging axioms, at least in part, *automatically*. We realized that the ontologies of most systems, regardless of their underlying formalism (e.g., description logics, propositional calculus, first-order logic, etc.), can be described as *signatures* in many-sorted logic. The relationships between these can be described, and from the description, bridging axioms can be extracted *automatically*.

This chapter describes *translation graphs*, their use in relating the ontologies of multiple systems, and how bridging axioms and, in some cases, translators can be extracted from them in an automated fashion. The contents of this chapter draw heavily from a publication of this research, *Provability-Based Semantic Interoperability via Translation Graphs* [74].

## 4.1 Formal Preliminaries

We treat ontologies as pairs of the form $\langle \Sigma, \Phi \rangle$ where $\Sigma$ is a signature in a many-sorted logic, and $\Phi$ is a set of sentences in $\Sigma$. While many-sorted logic is not employed by all ontology designers, it is appropriate for describing many ontological constructs, including modalities, has an impressive history within computer science and mathematics, and is reducible to standard first-order logic [50].

A *sort* is a domain, a universe, or a set of objects. There is a global set

of sorts, $S^*$. Generally, every signature will contain a sort corresponding to truth values. In traditional logics, this sort is the set $\{\mathbf{true}, \mathbf{false}\}$, but this needn't be the case. Many-valued logics, for instance, will use a different sort for truth values. A *functor* $f$ is a function $s_0 \times \ldots \times s_{n-1} \rightarrow s_n$ where $s_0, \ldots, s_n$ are elements of $S^*$. $\langle [s_0, \ldots, s_{n-1}], s_n \rangle$ is the *rank* of $f$ and denoted $Rank(f)$.

A *signature* $\Sigma$ is a tuple $\langle \sigma, \phi \rangle$ where $\sigma$ is a subset of $S^*$, called the sorts of $\Sigma$ and $\phi$ is a partial injective function from string-rank pairs to functors of the same rank. The range of $\phi$ is the set of functors of $\Sigma$. There is a restriction on $\phi$ that for every functor $f$ among $\Sigma$'s functors, each sort in $f$'s rank is one of $\Sigma$'s sorts.

A well-formed term of $\Sigma$ has a particular *interpretation* which denotes the application of corresponding functors to their arguments. E.g, if man(Socrates) is a sentence of $\Sigma_1$ and human(Sokrates) is a sentence of $\Sigma_2$, but both $\Sigma_1$ and $\Sigma_2$ map man and human, respectively, to the same functor $f$ of rank $\langle [s_1], s_0 \rangle$, and Socrates and Sokrates to the same functor $g$ of rank $\langle [], s_1 \rangle$, then the two sentences have the same interpretation.

## 4.2 Ontology Modifications

A number of operations can be defined on signatures which correspond to incremental modifications that might be performed on the signatures of ontologies. Four primitive operations on signatures are defined by the following equations

$$AddSort(s, \langle \sigma, \phi \rangle) = \langle \sigma \cup \{s\}, \phi \rangle \quad (4.1)$$

$$RemoveSort(s, \langle \sigma, \phi \rangle) = \langle \sigma - \{s\}, \phi \rangle \quad (4.2)$$

$$AddFunctor(w, f, \langle \sigma, \phi \rangle) = \langle \sigma, \phi \cup \{\langle \langle w, Rank(f) \rangle, f \rangle\} \rangle \quad (4.3)$$

$$RemoveFunctor(w, r, \langle \sigma, \phi \cup \{\langle \langle w, r \rangle, f \rangle\} \rangle) = \langle \sigma, \phi \rangle \quad (4.4)$$

subject to several restrictions. $RemoveSort(s, \langle \sigma, \phi \rangle)$ is undefined if any of the functors of $\langle \sigma, \phi \rangle$ use $s$. $AddFunctor(w, f, \langle \sigma, \phi \rangle)$ is undefined if $\langle w, arity(f) \rangle$ is already mapped to some functor. $RemoveFunctor(w, r, \langle \sigma, \phi \rangle)$ is undefined if $\phi$ does not map $\langle w, r \rangle$ to any functor.

With the primitive methods, simple ontologies can be constructed that spec-

ify only the vocabulary of a language. However, ontology consists not only in vocabulary, but also in the *meaning* of the vocabulary and the relationships among these terms. As a result, many knowledge representation languages include forms analogous to Athena's [6] `define-symbol` for defining symbols *axiomatically*. For instance, $\mathsf{MatGrandmotherOf}(x)$, denoting the maternal grandmother of $x$m can be defined in KIF using $\mathsf{MotherOf}(x)$ by `(deffunction MatGrandmotherOf (x) := (MotherOf (MotherOf x)))`.

Both classical mathematicians and logicians along with modern knowledge representation language designers have devoted a great amount of time to the subject of the forms that can be used in axiomatic definitions. Some definitions may be implemented as macro-like substitutions, while in other cases, the entire axiom must remain available for subsequent reasoning [31, Ch. 11].

## 4.3 Translation Graphs

We implemented a prototype of the structures and modifications described in the previous section, thereby providing a framework in which to perform natural ontology-related activities, such as ontology construction and mapping. Ontology construction becomes easy: Starting from an empty signature (i.e., a signature with no sorts or functors), existing ontologies can be recreated by adding the ontology's sorts, and then relations and function symbols. These reconstructed ontologies can then be related by adding the functors of one ontology to another with axiomatic definitions. Displaying the process graphically inspired translation graphs.

After initial experiments demonstrated the feasibility of this approach, we realized that the process could be used to describe the interoperability in the IKRIS workshop and experiments in interoperability between robust software systems, such as Oculus' GeoTime (§3.2), SUNY Albany's HITIQA [73], Attempto Controlled English [30, 11], and the RAIR Lab's own Slate (§3.1.1.1) and Solomon [13].

A *translation graph* is a directed graph whose vertices are signatures, and whose edges denote axiomatic relationships between the signatures of the graph. If signatures $\Sigma_i$ and $\Sigma_j$ are vertices of some translation graph and the edge $\langle \Sigma_i, \Sigma_j \rangle$ is in the graph, there is information associated with it that describes how information

represented in an ontology employing $\Sigma_i$ can be used in an ontology employing $\Sigma_j$. This property is transitive, and so a $\Sigma_u, \Sigma_v$ *path* contains information for using information under $\Sigma_u$ in $\Sigma_v$.

## 4.4 An Example

We present an example to show that translation graphs can be used to enable interoperability between ontologies whose subject domains intersect but are not identical, that queries can be answered with information from multiple ontologies, and that the information used to answer the query is not representable in all of the ontologies presented. (For the sake of readability and conciseness, we will ignore issues such as namespaces and the use of fundamental datatypes such as strings and numbers.)

We consider four separate software systems operating with four distinct ontologies amongst which information will be shared.

The first two systems are social networking programs which represent information about phone calls. The first system, $\mathcal{A}$, keeps records of the form $\mathsf{Called}(x, y)$ to denote that $x$ called $y$, where $x$ and $y$ are names of individuals. The second system, $\mathcal{B}$, uses $\mathsf{CalledBy}(x, y)$ to denote that $x$ was called by $y$, where $x$ and $y$ are names of individuals. $\mathcal{A}$ and $\mathcal{B}$ can be related with the primitive operations described earlier; the result is shown in Figure 4.1. The function $\mathsf{Called}$ is added to $\mathcal{B}$ with an axiomatic definition, yielding an intermediate signature. $\mathsf{CalledBy}$ is removed from the intermediate signature, resulting in $\mathcal{A}$. Tracing the path between the ontologies and collecting axioms along the way gives all the information needed to use information from one ontology in the other.

The axiomatic definition between $\mathcal{A}$ and $\mathcal{B}$ is a biconditional and could be optimized as a rewriting rule. That is, assertions in one ontology could be *rewritten* in terms of the other's vocabulary. The translation here is symmetric, and could be handled by schema-matching tools.

Next, we introduce a cellular phone company database $\mathcal{C}$ which has information about phone calls made on the cellular network, and keeps records of the form $\mathsf{Phoned}(n_1, n_2)$, where $n_1$ and $n_2$ are phone numbers between which calls have been

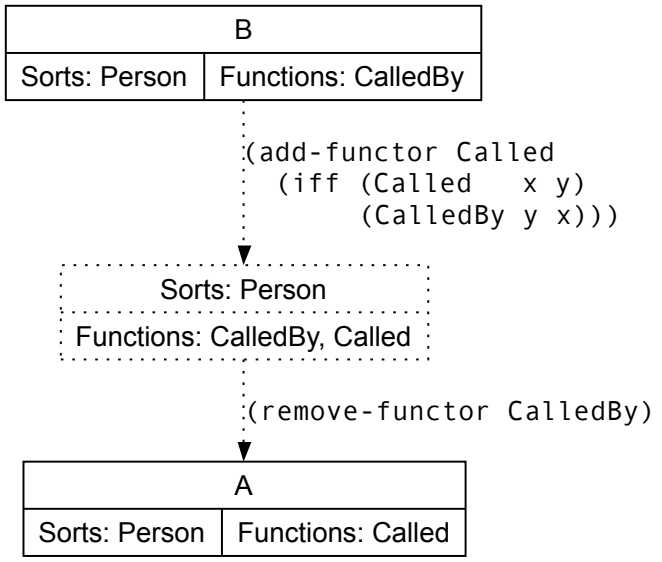


**Figure 4.1: Ontologies $\mathcal{A}$ and $\mathcal{B}$ are related.**

placed. Figure 4.2 illustrates the relationship between $\mathcal{C}$ and $\mathcal{A}$.

While no individual link in Figure 4.2 is particularly complicated, the addition of the axiomatically defined Owner deserves special note. Owner$(x)$ denotes the person who owns a phone number $x$. Owner is present in neither $\mathcal{A}$ nor $\mathcal{C}$, but its use in relating them does seem clear: Owner functions as a sort of semantic placeholder. Without an interpretation of Owner, information exchange would not be possible; there would be information missing. However, the use of translation graphs has allowed us to capture *what* is needed to exchange information meaningfully.

Another possibility is that Owner may stand for a non-logical function. For instance, in the process of exchanging information, occurrences of Owner$(x)$ might be replaced with the results of a database lookup or some procedural transformation (e.g., if phone numbers were a function of the characters comprising a person's name).

In this example, however, we integrate the database of a reverse phone number lookup system, $\mathcal{D}$. In this case, the information that $\mathcal{D}$ provides is not phone records, but pairs of phone numbers and their owners' names. $\mathcal{D}$ records that Owns$(x, y)$ when $x$, a person, owns the phone number $y$. The integration, shown in Figure 4.3, is straightforward.

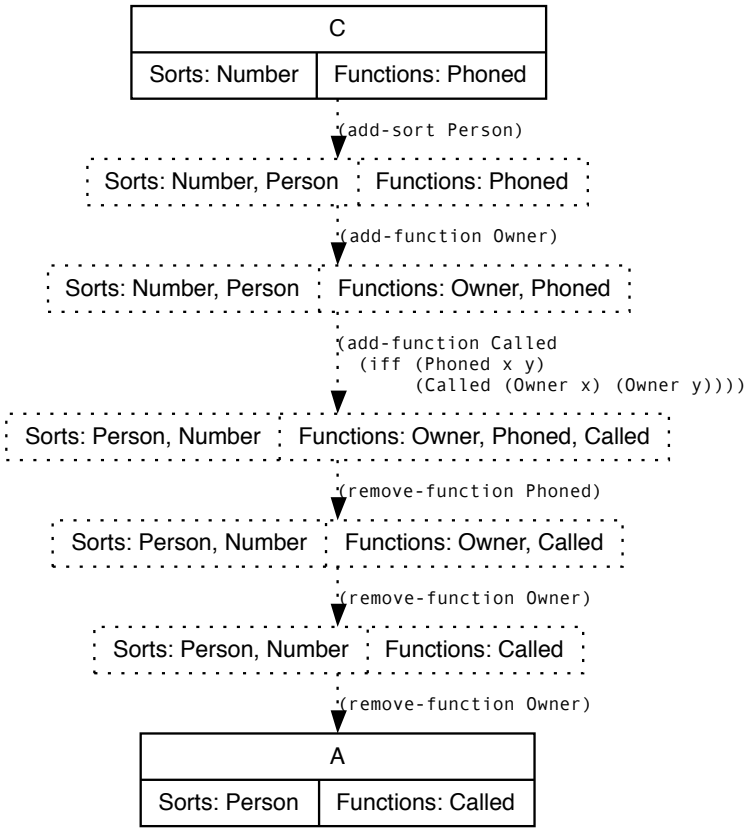


**Figure 4.2: Phone company $\mathcal{C}$ is related to $\mathcal{A}$.**

Having connected $\mathcal{A}$ with $\mathcal{B}$, and then $\mathcal{A}$, $\mathcal{C}$, and $\mathcal{D}$, enough work has been done to yield the translation graph shown in Figure 4.4. The graph can be used to describe the relationships between the ontologies, and the axiomatic relationships needed to answer queries about the contents of the four knowledge bases can be automatically extracted from it.

**Remarks.** In such a small example, the overall *structure* of the translation was not given much thought. In real systems, however, engineers must consider the implications of their translation structures. For example, in some situations, an interlingua and intertheory may be preferred, or in some cases it may not be appropriate or feasible [7, 53]. However, we present translation graphs without expressing preference among these possible architectures; translation graphs general enough to be applied in an architecture-agnostic manner.

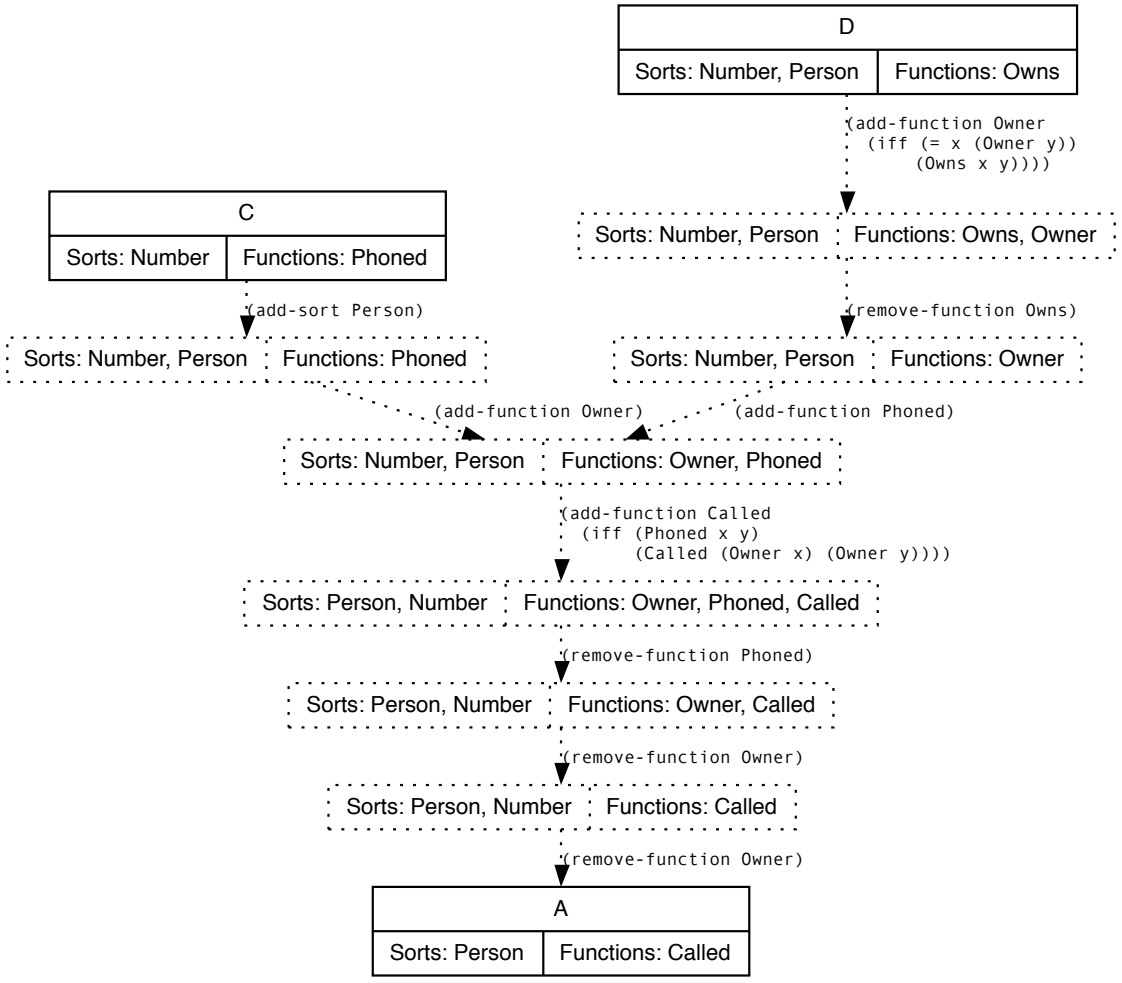


**Figure 4.3: The information in $\mathcal{D}$ is made available to $\mathcal{A}$ and $\mathcal{C}$.**

With the translation graph as given, it would be possible to run automated reasoners directly on the union of the knowledge bases and all the axioms extracted from the edges of the graph. Of course, intractability and undecidability make this a tricky technique, but there is an interesting parallel to Green's method. Green's method extracts plans that achieve particular goals from proofs that such plans exist [32]; with the naive method above, interoperability and translation are achieved as a *side effect* of automated theorem proving.

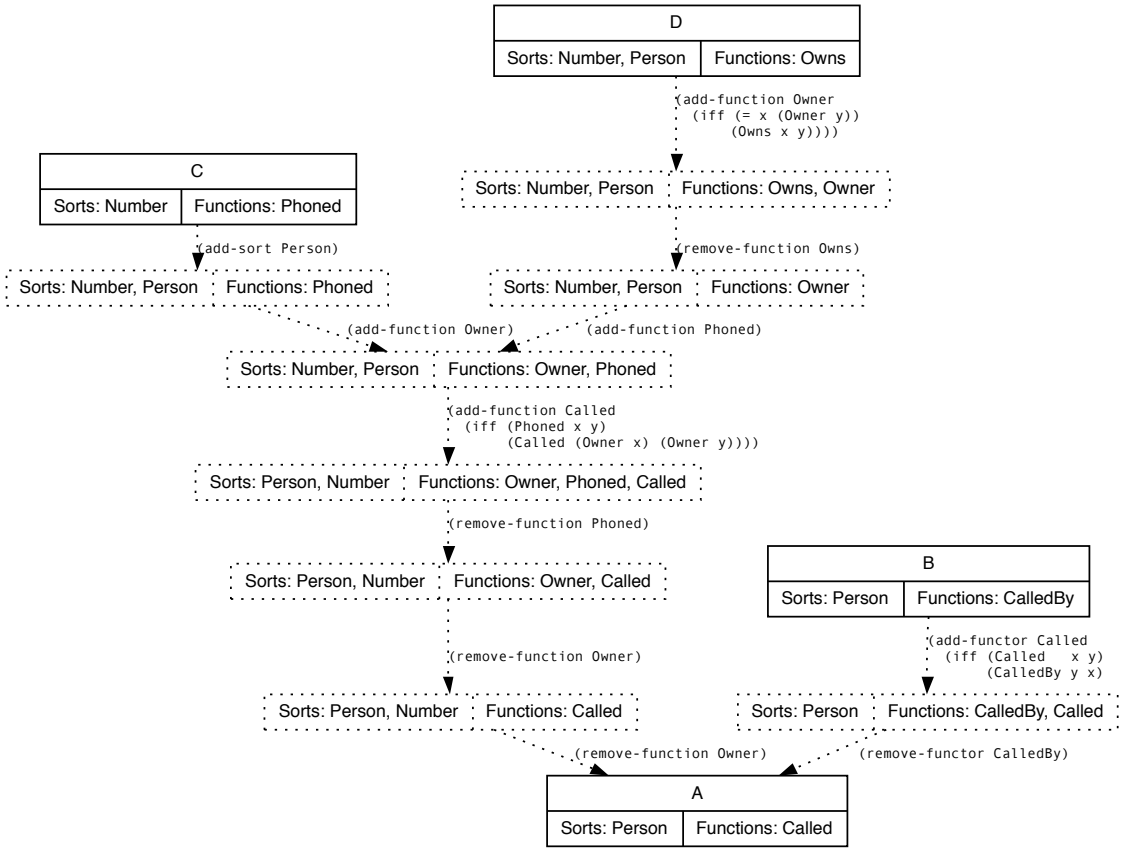


**Figure 4.4: The final translation graph of the relationships between the systems.**

# CHAPTER 5
# Future Work

Through the various experiments documented here, strides have been made in machine interoperability. Inspired by the results of the IKRIS workshop and by the ongoing exemplary work on and demonstrations of the interoperability work between GeoTime and Slate, systems in the intelligence community are investigating what they can do to become 'IKL-compliant'. With each new attempt at interoperability we take a meaningful step toward the seamless flow of information our information-driven society needs. Through continued research, we are refining the use of translation graphs for PBSI and discovering applications of semantic interoperability. There are a number of particular areas in which further research promises great benefits or would promote the adoption of PBSI.

## 5.1 Automaticity

The ultimate dream of this sort of R&D is full automaticity. Following a divide and conquer approach, translation graphs allow for the automatic production of bridging axioms. So, if translation graphs could be automatically produced, the dream would be reality. We are investigating the application of automatic programming [64] toward this goal. More immediately, some of the approaches in automated schema matching could be applied.

## 5.2 Sophisticated Ontology Representation

We built translation graphs with the signatures of many-sorted logic as nodes, for flexibility and convenience of expression, though such graphs lack some desirable features such as subsorting, sort hierarchies, and a standard language for describing the signatures themselves. There has been a great deal of research in what kind of reasoning [3] must be performed over ontologies [70], and there are many languages, such as RDF, DAML, and OWL, designed for the purpose of ontology description.

Building translation graphs from ontologies represented in these languages would allow us to work with many ontologies already constructed and in use today.

## 5.3 Categorizing Axiomatic Definitions

From certain types of axiomatic definitions we can extract rewriting rules (inline translations); indeed, to make the translation graph approach scale well, optimizations such as inline translations are almost certainly necessary. We believe more sophisticated rewriting rules and other types of procedures can be developed by examining *paths* in a translation graph, and will be pursuing this line of work.

## 5.4 Database Interoperability

Databases are one of the most common sources of information used on the web today. Massive, centralized, databases are uncommon, and so there is a great concern for enabling interoperability between databases and for the productive use of information from multiple databases.

This concern has spurred work enabling Slate to launch database queries using SQL. Figures 5.1 and 5.2 illustrate arguments in Slate being supported or denied by information in a relational database. Currently, the queries that Slate can launch are based on simple translations from first-order logical formulae into SQL queries, but we hope to enable Slate to generate queries much more intelligently. Used in this way, Slate will serve as a powerful tool for using information from multiple databases in semantically meaningful ways.

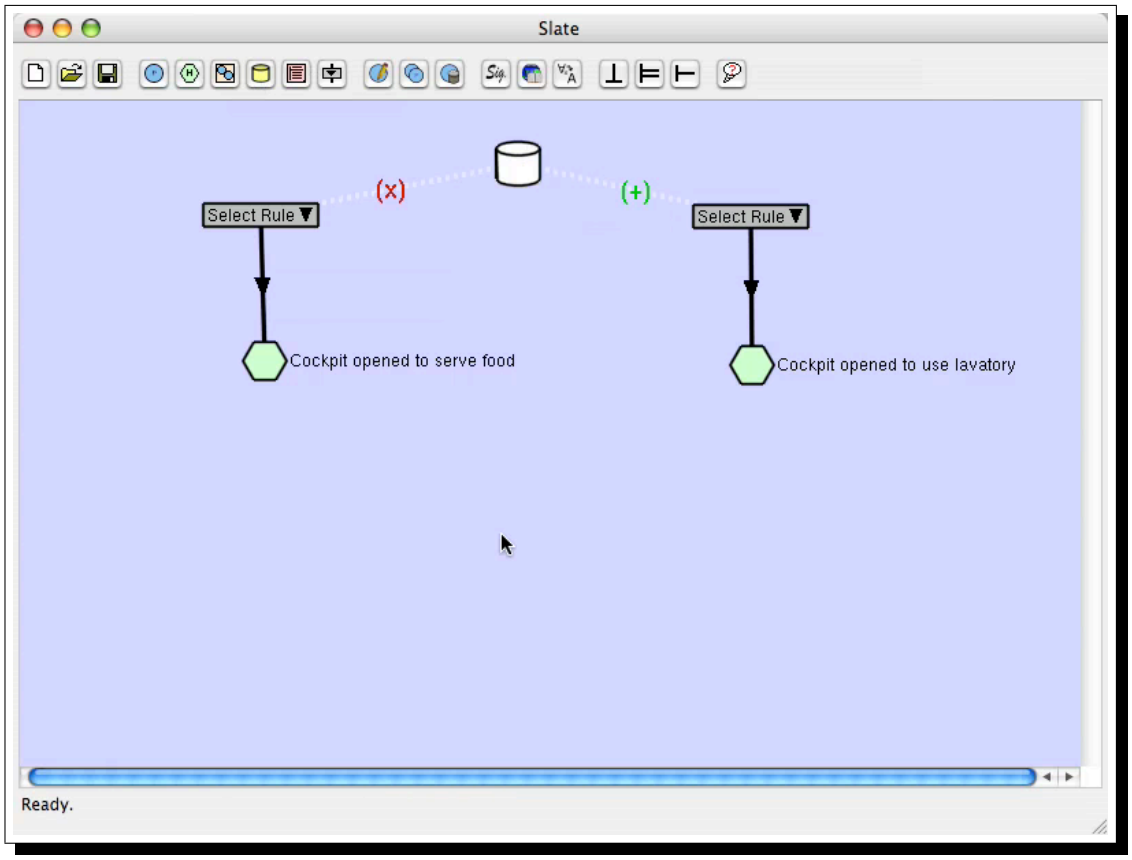


**Figure 5.1: Information from a relational database is retrieved by an SQL query automatically generated by Slate, and is used to support or deny arguments in the workspace.**

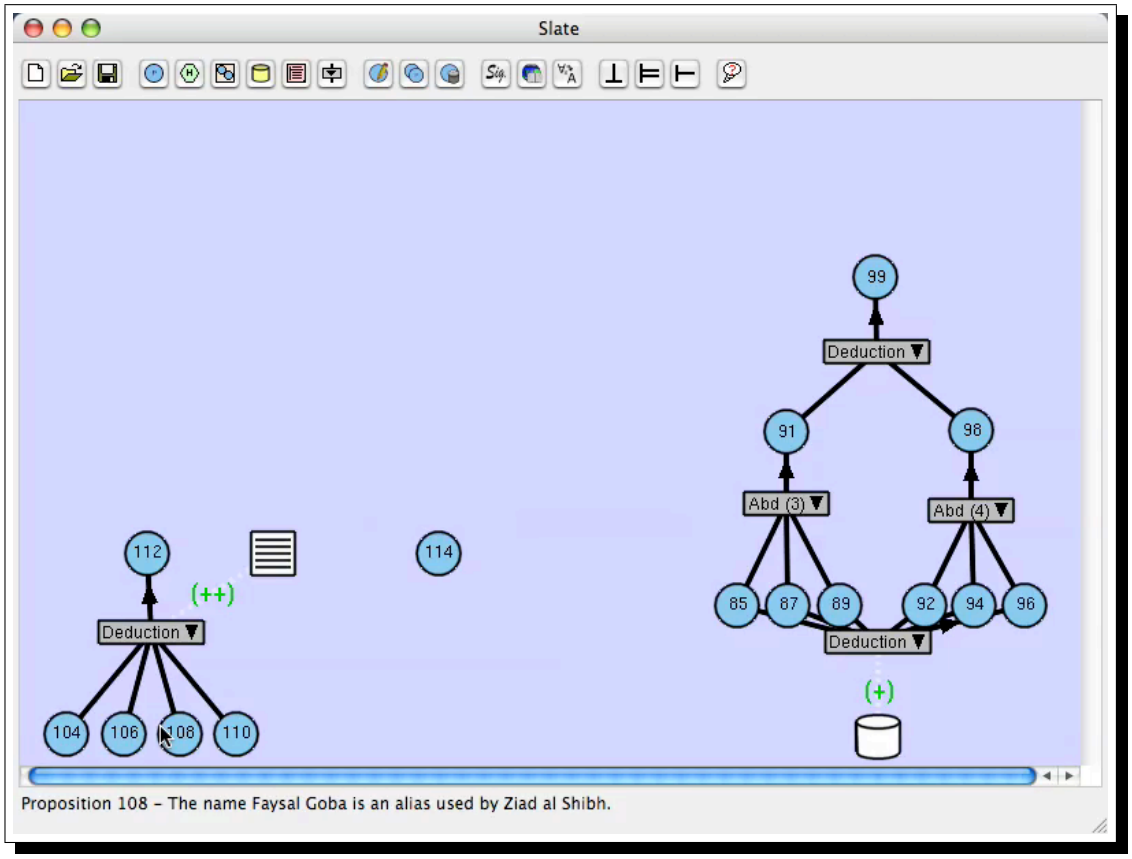


**Figure 5.2: The argument in the lower right hand corner of the Slate workspace is supported by information from a relational database.**

# APPENDIX A
# Slate ⇔ GeoTime Materials

The materials in this appendix are products of the interoperability effort between the Slate and GeoTime teams. `axioms.ikl` contains the axioms that related the OGT vocabulary to Slate's. `hypothesis.ikl` is a simple hypothesis that Slate generated. `preserved-information.ikl` records the equalities between the GUID strings used in the IKL produced form GeoTime output and the symbolic names for the same entities in Slate.

## A.1 axioms.ikl

```
(cl:comment
 'Bridging axioms to bridge between IKL representation of data from
 Oculus' GeoTime and a corresponding representation in Slate.
 Currently, much of the information in the GeoTime IKL is discarded.
 Some of this is reasonable [e.g., Slate has no corresponding mechanisms
 for viewStyles and other display information], but other information
 is more likely to be included later, such as information about time.

 The Slate representation in many-sorted-logic is as follows:

 Domains:
 * elementary events
 * compound events
 * places
 * targets

 Functions:
 * location-of : event -> place
 * participates-in : target x event -> boolean
 * compound-event-part : compound event x elementary event -> boolean

 In the intermediate representation, there are sort predicates for each
 of the domains, and objects are categorized into domains based on the
 available information in the GeoTime IKL file.

 When the types of objects have been inferred, the values of
 participates-in, location-of, and compound-event-part can be inferred
 from the remaining information in the GeoTime IKL.

 The equality or distinctness of objects in the GeoTime IKL is based
 on the distinctness (or equality) of the actual names in the text, as
 the names are UUIDs. Nonetheless, this distinctness is not expressible
 using axioms, and must be handled in the actual processing of the file.
 As such, parties must take care to preserve the identity of objects
 and UUIDs when exchanging information.

 Joshua Taylor     - tayloj@rpi.edu
 Andrew Shilliday  - shilla@rpi.edu
 Selmer Bringsjord - selmer@rpi.edu
```

```
 February 27, 2007')

(cl:comment
 'Type Definitions'
 (and
  (cl:comment
   'Elementary Events'
   (forall (e)
     (iff
      (event e)
      (and (rdf:type e gt:event)
           (cs:class e 'com.oculus.cs.geotime.model.event.EventElementary')))))
  (cl:comment
   'Compound Events'
   (forall (e)
     (iff
      (compound-event e)
      (and (rdf:type e gt:event)
           (cs:class e 'com.oculus.cs.geotime.model.event.EventCompound')))))
  (cl:comment
   'Places'
   (forall (p)
     (iff
      (place p)
      (and (rdf:type p gt:place)
           (cs:class p 'com.oculus.cs.geotime.model.place.Place')))))
  (cl:comment
   'Associations'
   (forall (a)
     (iff
      (association a)
      (and (rdf:type a cs:association)
           (cs:class a 'com.oculus.cs.model.general.association.StandardAssociation')))))
  (cl:comment
   'Targets'
   (forall (t)
     (iff
      (target t)
      (and (rdf:type t gt:target)
           (cs:class t 'com.oculus.cs.geotime.model.target.Target')))))))

(cl:comment
 'Location Axiom'
 (forall (p)
   (if (place p)
     (forall (e)
       (if (event e)
         (iff
          (= p (location-of e))
          (exists (a)
            (and (association a)
                 (or (and (cs:parent a e)
                          (cs:child  a p))
                     (and (cs:parent a p)
                          (cs:child  a e)))))))))))

(cl:comment
 'Participant Axiom'
 (forall (t)
   (if (target t)
     (forall (e)
       (if (event e)
         (iff
          (participates-in t e)
          (exists (a)
            (and (association a)
```

```
                (or (and (cs:parent a e)
                         (cs:child  a t))
                    (and (cs:parent a t)
                         (cs:child  a e)))))))))))

(cl:comment
 'Compound Event'
 (forall (e)
   (if (event e)
     (forall (c)
       (if (compound-event c)
         (iff
          (compound-event-part c e)
          (exists (a)
            (and (association a)
                 (or (and (cs:parent a e)
                          (cs:child  a c))
                     (and (cs:parent a c)
                          (cs:child  a e)))))))))))
```

## A.2 hypothesis.ikl

```
(cl:comment
 'The generated hypothesis: There will be an event in which Mark Davis, alias
Hamid Alwan, will participate, and the event will happen at the New York
Stock Exchange. This event is a bombing. This hypothesis is generated from
knowledge that Hamid Alwan has access to the NYSE, that he was trained in the
usage of explosives in Sudan and Afghanistan, and that he is wanted in
Canada.'
 (exists ("Bombing Event")
   (and (event "Bombing Event")
(participates-in "Bombing Event" "Mark Davis/Hamid Alwan")
(= "NYSE" (location-of "Bombing Event")))))
```

## A.3 preserved-information.ikl

```
(cl:comment 'UUID and object equality.'
 (and
  (= "gt:ba5bdb:1000a7acfc1:-7fa9" "Bad address")
  (= "gt:1b284a4:f83ff94049:-8000" "Select Foods, Springfield")
  (= "gt:ba5bdb:1000a7acfc1:-7fa3" "Listed Address")
  (= "gt:ba5bdb:1000a7acfc1:-7fa6" "employee")
  (= "gt:ba5bdb:1000a7acfc1:-7f58" "call from Select Foods")
  (= "gt:ba5bdb:1000a7acfc1:-7f9d" "Employee")
  (= "gt:16b7e0e:f85ab20662:-7ffd" "&quot;pick up carpet apr 25&quot;")
  (= "gt:163d208:f84a7da4ca:-7fd1" "Manager Missing")
  (= "gt:16b7e0e:f85ab20662:-7ffe" "&quot;pick up carpet apr 25&quot;1")
  (= "gt:163d208:f84a7da4ca:-7fd0" "Manager")
  (= "gt:163d208:f84a7da4ca:-7fd3" "Carpet Shop")
  (= "gt:163d208:f84a7da4ca:-7fd7" "Alwan Born")
```

```
(= "gt:ba5bdb:1000a7acfc1:-7f49" "call")
(= "gt:16b7e0e:f85ab20662:-7ff2" "call1")
(= "gt:16b7e0e:f85ab20662:-7ff1" "call2")
(= "gt:16b7e0e:f85ab20662:-7ff8" "phone")
(= "gt:16b7e0e:f85ab20662:-7ff9" "718-352-8479")
(= "gt:ba5bdb:1000a7acfc1:-7f91" "Transfer $8500")
(= "gt:163d208:f84a7da4ca:-7fe1" "Wanted in Canada")
(= "gt:163d208:f84a7da4ca:-7fe3" "SAUDI ARABIA")
(= "gt:163d208:f84a7da4ca:-7fe5" "SUDAN")
(= "gt:163d208:f84a7da4ca:-7fe7" "AFGHANISTAN")
(= "gt:1b57890:f845223c39:-7ff8" "Empire State")
(= "gt:163d208:f84a7da4ca:-7fe9" "CANADA")
(= "gt:163d208:f84a7da4ca:-7fdb" "Explosives Training")
(= "gt:1b57890:f845223c39:-8000" "cleared")
(= "gt:1b57890:f845223c39:-7ff4" "Empire State Vending")
(= "gt:163d208:f84a7da4ca:-7fdf" "Explosives training")
(= "gt:1b57890:f845223c39:-7ff2" "Employee1")
(= "gt:1b57890:f845223c39:-7ff3" "Employee2")
(= "gt:656758:f844f9e0bd:-8000" "NYSE")
(= "gt:656758:f844f9e0bd:-7ff9" "Access to NYSE")
(= "gt:656758:f844f9e0bd:-7ff3" "City Computers")
(= "gt:656758:f844f9e0bd:-7ff1" "employee1")
(= "gt:656758:f844f9e0bd:-7ff0" "employee2")
(= "gt:656758:f844f9e0bd:-7ff7" "Bad address1")
(= "gt:656758:f844f9e0bd:-7ff6" "Bad address2")
(= "gt:1fa8d3b:f84bc97d7c:-7ffb" "Fought in Afghan")
(= "gt:ba5bdb:1000a7acfc1:-7f82" "Manager1")
(= "gt:16b7e0e:f85ab20662:-7fdd" "call3")
(= "gt:16b7e0e:f85ab20662:-7fdc" "call4")
(= "gt:656758:f844f9e0bd:-7ffc" "Mark Davis/Hamid Alwan")
(= "gt:656758:f844f9e0bd:-7ffb" "1631 Webster Ave")
(= "gt:1b57890:f845223c39:-7ffd" "Address")
(= "gt:1b57890:f845223c39:-7ffe" "City Computer Corp")
(= "gt:1b57890:f845223c39:-7ffb" "Bagwant Dhaliwal/Sahim Albakri")
(= "gt:1b57890:f845223c39:-7ffa" "Address1")
(= "gt:eb9f58:f84554401e:-8000" "Hani al Hallak")
(= "gt:eb9f58:f84554401e:-7ffc" "Transfer $85001")
(= "gt:eb9f58:f84554401e:-7fff" "N. Bergen")
(= "gt:eb9f58:f84554401e:-7ffd" "Transfer $85002")
(= "gt:ba5bdb:1000a7acfc1:-7f5e" "&quot;pick up carpet apr 25&quot;2")
(= "gt:656758:f844f9e0bd:-7fed" "2462 Myrtle Ave, Queens")
(= "gt:656758:f844f9e0bd:-7fd2" "Address2")
(= "gt:656758:f844f9e0bd:-7fd1" "Address3")
(= "gt:163d208:f84a7da4ca:-7fcd" "C-4 found")))
```

# APPENDIX B
# Code

## B.1 signatures.lisp

`signatures.lisp` contains a Common Lisp implementation of the signature and translation graph structures and algorithms described in Chapter 4.

```
;;; You can use ASDF to load FSet, and FSet is ASDF-installable. If
;;; you don't use ASDF, comment out the follwing line and be sure to
;;; load FSet somehow before using this code.

#-:fset-ext-strings (asdf:operate 'asdf:load-op :fset)

(defpackage #:translation-graphs
  (:documentation
   "The translation graphs package provides an implementation of the
translation graphs described in \"Provability-Based Semantic Interoperability
via Translation Graphs\" Joshua Taylor, Andrew Shilliday, and Selmer
Bringsjord.  ONISW 2007.  It depends on FSet library of functional
set-theoretic structures (which is available through asdf-install, and from
http://common-lisp.net/project/fset/).

There are two interesting things that this code will allow one to do.  The
first is the generation of translation graph images.  When translation graphs
are built incrementally (such as in the examples), the resulting translation
graph can be printed in GraphViz's dot language.  GraphViz is freely
available (http://www.graphviz.org/) and images in a variety of formats can be
generated from dot files. These diagrams can be helpful for documenting the
relationships between ontologies.

The second interesting thing is the automatic extraction of bridging axioms
from a translation graph. The idea here is that given access to information in
a number ontologies, it might be possible to translation information from one
ontology into another.  This translation can be implemented procedurally, but
goverened by axioms extracted from a translation graph. Sometimes, translation
from one ontology to another is impossible, but the bridging axioms extracted
from a translation graph can still make some of the information from one
ontology useful to another ontology.")
  (:nicknames #:tg)
  (:shadowing-import-from #:fset
   #:set #:map #:with #:map-default :empty-map #:empty-set #:do-set
   #:do-map #:lookup #:range #:domain :convert #:less #:compare #:@)
  (:use :cl))

(in-package #:translation-graphs)
```

```
;;; Utilities

(defmacro with-gensyms ((&rest syms) &body body)
  "Bind each of syms to a gensym'd symbol, and evaluation body in this
context. with-gensyms is designed for building macros."
  `(let (,@(mapcar #'(lambda (sym)
       (list sym `',(gensym)))
   syms))
     ,@body))

(defmacro mvlet* (&whole whole bindings &body body)
  "Mvlet* combines the functionality of multiple-value-bind and let*.  Each
binding should be either a symbol, or a list with two elements.  If the
binding is a symbol, then it is bound to NIL by let (e.g., (let (x) ...)). If
it is a list, then if the car of the binding is a symbol, it is bound using
let to the second value of the list (e.g., (let ((x y)) ...)). If the car of
the binding is a list, then it is a list of bindings used with
multiple-value-bind (e.g., (multiple-value-bind (x y z) <values-form>
<body>))."
  (cond
    ((null bindings)
     `(progn ,@body))
    ((not (listp bindings))
     (error "Expected a list of bindings in ~A, but got ~A" whole bindings))
    (t (destructuring-bind (binding &rest bindings) bindings
 (if (symbolp binding)
     `(let (,binding)
(mvlet* ,bindings
  ,@body))
     (destructuring-bind (e v) binding
       (if (listp e)
   `(multiple-value-bind ,e ,v
      (mvlet* ,bindings
,@body))
   `(let ((,e ,v))
      (mvlet* ,bindings
,@body)))))))))

(defun in-map? (map key)
  "Is key a key in the map? Convenient mechanism for checking."
  (multiple-value-bind (value found?) (lookup map key)
    (values found? value)))

(defun mapped-to? (map value)
  "Does some key map to the value? Convenient mechanism for checking."
  (lookup (range map) value))

;;; Signatures

(defstruct signature
  "A signature is a tuple <s,f> where s is a mapping of sort names to sorts,
```

```
and f is a mapping of functor names to functors."
  (sorts    (empty-map))
  (functors (empty-map)))

(defmethod compare ((s1 signature) (s2 signature))
  "Signatures are composed of two maps.  Their specialized compare compares
one of the maps. If that comparison is not :equal, it is returned. Otherwise,
the other map is compared, and that comparison is returned."
  (let ((sorts-compare (compare (signature-sorts s1)
(signature-sorts s2))))
    (if (not (eq :equal sorts-compare))
sorts-compare
(compare (signature-functors s1)
 (signature-functors s2)))))

(defstruct functor
  "Functors are indentified by object equality. Argument-types is a list of
sorts objects, and value-type is a sort object."
  (argument-types ())
  (value-type     NIL))

(defun add-sort (signature sort-name &optional (sort (gensym) sort-p))
  "Return a new signature like signature with a mapping from sort-name to
sort. It is an error is sort-name already names a sort in the signature, or if
the sort is already named in the signature (this can only happen if sort is
provided)."
  (if (in-map? (signature-sorts signature) sort-name)
      (error "~A already names a sort in ~A." sort-name signature)
      (multiple-value-bind (mapped? name)
  (mapped-to? (signature-sorts signature) sort)
(if mapped?
    (error "~A is already named ~A in ~A." sort name signature)
    (make-signature
     :sorts (with (signature-sorts signature) sort-name sort)
     :functors (signature-functors signature))))))

(defun remove-sort (signature sort-name)
  "Return a new signature like signature where sort-name maps does not map to
any sort."
  (if (not (in-map? (signature-sorts signature) sort-name))
      signature
      (make-signature
       :sorts (less (signature-sorts signature) sort-name)
       :functors (signature-functors signature))))

(defun functor-sorts-ok? (functor signature)
  "Do all the sorts used in the functor appear in the signature?"
  (let ((sorts (range (signature-sorts signature))))
    (flet ((sort-mapped? (sort)
     (lookup sorts sort)))
      (and (sort-mapped? (functor-value-type functor))
   (every #'sort-mapped? (functor-argument-types functor))))))
```

```
(defun add-functor (signature functor-name functor)
  "Return a new signature like signature with a mapping from functor-name to
functor. It is an error if functor-name already names a functor in signature,
or if functor is already named in signature."
  (cond
    ((in-map? (signature-functors signature) functor-name)
     (error "~A already names a functor in ~A." functor-name signature))
    ((not (functor-sorts-ok? functor signature))
     (error "~A uses sorts not in ~A." functor signature))
    (t (multiple-value-bind (mapped? name)
   (mapped-to? (signature-functors signature) functor)
 (if mapped?
     (error "~A is already named ~A in ~A." functor name signature)
     (make-signature
      :sorts (signature-sorts signature)
      :functors (with (signature-functors signature)
      functor-name functor)))))))

(defun add-new-functor (signature functor-name value-type &rest argument-types)
  "Return a new signature like signature with a mapping from functor-name to a
new functor.  value-type and argument-type should be names of sorts in
signature."
  (flet ((get-sort (name)
   (multiple-value-bind (sort found?)
       (lookup (signature-sorts signature) name)
     (if found?
 sort
 (error "~A does not denote a sort in ~A." name signature)))))
    (add-functor
     signature
     functor-name
     (make-functor
      :argument-types (mapcar #'get-sort argument-types)
      :value-type (get-sort value-type)))))

(defun remove-functor (signature functor-name)
  "Return a new signature like signature in which functor-name does not map to
any functor."
  (if (not (in-map? (signature-functors signature) functor-name))
      signature
      (make-signature
       :sorts (signature-sorts signature)
       :functors (less (signature-functors signature) functor-name))))

;;; Translation Graphs

(defstruct edge
  "An edge is directed, and denotes a link from `from' to `to'.  Reason
provides a mechanism for annotating /why/ the link exists.  Reason is the slot
in which axiomatic relationships are recorded.  There is no programmatic
restriction on what sort of values can be used as reasons. The intended use is
```

```
that they represent bridging axioms, and these might be expressed differently
depending on ontology structure and the language being used. The printability
of reasons may have noticeable effects on GraphViz dot output."
  (from (make-signature) :type signature)
  (to   (make-signature) :type signature)
  reason)

(defstruct translation-graph
  "A translation graph is a graph in the standard sense.  Nodes is a set of
signatures, and edges is a set of edges on the nodes of the graph."
  (nodes (set))
  (edges (set)))

(defmacro tg-defun (name (graph signature &rest args) &body body)
  "Define a function named name which accepts at least two arguments, a graph
and a signature.  The final two forms of body should generate, respectively, a
value to use as a reason in the translation graph, and a new
signature. (Earlier body forms will be placed in the function definition as
with defun, and may be declarations or documentation.)  The function, named
name, will evaluate the signature form to produce a signature.  Two values
will be returned: the first is the new signature; the second is a translation
graph similar to graph, but with an edge from signature to the new signature."
  (with-gensyms (sig2)
    (destructuring-bind (reason-form sig-form) (last body 2)
      `(defun ,name (,graph ,signature ,@args)
 ,@(butlast body 2)
 (if (not (lookup (translation-graph-nodes ,graph) ,signature))
     (error "The signature ~A is not in the graph ~A."
    ,signature ,graph)
     (let ((,sig2 ,sig-form))
       (values ,sig2
       (make-translation-graph
:edges (with (translation-graph-edges ,graph)
     (make-edge
      :from ,signature
      :to ,sig2
      :reason ,reason-form))
:nodes (with (translation-graph-nodes ,graph)
     ,sig2)))))))))

(defun tg-start ()
  "Return as multiple values an empty signature and translation graph with a
single node (the signature)."
  (let* ((signature (make-signature))
 (graph (make-translation-graph :nodes (set signature))))
    (values signature graph)))

(tg-defun tg-add-sort (graph signature sort-name &optional (sort (gensym)))
  "As multiple values, return a new signature like signature in which
sort-name maps to sort, and a graph similar to graph in which there is an edge
from signature to the new signature."
  (list :add-sort sort-name sort)
```

```
  (add-sort signature sort-name sort))

(tg-defun tg-remove-sort (graph signature sort-name)
  "As multiple values, return a new signature like signature in which
sort-name does not map to any sort, and a graph similar to graph in which
there is an edge from signature to the new signature."
  (list :remove-sort sort-name)
  (remove-sort signature sort-name))

(tg-defun tg-add-functor (graph signature functor-name functor)
  "As multiple values, return a new signature like signature in which
functor-name maps to functor, and a graph similar to graph in which there is
an edge from signature to the new signature."
  (list :add-functor functor-name functor)
  (add-functor signature functor-name functor))

(tg-defun tg-add-new-functor
  (graph signature functor-name value-type &rest argument-types)
  "As multiple values, return a new signature like signature in which
functor-name maps to a new functor described by value-type and argument-types,
and a graph similar to graph in which there is an edge from signature to the
new signature."
  (list :add-new-functor functor-name)
  (apply #'add-new-functor
 signature functor-name value-type argument-types))

(tg-defun tg-add-functor-axiomatically
    (graph signature axiom functor-name functor)
  "As multiple values, return a new signature like signature in which
functor-name maps to functor, and a graph similar to graph in which there is
an edge from signature to the new signature which records the axiom."
  axiom
  (add-functor signature functor-name functor))

(tg-defun tg-add-new-functor-axiomatically
  (graph signature axiom functor-name value-type &rest argument-types)
  "As multiple values, return a new signature like signature in which
functor-name maps to a new functor described by value-type and argument-types,
and a graph similar to graph in which there is an edge from signature to the
new signature which records the axiom."
  axiom
  (apply #'add-new-functor
 signature functor-name value-type argument-types))

(tg-defun tg-remove-functor (graph signature functor-name)
  "As multiple values, return a new signature like signature in which
functor-name does not map to any functor, and a graph similar to graph in
which there is an edge from signature to the new signature."
  (list :remove-functor functor-name)
  (remove-functor signature functor-name))

;;; Printing DOT
```

```
(defun print-signature-description-dot
    (signature &optional (out *standard-output*))
  "Print a label suitable for a dot node with shape=\"record\"."
  (format out "{Sorts: ~{~A~^, ~}| Functions: ~{~A~^, ~}}"
  (convert 'list (domain (signature-sorts signature)))
  (convert 'list (domain (signature-functors signature)))))

(defun print-translation-graph-dot
    (graph &optional (out *standard-output*))
  "Print the dot representation of the translation graph to the specified
stream."
  (flet ((p (&rest args) (apply #'format out args)))
    (let ((ht (make-hash-table))
  (nodes (translation-graph-nodes graph))
  (edges (translation-graph-edges graph)))
      ;; the  extra lookup  here is  due to  the fact  that  edges may
      ;; record signatures which are no longer in the set (having been
      ;; replaced by signatures  which (fset:compare) :equal). calling
      ;; lookup  here ensures  that  we only  reference the  signature
      ;; which are actually currently in the set.
      (macrolet ((sig-name (sig)
   `(gethash (nth-value 1 (lookup nodes ,sig)) ht)))
(p "digraph {~%")
(p "    graph [];~%")
(p "    node [fontname=\"Arial\",shape=\"record\"];~%")
(p "    edge [fontsize=\"10\",fontname=\"Monaco\"];~2%")
(do-set (sig nodes)
  (let ((name (string (gensym))))
    (setf (sig-name sig) name)
    (p "    node [label=\"")
    (print-signature-description-dot sig out)
    (p "\"]; ~A;~%" name)))
(do-set (edge edges)
  (p "    ~A -> ~A [label=\"~A\"];~%"
     (sig-name (edge-from edge))
     (sig-name (edge-to   edge))
     (edge-reason edge)))
(p "}~%")))))
```

## B.2 examples

This section contains example code illustrating how signatures can be constructed and related, and how translation graphs can be extracted from this process.

# LITERATURE CITED


[1] OpenCyc, http://www.opencyc.org/, July 2007.

[2] Standard upper ontology working group (SUO WG), http://suo.ieee.org/, 2007.

[3] Grigoris Antoniou, Enrico Franconi, and Frank van Harmelen. Introduction to semantic web ontology languages. In Norbert Eisinger and Jan Małuszyński, editors, *Reasoning Web, Proceedings of the Summer School, Malta, 2005*, number 3564 in Lecture Notes in Computer Science, Berlin, Heidelberg, New York, Tokyo, 2005. Springer-Verlag.

[4] Aristotle and E. M. Edghill. Categories. translation online, 350BC.

[5] Konstantine Arkoudas. Formalizing natural deduction with assumption bases. Submitted for publication.

[6] Konstantine Arkoudas. Athena, http://www.cag.csail.mit.edu/~kostas/dpls/athena, 2005. http://www.cag.csail.mit.edu/~kostas/dpls/athena.

[7] David Barker-Plummer and Mark Greaves. Architectures for Heterogeneous Reasoning on Interlinguae, October 1998.

[8] S. Bringsjord and D. Ferrucci. Logic and artificial intelligence: Divorced, still married, separated...? *Minds and Machines*, 8:273–308, 1998.

[9] S. Bringsjord and Y. Yang. Representations using formal logics. In L. Nadel, editor, *Encyclopedia of Cognitive Science Vol 3*, pages 940–950. Nature Publishing Group, London, UK, 2003.

[10] Selmer Bringsjord. Declarative/logic-based computational cognitive modeling. In Ron Sun, editor, *The Handbook of Computational Cognitive Modeling*. Cambridge University Press, Cambridge, UK, forthcoming.

[11] Selmer Bringsjord, Konstantine Arkoudas, Micah Clark, Andrew Shilliday, Joshua Taylor, Bettina Schimanski, and Yingrui Yang. Reporting on some logic-based machine reading research. In *Proceedings of the 2007 AAAI Spring Symposium on Machine Reading*, January 2007.

[12] Selmer Bringsjord, Kostas Arkoudas, Deepa Mukherjee, Andrew Shilliday, Joshua Taylor, Micah Clark, and Elizabeth Bringsjord. The Multi-Mind Effect. In *Proceedings of the 2007 International Conference on Artificial Intelligence*, volume 1, pages 43–49. CSREA Press, June 25-28 2007.

[13] Selmer Bringsjord, Micah Clark, Andrew Shilliday, and Joshua Taylor. Solomon, http://www.cogsci.rpi.edu/solomon/.

[14] Selmer Bringsjord, Andrew Shilliday, and Joshua Taylor. Slate, http://www.cogsci.rpi.edu/slate/.

[15] Selmer Bringsjord, Andrew Shilliday, and Joshua Taylor. An Introduction to Slate. Introduction to Logic Lecture, August 2005.

[16] Saša Buvač and Richard Fikes. A declarative formalization of knowledge translation. In *CIKM '95: Proceedings of the fourth international conference on Information and knowledge management*, pages 340–347, New York, NY, USA, 1995. ACM Press.

[17] Alan Chappell, Selmer Bringsjord, Andrew Shilliday, Joshua Taylor, and William Wright. Integration Experiment with GeoTime, Slate, and VIKRS. ARIVA Principal Investigator Meeting Handout, March 2007.

[18] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, 2006.

[19] Common Logic Working Group. Common Logic Standard, http://cl.tamu.edu/, 2007.

[20] Common Logic Working Group. Information technology – Common Logic (CL): a framework for a family of logic-based languages (FINAL DRAFT). ISO/IEC FDIS 24707:2007(E), 2007.

[21] Andrew J. Cowell, Deborah L. McGuinness, Carrie F. Varley, and David A. Thurman. Knowledge-Worker Requirements for Next Generation Query Answering & Explanation Systems. In *Proceedings of the Workshop on Intelligent User Interfaces for Intelligence Analysis, International Conference on Intelligent User Interfaces (IUI 2006)*, Sydney, Australia, 2006.

[22] http://www.daml.org/ DARPA Agent Markup Program. DARPA Agent Markup Language Homepage, March 2003.

[23] Chris Deaton, Blake Shepard, Charles Klein, Corrinne Mayans, Brett Summers, Antoine Brusseau, Michael Witbrock, and Doug Lenat. The comprehensive terrorist knowledge base in cyc. In *Conference Proceedings of the 2005 International Conference on Intelligence Analysis* [55].

[24] Dejing Dou and Drew McDermott. Deriving axioms across ontologies. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 952–954, New York, NY, USA, 2006. ACM Press.

[25] Dejing Dou, Drew McDermott, and Peishen Qi. *Ontology Translation by Ontology Merging and Automated Reasoning*, pages 73–94. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser Basel, 2005.

[26] David Ferrucci. Text Analysis as Formal Inference for the Purposes of Uniform Tracing and Explanation Generation. Research Report RC23372, IBM, IBM Reseach Division, Thomas J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598, October 2004.

[27] David Ferrucci and Adam Lally. Accelerating corporate research in the development, application and deployment of human language technologies. In *SEALTS '03: Proceedings of the HLT-NAACL 2003 workshop on Software engineering and architecture of language technology systems*, pages 67–74, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[28] Richard E. Fikes, David Ferrucci, and David A. Thurman. Knowledge Associates for Novel Intelligence. In *Proceedings of the 2005 International Conference on Intelligence Analysis (IA 2005)*, McLean, VA, USA, May 2005.

[29] Richard E. Fikes, Alan Marwick, and David Thurman. Knowledge Associates for Novel Intelligence (KANI). Technical report, Knowledge Systems Laboratory, October 2003.

[30] Norbert E. Fuchs and Kaarel Kaljurand. Attempto Controlled English: Language, tools and applications. Lecture / Presentation, December 2006.

[31] Michael R. Genesereth and Richard E. Fikes. *Knowledge Interchange Format Version 3 Reference Manual*, December 1997.

[32] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[33] Joseph A. Goguen. Information integration in institutions. Paper for Jon Barwise memorial volume edited by Larry Moss, 2004.

[34] Joseph A. Goguen. Data, Schema, Ontology and Logic Integration. *Logic Journal IGPL*, 13(6):685–715, 2005.

[35] Joseph A. Goguen and Rod M. Burstall. Introducing institutions. In Edmund M. Clarke and Dexter Kozen, editors, *Logics of Programs*, volume 164 of *Lecture Notes in Computer Science*, pages 221–256. Springer, June 1984.

[36] Tom Gruber. ARPA Knowledge Sharing Effort public library, http://www-ksl.stanford.edu/knowledge-sharing/.

[37] Michael Gruninger. Common Logic standardization meeting, minutes. http://cl.tamu.edu/minutes/stanford-minutes.html, March 21-22 2002.

[38] R. V. Guha and Douglas B. Lenat. Cyc: a midterm report. *AI Magazine*, 11(3):32–59, 1990.

[39] Pat Hayes and Christopher Menzel. IKL Specification Document. IKL specification document, 2006.

[40] James Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, pages 30–37, March/April 2001.

[41] Jerry Hobbs. IKRIS scenarios fundamentals. Workshop document.

[42] Jerry R. Hobbs. *An OWL Ontology of Time*, July 2004.

[43] Jerry R. Hobbs and Feng Pan. An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(1):66–85, 2004.

[44] Oculus Info Inc. GeoTime enables visualization for concurrent spatial and temporal analysis. Press release, May 11 2005.

[45] Thomas Kapler, Robert Harper, and William Wright. Correlating Events with Tracked Movements in Time and Space: A GeoTime Case Study. In *Proceedings of the 2005 International Conference on Intelligence Analysis (IA 2005)*, McLean, VA, USA, May 2005.

[46] Thomas Kapler and William Wright. Geotime information visualization. In *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, pages 25–32, Washington, DC, USA, 2004. IEEE Computer Society.

[47] Thomas Kapler and William Wright. Geotime information visualization. *Information Visualization*, 4(2):136–146, 2005.

[48] Douglas B. Lenat and R. V. Guha. The evolution of cycl, the cyc representation language. *SIGART Bull.*, 2(3):84–87, 1991.

[49] Douglas B. Lenat, R. V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc: toward programs with common sense. *Communications of the ACM*, 33(8):30–49, 1990.

[50] Maria Manzano. *Extensions of First Order Logic*. Cambridge University Press, 1996.

[51] Deborah L. McGuinness. Explaining Question Answering Systems with Contexts. In Pavel Shvaiko, Jerome Euzenat, Alain Leger, Deborah L. McGuinness, and Holger Wache, editors, *Contexts and Ontologies: Theory, Practice and Applications: Papers from the 2005 AAAI Workshop*, pages 136–137, Menlo Park, CA, 2005. American Association for Artificial Intelligence.

[52] Deborah L. McGuinness, Li Ding, Alyssa Glass, Cynthia Chang, Honglei Zeng, and Vasco Furtado. Explanation Interfaces for the Semantic Web: Issues and Models. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI'06)*, November 2006.

[53] Christopher Menzel. Common Logic Standard. In *Santa Fe 2003 Metatdata Forum Symposium on Ontologies*, 2003. Presentation of the case for Common Logic.

[54] MITRE. IKRIS, http://nrrc.mitre.org/NRRC/ikris.htm, 2007.

[55] MITRE, Sponsored by the Office of the Assistant Director of Central Intelligence for Analysis and Production. *Conference Proceedings of the 2005 International Conference on Intelligence Analysis*, McLean, VA, USA, May 2005.

[56] Robert Neches. The knowledge sharing effort, July 26 1994.

[57] I. Niles and A. Pease. Origins of the ieee standard upper ontology, 2001.

[58] I. Niles and A. Pease. Linking lexicons and ontologies: Mapping WordNet to the suggested upper merged ontology. In *Proceedings of the 2003 International Conference on Information and Knowledge Engineering (IKE '03), Las Vegas, Nevada*, 2003.

[59] Ian Niles and Adam Pease. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, *Proceedings of the 2nd International Conference on Formal Ontology (FOIS-2001)*, 2001.

[60] A. Pease and W. Murray. An english to logic translator for ontology-based knowledge representation languages. In *Proceedings of the 2003 International Conference on Natural Language Processing and Knowledge Engineering*, pages 777–783, 2003.

[61] A. Pease, I. Niles, and J. Li. The suggested upper merged ontology: A large ontology for the semantic web and its applications, 2002.

[62] Adam Pease and John Li. Controlled English to Logic Translation. DRAFT, DO NOT DISTRIBUTE, May 2007.

[63] Pascale Proulx, Sumeet Tandon, Adam Bodnar, David Schroh, Robert Harper, and William Wright. Avian flu case study with nspace and geotime. In *IEEE Symposium On Visual Analytics Science And Technology 2006 (VAST 2006)*, pages 27–34, 2006.

[64] Charles Rich and Richard C. Waters. Automatic programming: Myths and prospects. *Computer*, 21(8):40–51, 1988.

[65] Craig Schenloff, Michael Gruninger, Florence Tissot, John Valois, Josh Lubell, and Jintae Lee. The process specification language (PSL) overview and version 1.0 specification, NISTIR 6459. Technical report, NIST, 2000.

[66] Craig Schenloff, Amy Knutilla, and Steven Ray. Proceedings of the process specification language (PSL) roundtable, NISTIR 6081. Technical report, NIST, 1997.

[67] Stewart Shapiro. Classical logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. http://plato.stanford.edu/archives/win2006/entries/logic-classical/, Winter 2006.

[68] Stuart C. Shapiro. A net structure for semantic information storage, deduction and retrieval. In *IJCAI*, pages 512–523, 1971.

[69] Nick Siegel, Blake Shepard, John Cabral, and Michael Witbrock. Hypothesis Generation and Evidence Assembly for Intelligence Analysis: Cycorp's Noöscape Application. In *Proceedings of the 2005 International Conference on Intelligence Analysis (IA 2005)*, McLean, VA, USA, May 2005.

[70] Barry Smith. The basic tools of formal ontology. In N. Guarino, editor, *Formal Ontology in Information Systems*, pages 19–28. IOS Press, 1998.

[71] John F Sowa. *Conceptual structures : information processing in mind and machine*. The Systems programming series. Addison-Wesley, Reading, Mass., 1984.

[72] John F. Sowa. Semantic networks, http://www.jfsowa.com/pubs/semnet.htm, June 2007.

[73] Tomek Strzalkowski, Sharon Small, Hilda Hardy, Boris Yamrom, Ting Liu, Paul Kantor, K. B. Ng, and Nina Wacholder. HITIQA: A Question Answering Analytical Tool. In *Conference Proceedings of the 2005 International Conference on Intelligence Analysis* [55].

[74] Joshua Taylor, Andrew Shilliday, and Selmer Bringsjord. Provability-based semantic interoperability via translation graphs. In *International Workshop on Ontologies and Information Systems for the Semantic Web (ONISW 2007)*, November 2007.

[75] Guilan Wang, Joseph A. Goguen, Young-Kwang Nam, and Kai Lin. Critical points for interactive schema matching. In J. X. Yu, X. Lin, H. Lu, and Y. Zhang, editors, *Advanced Technologies and Applications: Proceedings of Sitch Asia Pacific Web Conference*, pages 654–664, Hangzhou, China, 2004. Springer.

[76] Chris Welty. KIF working draft, http://cl.tamu.edu/discuss/goals.html, February 2002.

[77] Christopher Welty, J. William Murdock, Paulo Phineiro da Silva, Deborah L. McGuinness, David Ferrucci, and Richard E. Fikes. Tracking Information Extraction from Intelligence Documents. In *Proceedings of the 2005 International Conference on Intelligence Analysis (IA 2005)*, pages 2–6, McLean, VA, USA, May 2005.