Automatic Generation of Data Processing Workflows for Transportation Modeling

José Luis Ambite
University of Southern California
Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
ambite@isi.edu

Dipsy Kapoor
University of Southern California
Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
dipsy@isi.edu

ABSTRACT

Scientists, economists, and planners in government, industry and academia spend much of their time accessing, integrating, and analyzing data. However, many of their studies are one-of-a-kind with little sharing and reuse for subsequent endeavors. The Argos project seeks to improve the productivity of analysts by providing a framework that encourages reuse of data sources and data processing operations, and by developing tools to generate data processing workflows.

In this paper, we present an approach to *automatically* generate data processing workflows. First, we define a methodology for assigning formal semantics to data and operations according to a domain ontology, which allows sharing and reuse. Specifically, we define data contents using *relational* descriptions in an expressive logic. Second, we develop a novel planner that uses relational subsumption to connect the output of a data processing operation with the input of another. Our modeling methodology has the significant advantage that the planner can *automatically* insert adaptor operations wherever necessary to bridge the inputs and outputs of operations in the workflow. We have implemented the approach in a transportation modeling domain.

Categories and Subject Descriptors

H.2.5 [**Information Systems**]: Database Management—*Heterogeneous Databases*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; D.1.6 [**Software**]: Programming Techniques—*Logic Programming*

General Terms

Web Service Composition, Workflow, Information Integration, Knowledge Representation, Logic

1. INTRODUCTION

Much of the work of scientists and engineers in government, industry and academia is consumed by accessing, integrating, and analyzing data. This is particularly true in the planning and economic modeling government agencies. Unfortunately, there is a severe lack of tools to facilitate this process. Even when data is present in databases or other structured digital support, much of the integration is done manually by ad-hoc methods. Moreover, raw data is of limited utility. Usually these data are the input to models of more complex phenomena that produce additional data of interest. For example, in our transportation modeling domain, we derive truck traffic along specific highway links within a metropolitan area, based on far-removed source data such as employment, imports and exports.

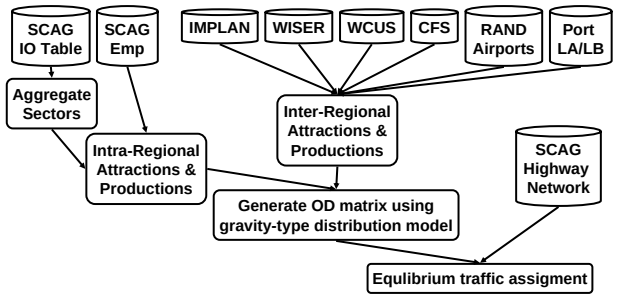
Our goal is to improve this state of affairs by providing a framework to (1) describe data and processing operations so that they can be shared and reused and (2) generate new data on demand by automatically composing data processing workflows using available sources and operations. In this paper, we present our solutions to these two challenges.

We have applied the Argos approach to a transportation modeling domain [4, 3, 13], which we will use as an example to ground the discussion. However, our approach is general and can be applied to produce data processing workflows in any other domain, as long as the data and operations are described in an suitable ontology of the domain.

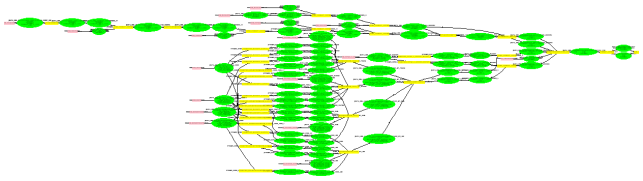
As an example of a data processing workflow consider Figure 1(a) that shows an abstract view of a model [14, 13] for estimating the flow of commodities in a metropolitan area based on secondary sources. This model was applied to the Los Angeles Consolidated Statistical Metropolitan Area (LACMSA), a region that includes the twin ports of Los Angeles and Long Beach (which together top the U.S. in terms of container shipments), as well as the Los Angeles International airport among others.

The model estimates the *intra-regional* trade based on employment data and an input-output transaction model of the local economy (produced by the Southern California Association of Governments — SCAG), resulting in a table of attractions and productions of different commodity sectors for each traffic analysis zone (TAZ) within the region.[1] To estimate the *inter-regional* trade, the model uses a variety of data sources, including seaborne commerce data from the Waterborne Commerce of the United States (WCUS),[2] airport economic statistics compiled by RAND Corporation,[3] and data from the Commodity Flow Survey (CFS) of the US Census Bureau,[4] WISER trade,[5] and IMPLAN.[6] The inter-regional trade attractions and productions per commodity are assigned to the TAZs of the entry/exit points in the region. For example, airborne imports of computer equipment are assigned to the TAZs corresponding to the airports in

[1] A Traffic Analysis Zone is a spatial region consisting of several census blocks. The LACMSA is partitioned into 3165 TAZs.
[2] http://www.iwr.usace.army.mil/ndc/data/datawcus.htm
[3] http://ca.rand.org/stats/economics/airport.html
[4] http://www.census.gov/econ/www/se0700.html
[5] http://www.wisertrade.org/
[6] http://www.implan.com/



(a) Abstract Data Processing Workflow



(b) Structure of the Complete Data Processing Workflow

Figure 1: Estimating Commodity Flows in the LACMSA region

Figure 2: Estimated Truck Volume in the Los Angeles Highway Network

the region. The intra- and inter-regional attractions and productions are converted to an Origin-Destination matrix between pairs of TAZs using a gravity model. Finally, a network equilibrium algorithm[7] assigns the origin-destination data to specific links in the highway network. Figure 2 shows graphically the final result of the workflow: an assignment of the flow of freight to each of the links of the highway network of the region of interest, in our case LACMSA. Thicker lines indicate higher truck traffic volumes. Those familiar with Los Angeles will recognize the heavy truck volume in the 710 freeway, one of the main routes to and from the ports of Los Angeles and Long Beach.

There is a host of challenges in producing a data processing workflow such as the transportation model described above. Since the data comes from a variety of sources, it may be expressed in different schemas, formats, and units. Therefore, the workflow needs to include many operations that perform different types of data conversion, for example, to translate a given measurement into different units — from tons to dollars to jobs to ton-miles to container units to trucks to passenger-car-equivalents. Also frequent is the need to translate economic data described in one industry/sector classification to another, for example, from the North American Industry Classification System (NAICS) to the Standard Classification of Transported Goods (SCTG), or from different versions of these classifications, for example, from NAICS 1997 to NAICS 2002.

There are many details that the abstract workflow of Figure 1(a) does not show. The full workflow, whose structure appears in Figure 1(b), contains over 50 data access and data processing operations. A detailed workflow that computes intra-regional data appears in Figure 10. This estimation model was originally implemented by a combination of manual steps and custom-designed programs. Our approach *automatically* generates such a data processing workflow in response to a user data request, including all the necessary data integration and translation operations.

In the remainder of the paper, we present our workflow generation approach. First, we define an ontology of the application domain and formally describe the data sources and operations according to this domain ontology using *relational* descriptions in an expressive logic. Second, we present our workflow planning algorithm and describe the different kinds of operations, including domain-specific and domain-dependent adaptor operations. Third, we provide an empirical evaluation of our approach. Fourth, we compare with related work. Finally, we discuss our contributions and our plans for future work.

2. DOMAIN AND DATA MODELING

One of the major challenges to automating computational workflows is understanding the data present in available sources, or required as inputs or produced as outputs of data processing operations. When presented with a new data source, a domain expert (after some effort) can accurately describe its contents. However, such understanding is rarely recorded and much less formalized unambiguously. To ensure a clear understanding of the semantics of the data, one needs to develop a formal ontology of each application domain. Thus, in close collaboration with our domain experts, we defined an ontology for transportation modeling.

2.1 Domain Ontology

We represent the concepts and relations of the domain ontology in the first-order logic language of the PowerLoom knowledge representation and reasoning system [20, 23, 9]. PowerLoom is the more expressive successor of the Loom [19] description logic. However, the techniques we present in this paper rely only on relational subsumption, i.e., query containment, so any other knowledge representation formalism from relational conjunctive queries [10, 15] to description logics such as DLR [16] could be applied.

PowerLoom provides logical inference services to the Argos system. In particular, we use its capabilities for *sub-*

[7]Specifically, User-Optimal-Strict (UO-S) On Network Assignment (NA) [25].

sumption of concepts and relations. Subsumption consists in proving that membership in a class or relation `P` logically implies membership in another class or relation `R`. It is essentially the same as *query containment* as used in database theory and information integration [10, 15]. However, PowerLoom operates on a more expressive language, a first-order language with recursion. PowerLoom is specially optimized to compute subsumption and is efficient in practice. First-order logical inference is undecidable, hence PowerLoom is incomplete. Nevertheless, in our experience we have defined an expressive domain ontology and PowerLoom proves the required inferences efficiently. A full description of the representational features of PowerLoom is outside the scope of this paper. We introduce the language by example and refer the interested reader to [23].[8] We assume the user has a basic knowledge of first-order logic.

Sample concept, instance, rule and constraint definitions of our transportation modeling ontology appear in Figure 3. The `Flow` concept represents a transfer of a product between two geospatial areas using a transportation mode measured during a particular time interval. For example, an instance of `Flow` is the domestic exports by `air` (a `Transportation-Mode`) of Pharmaceutical products from the Los Angeles Consolidated Metropolitan Statistical Area (LACMSA) in 2000, which amounts to (`hasValue`) 2226 million US dollars. The ontology also encodes information about well-known entities in the domain. For example, Figure 3(b) shows the fact that Los Angeles County (`g-LA`) is geographically contained in (is a `geoPartOf`) the LACMSA region (`g-LACMSA`), as well as Ventura County (`g-VT`), something not immediately apparent from the LACMSA name.[9] In addition, our ontology includes rules clarifying the semantics of the concepts and relations. For example, the *recursive* rule in Figure 3(c) specifies the transitivity of geospatial containment (`geoPartOf`). Finally, some complex constraints, such as disjointness, can be easily expressed in PowerLoom. For example, Figure 3(d) specifies that the different product classifications do not have any instances in common.

```
(defconcept Flow (?x) :<=>
 (exists (?o ?d ?p ?t ?u ?m ?v)
  (and (Data ?x)
       (hasOrigin ?x ?o) (Geo ?o)
       (hasDestination ?x ?d) (Geo ?d)
       (hasProduct ?x ?p) (Product ?p)
       (hasTimeInterval ?x ?t) (TimeInterval ?t)
       (hasUnit ?x ?u) (Unit ?u)
       (hasMode ?x ?m) (TransportationMode ?m)
       (hasValue ?x ?v) (Number ?v) )))

         (a) Concept Definition

(geoPartOf g-LA g-LACMSA) (USCounty g-LA)
(geoPartOf g-VT g-LACMSA) (USCounty g-VT)
(USGeo g-LACMSA) (TransportationMode tm-air)

         (b) Instance Assertions

(forall (?x ?z)
  (=> (exists (?y) (and (geoPartOf ?x ?y)
                        (geoPartOf ?y ?z)))
      (geoPartOf ?x ?z)))

         (c) Inference Rule

(mutually-disjoint-collection
  (setof IMPLAN NAICS SCTG SIC USC USCCOM))

      (d) Concept Disjointness Assertion
```

Figure 3: Argos Ontology: Sample Definitions

2.2 Relational Data Descriptions

Using this ontology we describe the data provided by sources, and required or computed by data processing services. In our domain, data are commonly represented as relational tables. We formally describe the tuples in such tables by concepts and relations of the ontology. Essentially, we associate a Local-as-View [15] definition to each source data table and each input and output of a data processing operation. For example, Figure 4 shows the description of a source table `LACMSA-2000-EMP` that provides the number of jobs in 2000 for each traffic-analysis zone (`TAZ`) contained in the LACMSA for products categorized following the 1999 Standard Industrial Classification (SIC) codes (with a granularity of 4 digits).

In our representation we can describe both complete and incomplete data sources. Defining a data source relation as *complete* means that it contains *all* the tuples that satisfy the ontology definition. We indicate completeness by using an if-and-only-if definition, that is, by using the logical biconditional (`<=>`). For example, the relation definition in Figure 4 states that the table contains values for *all* the products of type `Product-sic-4-1999` for *all* the TAZs in the LACMSA. Conversely, defining a data source relation as *incomplete* means that although all the tuples in the source relation satisfy the definition, the relation may not contain all possible such tuples. That is, there may be other sources that contain additional tuples that satisfy the definition. We indicate incompleteness by using an if definition, that is, by using only logical implication (`=>`). For example, defining the table `LACMSA-2000-EMP` as incomplete would mean that there could be tuples missing for some TAZ or some product.

```
(defrelation LACMSA-2000-EMP
   (?county ?jobs ?product ?taz) :<=>
 (exists (?o)
   (and (Measurement ?o)
        (hasGeo ?o ?taz) (TAZ ?taz)
          (geoPartOf ?taz ?county) (USCounty ?county)
          (geoPartOf ?taz g-LACMSA)
        (hasUnit ?o u-NumberOfJobs)
        (hasProduct ?o ?product)
          (Product-SIC-4-1999 ?product)
        (hasTimeInterval ?o year2000)
        (hasValue ?o ?jobs)(Number ?jobs) ))
```

Figure 4: Data Description for the LACMSA-2000-EMP source table

Complete and incomplete definitions correspond, respectively, to the *closed-world* and *open-world* semantics in data integration systems [11, 1, 15]. Web sources are often incomplete. For example, a website like imdb.com is incomplete. Although it contains information about a large number of movies, there is no guarantee that it covers every movie ever produced. However, the site oscars.com lists every movie

[8] PowerLoom syntax conforms to the Knowledge Interchange Format [12]. KIF is a proposed ANSI standard language for the interchange of knowledge among disparate programs. It has a prefix notation like Lisp. Variables start with '?'.

[9] The LACMSA comprises the counties of Los Angeles, Ventura, San Bernardino, Riverside and Orange.

that has been nominated or has received an Oscar, and is therefore complete for such movies. An advantage of our modeling methodology is that we can precisely define the contents of a source and specify whether it is a complete or incomplete set.

In our transportation modeling domain complete descriptions are customary. Moreover, complete data is often required by many types of operations. In particular, aggregation operations cannot be properly computed unless the system has a complete dataset. For example, computing the total output of a region for a given commodity cannot be accurately computed unless we had information from all the TAZs. Of course, in some cases approximations may be acceptable. For example, to approximate the average family income in Los Angeles County, it may be sufficient to use a source that provides family income for selected census tracts, instead of requiring a source with data for all census tracts. In this paper we focus on reasoning with complete descriptions, but our approach can handle both complete and incomplete data descriptions (see Section 3.1).

In the next section, we show how we use these data descriptions to specify the contents of sources and the inputs and outputs of operations.

3. AUTOMATICALLY COMPOSING DATA PROCESSING WORKFLOWS

Our objective is to provide a framework to automatically generate computational workflows that can answer user data requests based on available sources and data processing operations. We assume that most of the workflow components (sources and operations) have been developed independently and are outside our control. For example, operations can be deployed as web services that are hosted at different servers in the web, or they can be functions from third-party software libraries. Similarly, sources can be databases or other web services. In the remainder of the paper we will use the term operation and service interchangeably. We view a source as an operation that does not require inputs.

Automatically generating a workflow presents two main challenges. First, sources and operations may use different schemas. Second, the data produced by a source or operation may not be input directly into other operations, but need some kind of transformation.

The Argos framework addresses both these challenges. First, we resolve the semantic heterogeneity by describing the data in a common ontology. As we presented in the previous section, we associate a Local-as-View [15] description with each data relation consumed or produced by a service. Thus, a service is represented by the intensional specification of its input/output signature. Second, we provide a set of *domain-independent relational* operations and a framework to define *generic domain-dependent* operations that can bridge the differences between the input required by an operation and the output provided by another.

In this section, we first present the planning algorithm that generates the workflow. Then, we describe the three types of operations that Argos supports: domain-dependent, domain-independent, and generic domain-dependent.

3.1 Planning Algorithm

Our planner for workflow composition performs a regression search in plan space in the same fashion as partial-order planners such as UCPOP [22] and Sage [18]. The planner starts with the user data request as a goal and terminates the search when it finds a complete plan, that is, a plan where all data inputs to the component services are produced by other operations or sources. The planning algorithm appears in Figure 5.

```
planner(P, A)
  select pair < g, s1 > from agenda A
  choose < P', A' > ∈ plan-refinements(P, A, < g, s1 >)
  planner(P', A')

plan-refinements(P, A, < g, s1 >)
  union(
    reuse-service(P, A, < g, s1 >)
    add-domain-service(P, A, < g, s1 >)
    ;; add generic domain-dependent operators
    add-product-conversion(P, A, < g, s1 >)
    add-unit-conversion(P, A, < g, s1 >)
    ;; add relational domain-independent operators
    add-selection(P, A, < g, s1 >)
    add-projection(P, A, < g, s1 >)
    add-join(P, A, < g, s1 >)
    add-union(P, A, < g, s1 >) )

reuse-service(P, A, < g, s1 >)
  ∀s2 ∈ services(P) such that
      ∃g2 ∈ outputs(s2) such that equivalent(g2, g)
    P' := add-data-link(P, < s2, g2, g, s1 >)
    A' := A − {< g, s1 >}
    push < P', A' > into refinements
  return refinements

add-domain-service(P, A, < g, s1 >)
  ∀s2 ∈ services(Domain) such that
      ∃g2 ∈ outputs(s2) such that equivalent(g2, g)
    P' := add-data-link( add-step(P, s2), < s2, g2, g, s1 > )
    A' := A − {< g, s1 >} ∪ {< g2i, s2 > / g2i ∈ inputs(s2) }
    push < P', A' > into refinements
  return refinements
```

Figure 5: Argos Planning Algorithm

The algorithm keeps an agenda A of services with unachieved inputs. Each element in the agenda is a pair $< g, s >$ that consists of a data description g which is an unachieved input of service s. The planner non-deterministically chooses a plan refinement that solves an element from the agenda (that is, it searches the space of plans). In the Argos planner, we consider three types of plan refinements: domain-dependent, generic domain-dependent and domain-independent. We describe each of these types in the following subsections.

In each plan refinement, the basic operation is to satisfy the input of a service with the output of another service. In order to ensure that the input and the output data relations are semantically compatible, the planner performs an *equivalence* test, that is, it checks subsumption in both directions. If equivalent, it establishes a *data link* from the output relation of one operator to the input relation of another. This mechanism is analogous to the establishment of a causal link in plan-space planning [22] where the effect of an operator produces a precondition of another operator. However, instead of using simple unification to match a precondition with an effect predicate, our planner uses subsumption (equivalence), because our inputs and outputs are universally quantified formulas that represent data re-

lations. If an operation allows incomplete inputs, then the planner does not need to compute equivalence. In order to establish the data link between two operations, it suffices to prove that the description for the output of the provider operation is contained in the input of the consumer operation.

Since in our planning domain there are no negated effects (the operations do not destroy information), there is no need for threat detection as in classical partial-order planning.

The Argos planner uses a best-first search strategy with a simple (non-admissible) heuristic that prefers to work on plans with the least number of unachieved inputs and contains the least operations already in the plan. As a heuristic optimization, the planning algorithm prefers to reuse an operation already in the plan rather than to insert a new one. This strategy usually leads to plans with a minimal number of operators in an efficient manner.

3.2 Domain-Dependent Services

Domain-dependent services are simply described by an input/output signature with predefined data relations. For example, Figure 6(b) shows the aggregation operation `op--EmpByTAZ` that takes as input the data described by the relation `LACMSA-2000-EMP-Point` shown in Figure 6(a), and produces as output the data described by the relation `LACMSA--2000-EMP`, whose definition appeared in Figure 4. Intuitively, the service sums the number of jobs for each SIC industry code for the work locations within a TAZ, for all TAZs in the LACMSA region.

A service can have multiple input and output data relations. The binding to the implementation of the service is specified by the `hasImplementation` property. The implementation can be a web service or a local program.

The `add-domain-service` plan refinement adds a domain-dependent service to the plan. It simply looks for services with some output that can satisfy an unachieved input by testing subsumption/equivalence directly.

```
(defrelation LACMSA-2000-EMP-Point
   (?g ?county ?jobs ?product ?taz) :<=>
 (exists (?o)
   (and (Measurement ?o)
        (hasGeo ?o ?g) (USPoint ?g)
          (geoPartOf ?g ?taz) (TAZ ?taz)
          (geoPartOf ?taz ?county) (USCounty ?county)
          (geoPartOf ?county g-LACMSA)
        (hasUnit ?o u-NumberOfJobs)
        (hasProduct ?o ?product)
          (Product-SIC-4-1999 ?product)
        (hasTimeInterval ?o year2000)
        (hasValue ?o ?jobs)(Number ?jobs) )))
```

(a) Service Input

```
(Service op-EmpByTAZ)
  (hasInput op-EmpbyTAZ LACMSA-2000-EMP-Point)
  (hasOutput op-EmpByTAZ LACMSA-2000-EMP)
  (hasImplementation op-EmpByTAZ "imp-EmpByTAZ")
```

(b) Service Definition

Figure 6: Sample Domain-Dependent Service

3.3 Domain-Independent Adaptors

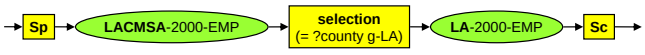
In order to bridge the inputs and outputs of different services, Argos provides a set of built-in domain-independent *adaptor* services that correspond to the relation algebra operations: selection, projection, join, and union.

Selection. The selection plan refinement checks whether a data relation can be achieved by a selection operation on the output of an existing service. However, since our descriptions are significantly more expressive that those used in databases, we need to take special care that such additional constraints produce the desired relations and can be computed from available relations. This may involve reasoning with background information in the ontology, not just the data definitions.

We describe the algorithm for the `add-selection` plan refinement by example. Assume that a service `Sc` requires as input the employment data for the TAZs in Los Angeles County, as described by the `LA-2000-EMP` relation of Figure 7(a). Further assume that there is another service `Sp` that is able produce the employment data for all the TAZs of the LACMSA, that is, one of the outputs of `Sp` is the `LACMSA-2000-EMP` relation described in Figure 4. Note the differences between the two data descriptions. The `LA--2000-EMP` table has three columns and contains data only for TAZs in Los Angeles county (`g-LA`). The `LACMSA-2000--EMP` table has four columns and contains data for TAZs in the five-county LACMSA. Intuitively, the system can prove that since Los Angeles is a geographical part of the LACMSA (cf. Figure 3), the relation `LA-2000-EMP` is contained in `LACMSA-2000-EMP`. Moreover, since `LACMSA-2000--EMP` has a column with the county, the planner can introduce a selection operation `(= ?county g-LA)` that adapts the output of the producer service `Sp` to the input required by the consumer service `Sc`. The resulting plan is shown in Figure 7(b).

```
(defrelation LA-2000-EMP (?jobs ?product ?taz) :<=>
  (exists (?o)
    (and (Measurement ?o)
         (hasGeo ?o ?taz) (TAZ ?taz)
           (geoPartOf ?taz g-LA)
         (hasUnit ?o u-NumberOfJobs)
         (hasProduct ?o ?product)
           (Product-SIC-4-1999 ?product)
         (hasTimeInterval ?o year2000)
         (hasValue ?o ?jobs)(Number ?jobs) ))
```

(a) Input Required by Service `Sc`



(b) Plan with Relational Selection Adaptor

Figure 7: Domain-Independent Adaptor: Selection

Projection, Join and Union. The projection, join and union plan refinements have a similar purpose as the corresponding relational algebra operations. They introduce a projection, join or union service to enable the match of inputs and outputs. The algorithms for these refinements are analogous to the selection refinement we described above.

The algorithm for the projection plan refinement searches for an output relation of a service whose projection into the desired attributes is equivalent to/contained in the desired input relation. The algorithm for the join (resp. union) plan refinement searches for outputs relations of services whose conjunction (resp. disjunction) is equivalent to/contained in the desired input relation.

A detailed description of the logical machinery that makes the search for candidates relations in the plan refinements efficient is outside of the scope of this paper. We briefly note that each data relation is associated with a set of distinguished concepts in the ontology. In order to efficiently locate relations relevant for a given plan refinement, the algorithm first computes subsumption among the distinguished concepts. Then, it filters the associated relations according to the types of theirs arguments. In general, an input relation of a service can be seen as a query and the outputs of services as view definitions, thus to match them we need to solve a query rewriting problem [15].

3.4 Generic Domain-Dependent Adaptors

There are a variety of operators that lie between the completely domain-specific operators that are described by predefined input and output datasets (Section 3.2), and the domain-independent operators that are applicable to any dataset description regardless of the domain (Section 3.3).

A situation that occurs repeatedly in our domain is the need for translation of data expressed in one product classification into another. Economic data is reported in a variety of classifications, including the Standard Classification of Transported Goods (SCTG), the Standard Industrial Classification (SIC), the North American Industry Classification System (NAICS), among many others. Unfortunately, there is no shortage of *standard* classifications. Thus, when integrating data from different sources and operations, the system must translate between these classifications.

By analyzing nationwide data, our domain experts have carefully derived what proportion of a product category in one classification would correspond to another product category in another classification. For example, they estimate that a value for SIC code 1382 "Oil and Gas Field Exploration Services" can be assigned to NAICS codes 213112 "Support Activities for Oil and Gas Operations" and 541360 "Geophysical Surveying and Mapping Services" in proportions of 0.58 and 0.42, respectively.[10] Given such `Product-Conversion` tables, it is a relatively straightforward computation to translate data expressed in one product classification into another.

Product conversion is a prime example of a generic, but domain-dependent operator. We could certainly define a host of domain-specific operators in the format of Section 3.2 that specify translations between pairs of product classifications. However, defining such operations manually would be tedious and error-prone. Thus, we added a generic product conversion refinement to the planner.

When the planner needs to satisfy a given request for products in a classification C2, the `add-product-conversion` refinement introduces a product conversion service and subgoals for obtaining the data in a classification C1 and a conversion table from C1 to C2. As an optimization, the refinement checks the service descriptions to ensure that the C1-to-C2 conversion table is the output of an available service. Figure 8 shows an example. Assume that a service `Sc` requires as input the employment data for LACMSA by 6-digit NAICS industry codes as described by the relation in Figure 8(a). First, the algorithm retrieves from the ontology conversion tables into 6-digit NAICS by issuing the query shown in Figure 8(b). Assume that it finds a source `Sp` that produces the relation `SIC2NAICS` of Figure 8(c). Second, the algorithm adds a product conversion service and subgoals on obtaining a data relation with the same definition as the originally desired relation except that the product classification is in SIC instead of NAICS codes. The subgoal is described with the anonymous[11] relation shown in Figure 8(d) The resulting plan is shown in Figure 8(e).

[10]In particular, for NAICS and SIC, our domain experts used employment data from the Census Bureau (Table 2 in http://www.census.gov/epcd/ec97brdg/97x-cs3.pdf)

3.5 Sample Workflow

Consider a user request for the intra-regional supply (production) in millions of US dollars for the year 2000 for fuel products (USC commodity code 10)[12] for all the TAZs in the LACMSA region. The formal definition for the requested data, which for convenience we labeled `Data-Rel-Intra-Regional-Supply-2000-USCCOM10`, appears in Figure 9.

The workflow that the planner generates in response to this data request appears in Figure 10, where rectangles represent sources and operations, and ellipses represent the data-relations that are produced or consumed by them. The data relations show their name and arguments.

The workflow contains operations from each of the types we described in the previous sections. `SOURCE-SCAG2000EMP` is a source that provides employment data for work locations within the LACMSA by SIC-4-1999 codes. That is, it outputs the relation from Figure 6(a),[13] which in turn is the input of the `OP-EMPBYTAZ` domain-specific operator that was described in Figure 6 of Section 3.2.

Adaptor services can be chained. In particular, note the chain of three system-generated product conversions (`SYSGEN--OP-PRODUCT-CONVERSION-19, -15` and `-13`) that translate employment data from NAICS to SCTG to USC to USCOM industry sector codes.

Finally, since the user requests the intra-regional supply for only one particular product (`prod-usccom-10`), the planner introduces a domain-independent select operation (`SYSGEN-OP-SELECTION-1`) to produce the required output.

4. EMPIRICAL EVALUATION

We tested our workflow planner using two ontologies. The first is our production ontology, *Argos* (`A`), that was created by consulting our domain experts. It contains 162 concepts, 67 relations, and 28 domain service descriptions (17 sources and 11 operations, with a total of 37 input and 32 output data relations). The data sources used different product classifications (SCTG, NAICS, SIC, IMPLAN). With the help of 5 conversion tables defined by the domain experts, these products classifications are eventually mapped into one product classification for uniformity. The second ontology, *Extended Argos* (`EA`), we defined with the purpose of testing the planner in a domain where cycles of operations are possible. This ontology includes 17 product con-

[11]Anonymous relations in PowerLoom are denoted with the `kappa` symbol (by analogy to anonymous functions in the lambda calculus).

[12]`prod-usccom-10` includes coal, crude petroleum, gasoline, aviation fuel, fuel oils, and related products.

[13]The workflow in Figure 10 shows the full normalized names we use in our production ontology. `DATA-REL-EMPLOYMENT-2000-LACMSA-POINT-SIC` corresponds to `LACMSA-2000-EMP-Points`, and similarly for the other relations. In the paper we have shortened some of the names for ease of presentation.

```
(defrelation LACMSA-2000-EMP-NAICS
  (?county ?jobs ?product ?taz) :<=>
 (exists (?o) (and (Measurement ?o)
   (hasGeo ?o ?taz) (TAZ ?taz)
     (geoPartOf ?taz ?county) (USCounty ?county)
     (geoPartOf ?taz g-LACMSA)
   (hasUnit ?o u-NumberOfJobs)
   (hasProduct ?o ?product)
     (Product-NAICS-6-2002 ?product)
   (hasTimeInterval ?o year2000)
   (hasValue ?o ?jobs)(Number ?jobs) )))
```

(a) Desired Data Relation

```
(retrieve all (?s ?x)
  (and (source ?s) (hasOutput ?s ?x)
       (subset-of ?x
         (kappa (?fp ?tp ?proportion)
           (exists (?o)
             (and (ProductConversion ?o)
                  (fromProduct ?o ?fp)
                  (toProduct ?o ?tp)
                    (Product-NAICS-6-2002 ?tp)))))))
```

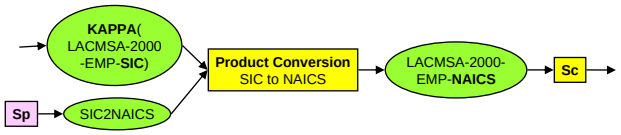
(b) Ontology Query to Retrieve Product Conversion Relations

```
(defrelation SIC2NAICS (?fp ?tp ?proportion) :<=>
  (exists (?o)
   (and
    (ProductConversion ?o)
    (fromProduct ?o ?fp) (Product-SIC-4-1999 ?fp)
    (toProduct ?o ?tp)(Product-NAICS-6-2002 ?tp)
    (hasValue ?o ?proportion) (Number ?proportion))))
```

(c) Product Conversion Relation

```
(kappa (?county ?jobs ?product ?taz) :<=>
 (exists (?o) (and (Measurement ?o)
   (hasGeo ?o ?taz) (TAZ ?taz)
     (geoPartOf ?taz ?county) (USCounty ?county)
     (geoPartOf ?taz g-LACMSA)
   (hasUnit ?o u-NumberOfJobs)
   (hasProduct ?o ?product)
     (Product-SIC-4-1999 ?product)
   (hasTimeInterval ?o year2000)
   (hasValue ?o ?jobs)(Number ?jobs) )))
```

(d) Data Relation Subgoal



(e) Plan with Product Conversion Adaptor

Figure 8: Generic Domain-Dependent Adaptor: Product Conversion

version tables that can convert between any pair of classifications and may lead to infinite cycles of product conversions. It contains 173 concepts, 79 relations, 40 domain services (29 sources and 11 operations, with a total of 37 inputs and 44 outputs).

Table 1 shows the planning performance on 18 typical queries. For example, the third row, query `Q3`, shows the results for the `LA-2000-EMP` relation of Figure 7. Query `Q17` asks for the total (intra- plus inter-regional) demand for all

```
(defrelation
  Data-Rel-IntraRegional-Supply-2000-USCCOM10
   (?otaz ?dollars) :<=>
  (exists (?o)
    (and (Flow ?o)
      (hasOrigin ?o ?otaz) (TAZ ?otaz)
      (geoPartof ?otaz g_LACMSA)
      (hasDestination ?o ?dtaz) (TAZ ?dtaz)
      (geoPartof ?dtaz g_LACMSA)
      (not (= ?otaz ?dtaz))
      (hasTimeInterval ?o year2000)
      (hasUnit ?o u_MillionUSD)
      (hasMode ?o tm_allModes)
      (hasProduct ?o prod_usccom_10)
      (hasValue ?o ?dollars) (EXPORTS ?dollars))))
```

Figure 9: User Data Request

TAZs in the LACMSA region for all transportation modes. Query `Q18` asks for the truck traffic volume for all links of the LACMSA highway network. In response to `Q18`, Argos generates the workflow of Figure 1(b). Figure 2 shows the results of query `Q18` displayed in ArcGIS.

We tested the workflow planning algorithm with the two ontologies (`A`, `EA`). We report the number of services (sources and operations) and data links in the resulting workflows, the total workflow generation time and the portion spent in PowerLoom reasoning (both in seconds), the number of subsumption tests, and the number of search nodes (partial plans) generated and visited. The experiments were run on a laptop running Windows XP with a 2GHz processor and 2GB of memory.

Overall, we find the planning performance satisfactory for data processing workflows, where the execution time of the workflow dominates. For example, consider the queries that generate the largest workflows: `Q17` and `Q18`. For `Q17` the planning time is 61.58 seconds, generating a workflow with 53 services. The execution time is 224 seconds, processing a total of 2056247 tuples, and producing a result relation with 74350 tuples. For `Q18` the planning time is 73.29 seconds, generating a workflow with 54 services. The execution time is 1280 seconds, processing a total of 1980860 tuples, and producing a result relation with 89356 tuples.[14]

The results show that the increased possibilities for product conversion in ontology `EA` increases the planning time in the largest plans (from approx. 73 to 127 seconds) due to an increased number of subsumption queries, but on the other hand they lead to shorter plans (from 54 to 51 services).

We also tested unsatisfiable requests that could lead to an infinite chain of product conversion operators. In ontology `A` the products conversions form an acyclic directed graph, so there is no possibility of infinite subgoaling. However, in ontology `EA` there are cycles. Thus, we added a limit to the depth of chains of instantiations of the same operation, in our example, product conversion chains. Experimentally, to prove a query unsatisfiable with chain limits of length 3 and 5, the planner using ontology `A` takes 4 and 4.17 seconds, respectively, and using ontology `EA` it takes 89.70 and 236.42 seconds, respectively. There is also a practical reason for such limit. Since each product conversion is an approximation, a long chain would produce very low quality data.

[14]The last step in the workflow for Q18, the network equilibrium algorithm which computes truck traffic in each link in the highway system, is particularly time consuming.

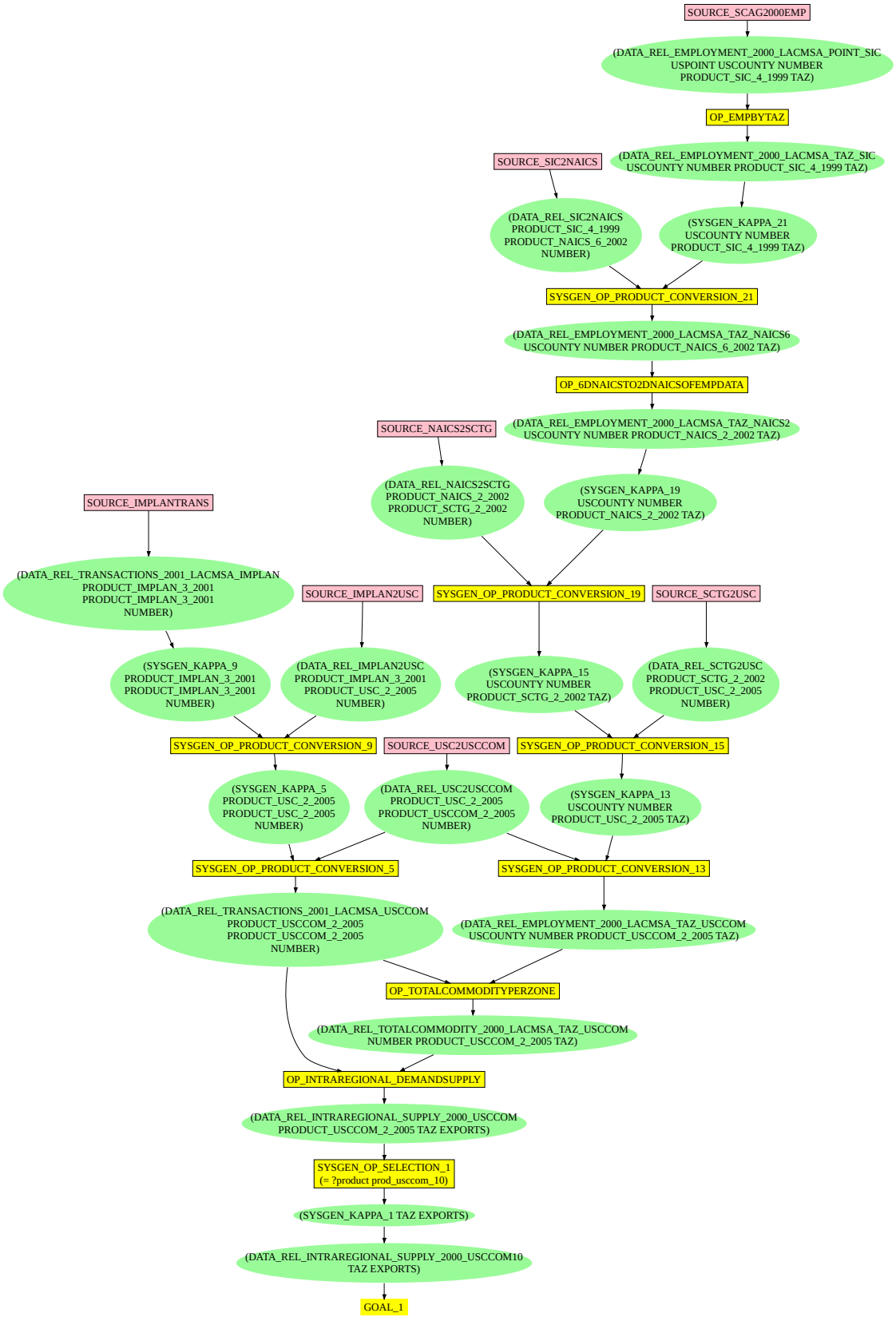


Figure 10: Sample workflow that computes the intra-regional supply of fuel products (USC commodity product code 10) for the TAZs in the LACMSA region in millions of US dollars for the year 2000

Query	Services		DataLinks		Planning Time (s)		PowerLoom Time (s)		Subsumptions		Plans Generated		Plans Visited	
	A	EA	A	EA	A	EA	A	EA	A	EA	A	EA	A	EA
Q1	2	2	1	1	1.98	2.22	1.98	2.2	8	8	1	1	1	1
Q2	3	3	2	2	2.74	3.36	2.74	3.36	16	16	2	2	2	2
Q3	4	4	3	3	4.36	5.23	4.33	5.15	30	36	3	9	3	3
Q4	5	5	4	4	3.69	3.61	3.52	3.59	34	42	4	12	4	4
Q5	6	4	5	3	3.27	5.89	3.23	5.84	50	82	8	23	5	7
Q6	11	11	10	10	5.22	6.2	5.19	6.19	76	76	10	10	10	10
Q7	11	11	10	10	5.47	5.89	5.45	5.84	76	76	10	10	10	10
Q8	11	11	10	10	5.52	6.22	5.48	6.2	76	76	10	10	10	10
Q9	11	11	10	10	5.5	5.89	5.44	5.86	76	76	10	10	10	10
Q10	12	9	11	8	5.94	16.45	5.94	16.3	117	325	16	98	13	31
Q11	17	12	17	12	6.22	19	6.09	18.66	151	389	24	121	19	39
Q12	18	13	19	14	7.09	19.61	6.92	19.22	159	397	27	124	21	41
Q13	18	13	19	14	7.14	19.61	7	19.33	159	397	27	124	21	41
Q14	33	33	44	44	15.64	16.84	13.36	14.44	250	250	44	44	44	44
Q15	33	33	44	44	15.88	16.86	13.61	14.48	250	250	44	44	44	44
Q16	53	48	69	64	60.25	89.59	22.34	68.59	442	722	78	187	71	95
Q17	53	48	69	64	61.58	90.39	23.54	69.1	442	722	78	187	71	95
Q18	54	51	71	67	73.29	127.67	69.16	115.75	435	600	78	171	75	110

Table 1: Experimental Results

5. RELATED WORK

Our planner is inspired by the representation of information gathering actions of the Sage planner [18] of the SIMS [6] mediator, where knowledge preconditions and effects were represented as queries. However, our work has several major differences with SIMS. First, our domain and service descriptions, expressed in PowerLoom, are significantly more expressive than SIMS's use of the Loom description logic. For example, relational subsumption over recursive descriptions, as required by the `geoPartOf` relation in the selection example of Figure 7 could not be performed in SIMS. Second, our planner proves subsumption to link inputs and outputs, while SIMS relied on syntactic matching to establish causal links. On the other hand, SIMS included optimization techniques [5], like pushing selections, that we have not (yet) incorporated in our planner.

Our domain modeling builds on the idea of faceted representations (e.g. [24]). In particular we structured some of the main concepts in the ontology along basic dimensions like location, time, product, similarly to the Energy Data Collection project [2]. However, we have a more refined domain ontology. More critically, since [2] was based on SIMS, its ability to describe arbitrary data sources was limited.

There is research on mixed-initiative composition of scientific workflows in physics [17] and bioinformatics [27] that leverages semantic descriptions. However, Argos uses a more expressive description language and focuses on automatic composition.

The TAMBIS system [7] performed integration of heterogeneous data and analysis tools in a bioinformatics domain. They used a domain ontology expressed in the GRAIL description logic as a basis of the integration. We share many of the same goals as TAMBIS. However, our local-as-view relational description and use of relation subsumption (as opposed to TAMBIS's concept subsumption) yields a more expressive and principled system that can produce more flexible workflows.

There has been research on automatic web service composition within the AI planning community. Blythe et al. [8] developed a planner to compose executable grid workflows, which has been applied to physics problems. Sirin et al. [26] applied hierarchical task network planning to automatically compose web services. McIlraith and Son [21] composed web services using procedures in the logical action language Golog. These systems handle web services with negative preconditions and effects. However, Argos offers a more expressive data representation and uses relational subsumption as the primary mechanism to construct the plans.

6. DISCUSSION AND FUTURE WORK

We have presented a logic-based approach to automatically compose data processing workflows. We describe data sources and operations using a formal ontology. The planner uses the ontology and logical inference, provided by the PowerLoom reasoner, to construct workflows that answer user data requests. We have demonstrated the system in a transportation modeling domain.

In the immediate future, we plan to scale the number of operators and sources that we have modeled and implemented for our commodity flow domain. In addition, we want to test broaden the scope of work to other questions of spatial urban structure. Such extension would test how general our current ontology is. We expect that we can reuse most of the current ontology and operations.

We intend to incorporate cost optimization knowledge into the planner, so that it generates more efficient plans. Some of the optimizations include pushing selections closer to the sources, and choosing between alternative services with different performance/quality-of-service characteristics.

Finally, although in this paper we have focused on representing and reasoning with *complete* data relations, the logical machinery in the planner can also handle incomplete data sources. The planner just needs to compute subsumption in one direction instead of both. We plan to investigate domains that mix complete and incomplete data.

7. ACKNOWLEDGMENTS

We would like to thank the current and former members of the Argos group for their contributions, including our social scientists: Genevieve Giuliano, Peter Gordon, Qisheng Pan, LanLan Wang, and JiYoung Park; computer scientists: Stefan Decker, Andreas Harth, Mountu Jinwala, Naqeeb Abbasi, Matthew Weathers and Karanbir Jassar; as well as our government partner agencies. Special thanks to Qisheng

Pan for contributing his gravity model and freight distribution code to the Argos project.

This material is based upon work supported by the National Science Foundation under Award No. EIA-0306905. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Seattle, WA, 1998.

[2] J. L. Ambite, Y. Arens, E. Hovy, A. Philpot, L. Gravano, V. Hatzivassiloglou, and J. Klavans. Simplifying data access: The energy data collection (EDC) project. *IEEE Computer, Special Issue on Digital Government*, 34(2), February 2001.

[3] J. L. Ambite, G. Giuliano, P. Gordon, A. Harth, L. Wang, Q. Pan, and S. Decker. Argos: Dynamic composition of web services for goods movement analysis and planning. In *Proceedings of the 5th Annual National Conference on Digital Government Research (dg.o2004)*, Seattle, Washington, 2004.

[4] J. L. Ambite, G. Giuliano, P. Gordon, Q. Pan, and S. Bhattacharjee. Integrating heterogeneous sources for better freight flow analysis and planning. In *Proceedings of the Second National Conference on Digital Government Research (dg.o 2002)*, Redondo Beach, California, 2002.

[5] J. L. Ambite and C. A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformational approach. *Artificial Intelligence*, 118(1-2):115–161, Apr. 2000.

[6] Y. Arens, C. A. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration*, 6(2/3):99–130, 1996.

[7] P. G. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens. TAMBIS: Transparent access to multiple bioinformatics information sources. In *6th International Conference on Intelligent Systems for Molecular Biology*, Montreal, Canada, 1998.

[8] J. Blythe, E. Deelman, and Y. Gil. Automatically composed workflows for grid environments. *IEEE Intelligent Systems*, pp 16–23, July/August 2004.

[9] H. Chalupsky and T. Russ. WhyNot: Debugging failed queries in large knowledge bases. In *Proceedings of the 14th Innovative Applications of Artificial Intelligence Conference (IAAI-02)*, Menlo Park, CA 2002.

[10] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the 9th ACM Symposium on Theory of Computing (STOC)*, Boulder, CO, 1977.

[11] O. M. Duschka. *Query Planning and Optimization in Information Integration*. PhD thesis, Stanford University, 1997.

[12] M. R. Genesereth. Knowledge interchange format. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, 1991.

[13] G. Giuliano, P. Gordon, Q. Pan, J. Park, and L. Wang. Estimating freight flows for metropolitan highway networks using secondary data sources. In *Proceedings of Transportation Research Board Commodity Flow Survey Conference*, Transportation Research Circular, E-C088, July 2005.

[14] P. Gordon and Q. Pan. Assembling and processing freight shipment data: developing a gis-based origin-destination matrix for southern california freight flows. Final report of METRANS research project 99-25, University of Southern California, 2001. www.metrans.org/research/final/99-25_Final.pdf

[15] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[16] I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a description logic. In *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR'2000)*, 2000.

[17] J. Kim, M. Spraragen, and Y. Gil. An intelligent assistant for interactive workflow composition. In *International Conference on Intelligent User Interfaces*, Madeira, Portugal, 2004.

[18] C. A. Knoblock. Building a planner for information gathering: A report from the trenches. In *3rd International Conference on Artificial Intelligence Planning Systems*, Edinburgh, Scotland, 1996.

[19] R. MacGregor. A deductive pattern matcher. In *Proceedings of the 7th National Conference on Artificial Intelligence*, Saint Paul, MN, 1988.

[20] R. MacGregor. A description classifier for the predicate calculus. In *12th National Conference on Artificial Intelligence*, Seattle, WA, 1994.

[21] S. McIlraith and T. C. Son. Adapting Golog for composition of semantic web services. In *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, 2002.

[22] J. S. Penberthy and D. S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, 1992.

[23] PowerLoom. The PowerLoom knowledge representation & reasoning system. www.isi.edu/isd/LOOM/PowerLoom, 2003.

[24] W. Pratt, M. A. Hearst, and L. M. Fagan. A knowledge-based approach to organizing retrieved documents. In *16th National Conference on Artificial Intelligence*, Menlo Park, CA, 1999.

[25] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice Hall, New Jersey, 1985.

[26] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau. HTN planning for web service composition using SHOP2. *Web Semantics*, 1(4):377–396, 2004.

[27] R. D. Stevens, A. J. Robinson, and C. A. Goble. myGrid: personalised bioinformatics on the information grid. *Bioinformatics*, 19(1):302–304, 2003.