Automatically Composing Data Workflows with Relational Descriptions and Shim Services*

José Luis Ambite and Dipsy Kapoor

University of Southern California, Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292, USA
{ambite,dipsy}@isi.edu
http://www.isi.edu/~argos

Abstract. Many scientific problems can be represented as computational workflows of operations that access remote data, integrate heterogeneous data, and analyze and derive new data. Even when the data access and processing operations are implemented as web or grid services, workflows are often constructed *manually* in languages such as BPEL. Adding semantic descriptions of the services enables automatic or mixed-initiative composition. In most previous work, these descriptions consists of semantic types for inputs and outputs of services or a type for the service as a whole. While this is certainly useful, we argue that is not enough to model and construct complex data workflows.

We present a planning approach to *automatically* constructing data processing workflows where the inputs and outputs of services are *relational* descriptions in an expressive logic. Our workflow planner uses relational subsumption to connect the output of a service with the input of another. This modeling style has the advantage that adaptor services, so-called *shims*, can be *automatically* inserted into the workflow where necessary.

1 Introduction

Much of the work of scientists, economists, and engineers is consumed by accessing, integrating, and analyzing data. Recently, there has been a significant effort to support computational workflows in fields such as physics (e.g., [11,6]) and bioinformatics (e.g. [5,22]). This research leverages domain ontologies to facilitate workflow construction, usually by defining the workflow components as semantic web services. Such semantic descriptions are of two kinds: (1) the service as a whole is classified according to an ontology of service types (e.g., [21,11]), or (2) the inputs and outputs of services are typed with concepts defined in a domain ontology (e.g. [5,16,11,20]). Though useful for service discovery, these approaches do not describe the data manipulated by the service in sufficient detail. First, the inputs and outputs of a service are usually related, so that a service is better described as having relational inputs and outputs instead of a list of apparently independent single-type inputs or outputs. In data-intensive

* Work supported by National Science Foundation Award EIA-0306905.

K. Aberer et al. (Eds.): ISWC/ASWC 2007, LNCS 4825, pp. 15–29, 2007.

applications tabular data is the norm, so services that process such data must consume and produce tables natively. Second, having relational descriptions of the services' inputs and outputs allows our planner to *automatically* introduce adaptor services, so-called *shims* [10], that transform the output of one service to the input of another. In this paper, we present the Argos framework to (1) describe data processing services and (2) automatically generate new data on demand by automatically composing data processing workflows.

We have applied the Argos approach to a transportation modeling domain [7,2], which we 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.

As an example consider Figure 1(a) that shows an abstract workflow that computes truck traffic in the highways of the Los Angeles Consolidated Statistical Metropolitan Area (LACMSA). The workflow estimates the *intra-regional* trade based on employment data and an input-output transaction model of the local economy (provided 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 data from the Commodity Flow Survey (CFS) of the US Census Bureau, the Waterborne Commerce of the United States (WCUS), and airport statistics from RAND. The inter-regional 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 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 assigns the freight flow to specific highway links. Figure 1(c) shows graphically the final result of the workflow: the flow of freight in the LACMSA highway network.

There are many challenges in producing data processing workflows 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 perform different types of data conversion, for example, to translate between different units (e.g., from tons to dollars to jobs to container units to passenger-car-equivalents), or to translate economic data described in one industry/sector classification to another (e.g., from the North American Industry Classification System – NAICS – to the Standard Classification of Transported Goods – SCTG –, or from the NAICS 1997 version of the standard to the NAICS 2002 version).

The workflow of Figure 1(a) abstracts many details. The full workflow, whose structure appears in Figure 1(b), contains over 50 data access and data processing operations. This estimation model was originally implemented by a combination

[1] A TAZ is a spatial region consisting of several census blocks. The LACMSA is partitioned into 3165 TAZs.

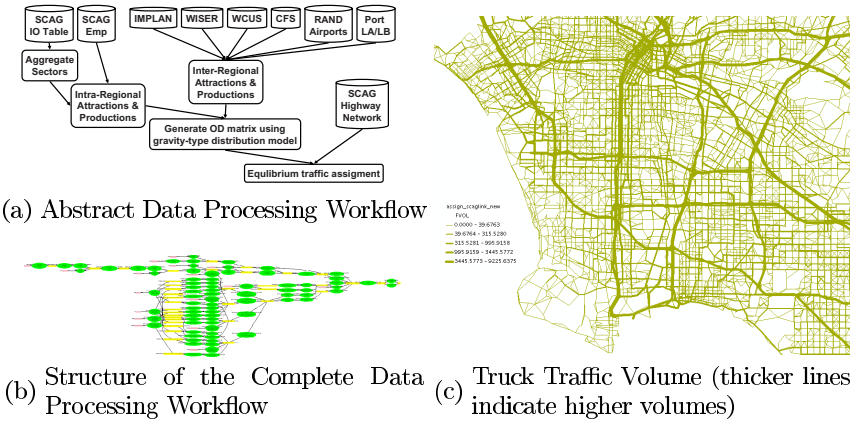


Fig. 1. Estimating Truck Traffic in the Los Angeles Highway Network

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.

The remainder of the paper is structured as follows. First, we describe the domain ontology and the data representation. Second, we present our workflow planning algorithm that uses both domain-dependent and domain-independent services. Third, we provide a empirical evaluation. Fourth, we compare with related work. Finally, we discuss our contributions and plans for future work.

2 Domain and Data Modeling

One of the major challenges to automating computational workflows is understanding source data, and data consumed or produced by services. To provide a clear understanding of the semantics of the data, we describe the data according to an ontology of the application domain.

Domain Ontology. We represent the domain ontology in the first-order logic language of the PowerLoom system [14,18]. PowerLoom is the more expressive successor of the Loom [13] description logic. PowerLoom is specially optimized to compute both concept and relation subsumption. 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. 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 [8] to description logics such as DLR [9] could be applied.

Sample concept, instance, rule and constraint definitions of our transportation modeling ontology appear in Figure 2. 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 `TransportationMode`) of Pharmaceutical products from 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 2(b) shows the fact that Los Angeles County (`g-LA`) is geographically contained in (is a `geoPartOf`) the LACMSA region (`g-LACMSA`), as is Ventura County (`g-VT`), something not immediately apparent from the LACMSA name.[2] The ontology includes rules definitions, such as the *recursive* rule in Figure 2(c) that specifies the transitivity of geospatial containment (`geoPartOf`), as well as disjointness constraints, such as the statement in Figure 2(d) that specifies that different product classifications do not have 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

Fig. 2. Argos Ontology: Sample Definitions

Relational Data Descriptions. Using this ontology we describe the data provided by sources, and required or computed by services. In our domain, data are commonly represented as relational tables. We formally describe the tuples in such tables by formulas over concepts and relations of the ontology. Essentially, we associate a Local-as-View [8] definition to each source relation and to each input and output of a data processing operation. For example, Figure 3(a) shows the description of a source table `LACMSA-2000-EMP` that provides the number of jobs in 2000 for each `TAZ` in the LACMSA for products categorized following the 1999 Standard Industrial Classification (SIC) with a granularity of 4 digits.

In Argos we used *factored* data descriptions. Instead of considering the body of a description as a collection of predicates all of equal importance, we group the predicates into meaningful concepts for the domain, and then use these concepts as appropriate in the body of the relational definitions. For example, Argos uses the factored definitions of Figure 3(b) instead of the direct description of Figure 3(a). Although the direct and the factored representations are semantically

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

```
(defrelation R-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)
      (hasProduct ?o ?product)
        (Product-SIC-4-1999 ?product)
      (hasTimeInterval ?o year2000)
      (hasUnit ?o u-NumberOfJobs)
      (hasValue ?o ?jobs)(Number ?jobs)))

    (a) Direct Definition

(relation-concept-mapping
  LACMSA-2000-EMP
  Employment-2000-LACMSA-TAZ-SIC))

    (c) Bookkeeping
```

```
(defrelation LACMSA-2000-EMP
  ((?county USCounty) (?jobs Number)
    (?p Product-SIC-4-1999) (?taz TAZ)) :<=>
 (exists (?o)
   (and (Employment-2000-LACMSA-TAZ-SIC ?o)
     (hasProduct ?o ?p) (hasValue ?o ?jobs)
     (hasGeo ?o ?taz) (geoPartOf ?taz ?county))))

(defconcept Emp-2000-LACMSA-TAZ-SIC (?o) :<=>
 (exists (?p ?taz ?jobs)
   (and (Measurement ?o)
     (hasProduct ?o ?p) (Product-SIC-4-1999 ?p)
     (hasGeo ?o ?taz) (TAZ ?taz)
     (geoPartOf ?taz g-LACMSA)
     (hasTimeInterval ?o 2000)
     (hasUnit ?o u-NumberOfJobs)
     (hasValue ?o ?jobs))))

   (b) Factored Definition
```

Fig. 3. Data Description for the LACMSA-2000-EMP source table

equivalent, factoring has knowledge engineering and reasoning advantages. From the knowledge engineering perspective, the concept definitions are more modular and can be reused in the definitions of many data relations. From the reasoning perspective, it is more efficient to compute subsumption between concepts than between relations (although PowerLoom can compute both). For example, two relations of the same arity may represent identical semantic information, but have their arguments in a different order. In order to prove semantic equivalence, our system would need to explore the permutations of the arguments, which may be expensive. Using the associated concept descriptions, the system can easily prove concept equivalence first and worry about the arguments in a second phase (cf. Section 3.3). As we describe later, storing the mapping between the relational description and distinguished concepts in its body (cf. Figure 3(c)), contributes to more efficient reasoning during planning.

In our representation we can describe both complete and incomplete data sources, corresponding respectively, to the *closed-world* and *open-world* semantics in data integration systems [8]. 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. For example, the relation definition in Figure 3 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. For example, defining the table `LACMSA-2000-EMP` as incomplete would mean that there could be tuples missing for some TAZ or some product. In our transportation modeling domain complete descriptions are customary. In this paper

we focus primarily on reasoning with complete descriptions, but our approach can handle both complete and incomplete data descriptions (see Section 3.1).

3 Automatically Composing Data Processing Workflows

Argos automatically generates a workflow that answers user data requests by composing available sources and data processing operations. We assume that sources and operations are outside our control. For example, operations may be web services or functions from third-party libraries. Similarly, sources can be databases or other web services. We use the terms operation and service interchangeably. A source is an operation that does not require inputs.

Automatically generating a workflow presents two main challenges. First, services may use different schemas. Second, the data produced by a service may not be input directly into another, but may need some kind of transformation (*shim*). Argos addresses both these challenges. First, we resolve the semantic heterogeneity by describing the data in a common ontology. We associate an expressive Local-as-View [8] description with each data relation consumed or produced by a service. 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 inputs and outputs of services.

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 [17] and Sage [12]. 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 4. 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. The Argos planner considers three types of plan refinements: domain-dependent, generic domain-dependent and domain-independent.

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 a *relational* subsumption or equivalence test. If the test succeeds, the planner establishes a *data link* from the output relation of one service to the input relation of the other service. This mechanism is analogous to the establishment of a causal link in plan-space planning [17] where the effect of an operator produces a precondition of another operator. However, instead of using simple unification to match

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

add-domain-service(P, A, [g, s1])
  ∀s2 ∈ services(Domain) such that
      ∃g2 ∈ outputs(s2) such that
        equivalent(g2, g) (or subset(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

plan-refinements(P, A, [g, s1])
  return union(
    reuse-service(P, A, [g, s1])
    add-domain-service(P, A, [g, s1])
    ;; generic domain-dependent ops
    add-product-conversion(P, A, [g, s1])
    add-unit-conversion(P, A, [g, s1])
    ;; relational domain-independent ops
    add-selection(P, A, [g, s1])
    add-projection(P, A, [g, s1])
    add-join(P, A, [g, s1])
    add-union(P, A, [g, s1]) )
```

Fig. 4. Argos Planning Algorithm

a precondition with an effect predicate, our planner uses relational subsumption (or equivalence), because our inputs and outputs are universally quantified formulas that represent data relations. If an operation allows incomplete inputs, it suffices to prove that the description for the output of the provider operation is contained in the input of the consumer operation.

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.

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. Our planner is limited to generating workflows that are directed acyclic graphs.

3.2 Domain-Dependent Services

Domain-dependent services are described by an input/output signature with predefined data relations. For example, the relation `LACMSA-2000-EMP` of Figure 3 describes the output of a service. A service can have multiple input and output data relations. Each service description is bound to a service implementation that can be a web service or a local program.

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. Since our descriptions are significantly more expressive that those used in databases, proving the applicability of selection involves reasoning with background information in the ontology, not just the data definitions.

add-selection($P, A, < wantR, s_c >$)
  $\forall$ $haveR \in$ outputs(s_p) $\wedge$ $s_p \in$ services($Domain$) $\wedge$
      relation-concept-mapping($haveR, wantSC$) $\wedge$ superconcept($wantSC, wantC$) $\wedge$
      relation-concept-mapping($wantR, wantC$)
    [$compatible, mapping, selections$] := compatible-signatures($haveR, wantR$)
    if $compatible$ then
      $argsSelR$:= apply($mapping$,args($haveR$))
      $selR$:= (kappa ($argsSelR$) (and $haveR$ $selections$))
      if equivalent($selR, wantR$) then
        s_σ := create-select-service($haveR, selections, selR$)
        P' := add-data-links(add-step(P, s_σ), $< s_\sigma, wantR, s_c >$)
        A' := A $- \{< wantR, s_c >\} \cup \{< haveR, s_\sigma >\}$
        push $< P', A' >$ into refinements
  return refinements

compatible-signatures($haveR, wantR, wantC$)
  Check that args($haveR$) and args($wantR$) satisfy conditions:
    1. $\forall$ $wa \in$ args($wantR$) $\exists$ $ha \in$ args($haveR$) equivalent(type(wa),type(ha))
      push $[wa \mapsto ha]$ into $mapping$
    2. $\forall$ $a \in$ args($haveR$) $\wedge$ $a \notin mapping$
        $\exists K \in$ definition($wantC$) $\wedge$ individual(K) $\wedge$ equivalent(type(K), type(a))
      push ($= a$ K) into $selections$
  return [(and 1 2), $mapping, selections$]

Fig. 5. Selection Plan Refinement

The algorithm for the `add-selection` plan refinement appears in Figure 5. The process is better described using an 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 6(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 3. 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. Figure2), `LA-2000-EMP` is a subset of `LACMSA-2000-EMP`, and thus the planner can just select from it to achieve the goal.

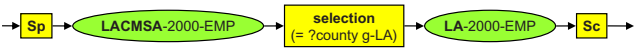
The relations that are candidate for a selection operation must be a superset of the goal relation. The `add-selection` refinement first finds each candidate relation $haveR$, which is the output of an existing service `Sc`, and whose associated concept $wantSC$ is a superconcept of the associated concept $wantC$ of the goal relation $wantR$. In our example, $wantR$ is `LA-2000-EMP`, $wantC$ is `Employment--2000-LA-TAZ-SIC`, $wantSC$ is `Employment-2000-LACMSA-TAZ-SIC`, and $haveR$ is `LACMSA-2000-EMP`. The `compatible-signatures` procedure tests whether the arguments of the candidate and the goal relations are compatible by testing two conditions. The first condition finds a mapping between arguments of the same

```
(defrelation LA-2000-EMP ((?taz TAZ) (?p Product-SIC-4-1999) (?jobs Number)) :<=>
  (exists (?o) (and (Employment-2000-LA-TAZ-SIC ?o)
    (hasProduct ?o ?p) (hasGeo ?o ?taz) (TAZ ?taz) (hasValue ?o ?jobs))))

(defconcept Employment-2000-LA-TAZ-SIC (?o) :<=> (exists (?p ?taz ?jobs)
  (and (Measurement ?o) (hasGeo ?o ?taz) (TAZ ?taz) (geoPartOf ?taz g-LA)
       (hasProduct ?o ?p) (Product ?p) (ofClassification ?p SIC-4-1999)
       (hasUnit ?o u-NumberOfJobs) (hasTimeInterval ?o 2000) (hasValue ?o ?jobs))))

(relation-concept-mapping LA-2000-EMP Employment-2000-LA-TAZ-SIC)
```

(a) Input Required by Service Sc

```
(kappa ((?taz TAZ) (?p Product-SIC-4-1999) (?jobs Number))
  (and (LACMSA-2000-EMP ?c ?jobs ?p ?taz) (= ?c g-LA)))
```

(b) Output of Selection Operator (which is equivalent to input required by Service Sc)



(c) Plan with Relational Selection Adaptor

Fig. 6. Domain-Independent Adaptor: Selection

type. In our example, the arguments types of `LA-2000-EMP` map to the fourth (`TAZ`), third (`Product-SIC-4-1999`), and second (`Number`) arguments of `LACMSA-2000-EMP`. The second condition tests whether there are constants/individuals in the definition of the concept *wantC* associated with the goal relation *wantR* of the same type as the unmapped arguments of the candidate relation *haveR*. In our example, the unmapped argument is `USCounty` and the definition of `Employment--2000-LA-TAZ-SIC` contains the individual `g-LA` which is an instance of `USCounty`. Finally, the `add-selection` refinement checks whether the anonymous[3] relation definition *selR*, the conjunction of relation *haveR* and found *selections* (shown in Figure 6(b)), is equivalent to the goal relation *wantR* (`LA-2000-EMP`). If so, the refinement succeeds and adds a selection service to the plan, as shown in Figure 6(c).

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 (union) plan refinement searches for outputs relations whose conjunction (disjunction) is equivalent to/contained in the desired input relation.

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 (cf.

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

Section 3.2), and the domain-independent operators that are applicable to any dataset description regardless of the domain (cf. Section 3.3).

Product conversion is a prime example of a generic, but domain-dependent operator. Economic data is reported in a variety of classifications, such as SIC or NAICS. Thus, when integrating data from different sources, the system must translate between classifications. Instead of defining a host of domain-specific operators, we added a generic product conversion refinement to the Argos library.

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 a conversion table from C1 to C2 and the desired data in classification C1. 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 7 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 7(a) (for simplicity, we show the direct not the factored representation). First, the algorithm finds sources for conversion tables into 6-digit NAICS by issuing the query shown in Figure 7(b) against the ontology. Assume that it finds a source `Sp` that produces the relation `SIC2NAICS` of Figure 7(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 resulting plan is shown in Figure 7(d). Our system also includes a unit conversion operator that works in a similar fashion.

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 conversion 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 6. Query `Q17` asks for the total (intra- plus inter-regional) demand for all 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

```
(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)
   (hasProduct ?o ?product) (Product-NAICS-6-2002 ?product)
   (hasUnit ?o u-NumberOfJobs) (hasTimeInterval ?o year2000) (hasValue ?o ?jobs))))
```

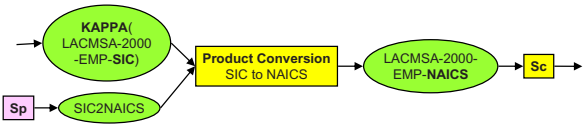
(a) Desired Data Relation

```
(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) )))
```

(c) Product Conversion Relation



(d) Plan with Product Conversion Adaptor

Fig. 7. Generic Domain-Dependent Adaptor: Product Conversion

response to `Q18`, Argos generates the workflow of Figure 1(b). Figure 1(c) shows the results of query `Q18` displayed in ArcGIS.

We tested the workflow planning algorithm with ontologies `A` and `EA`. We report the number of services 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. Consider queries `Q17` and `Q18` that generate the largest workflows. 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.[4]

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).

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

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.

Table 1. Experimental Results

Query	Services		Data Links		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

5 Related Work

Our planner is inspired by the representation of information gathering actions of the Sage planner [12] of the SIMS [4] 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 6 could not be performed in SIMS. Second, Argos uses subsumption to established data links, while SIMS relied on syntactic matching to establish causal links. On the other hand, SIMS included cost optimization techniques [3], like pushing selections, that we have not (yet) incorporated in our planner.

The TAMBIS system [5] integrated 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 goals of

TAMBIS. However, our local-as-view relational descriptions 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.

With respect to the shim classification of [10], our adaptors fall mostly in the the semantic translator category (e.g., product conversion). However, the relational-algebra adaptors (Section 3.3) constitute a new class of adaptors.

Szomszor et al. [23] present an approach to syntactic mediation in web service workflows that automatically inserts type adaptors that translate XML data. In contrast to their work, we do not focus on specifying the implementation of a translation operator, only on describing semantically the input/output signature. Incidentally, since we deal with relational data, sometimes translations can be implemented as SQL queries (with aggregation), as is the case in the product conversion adaptor.

Our domain modeling builds on the idea of faceted representations (e.g. [19]). 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 (EDC) project [1]. However, we have a more refined domain ontology. More critically, since EDC was based on SIMS, its ability to describe data was limited.

Research on mixed-initiative composition of scientific workflows [11,22] also leverages semantic descriptions. However, Argos uses a more expressive description language and focuses on automatic composition.

There has been research on automatic web service composition within the AI planning community: executable grid workflows [6], HTN-based composition [20], composition using Golog-procedures [15]. Some of these systems are more expressive than Argos since they model state change. However, Argos provides more expressive data representation and manipulation.

6 Discussion and Future Work

We have presented a logic-based planning approach to automatically compose data processing workflows. We describe data and services using a formal ontology. The planner uses the ontology and relational subsumption, provided by the PowerLoom reasoner, to construct workflows that answer user data requests.

We are currently working on using query reformulation to link services' inputs and outputs. Inputs can be seen as queries and outputs as views (in the database sense), so standard techniques for answering queries using views [8] can be applied. We are exploring more restricted languages like conjunctive queries, as well as developing a query reformulation algorithm for the PowerLoom first-order logic based on abductive reasoning.

Finally, we plan to incorporate cost optimization knowledge into the planner, so that it generates more efficient workflows, for example, by pushing selections closer to the sources and other optimizations.

References

1. Ambite, J.L., Arens, Y., Hovy, E., Philpot, A., Gravano, L., Hatzivassiloglou, V., Klavans, J.: Simplifying data access: The energy data collection (EDC) project. IEEE Computer, Special Issue on Digital Government 34(2) (February 2001)
2. Ambite, J.L., Kapoor, D.: Automatic generation of data processing workflows for transportation modeling. In Proceedings of the 8th Annual International Conference on Digital Government Research (dg.o2007), Philadelphia, PA, USA (2007)
3. Ambite, J.L., Knoblock, C.A.: Flexible and scalable cost-based query planning in mediators: A transformational approach. Artificial Intelligence 118(1-2), 115–161 (2000)
4. Arens, Y., Knoblock, C.A., Shen, W.-M.: Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration 6(2/3), 99–130 (1996)
5. Baker, P.G., Brass, A., Bechhofer, S., Goble, C., Paton, N., Stevens, R.: TAMBIS: Transparent access to multiple bioinformatics information sources. In Procedings of the 6th International Conference on Intelligent Systems for Molecular Biology, Montreal, Canada (1998)
6. Blythe, J., Deelman, E., Gil, Y.: Automatically composed workflows for grid environments. IEEE Intelligent Systems, 16–23 (July/August 2004)
7. Giuliano, G., Gordon, P., Pan, Q., Park, J., Wang, L.: Estimating freight flows for metropolitan highway networks using secondary data sources. In Proceedings of the Transportation Research Board Commodity Flow Survey Conference, Transportation Research Circular, E-C088, 154–158 (July 2005)
8. Halevy, A.Y.: Answering queries using views: A survey. The VLDB Journal 10(4), 270–294 (2001)
9. Horrocks, I., Sattler, U., Tessaris, S., Tobies, S.: How to decide query containment under constraints using a description logic. In: Parigot, M., Voronkov, A. (eds.) LPAR 2000. LNCS (LNAI), vol. 1955, Springer, Heidelberg (2000)
10. Hull, D., Stevens, R., Lord, P., Wroe, C., Goble, C.: Treating shimantic web syndrome with ontologies. In Proceedings of the 1st AKT workshop on Semantic Web Services (AKT-SWS04), Milton Keynes, UK (2004)
11. Kim, J., Spraragen, M., Gil, Y.: An intelligent assistant for interactive workflow composition. In Proceedings of the International Conference on Intelligent User Interfaces, Madeira, Portugal (2004)
12. Knoblock, C.A.: Building a planner for information gathering: A report from the trenches. In Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems, Edinburgh, Scotland (1996)
13. MacGregor, R.: A deductive pattern matcher. In Proceedings of the 7th National Conference on Artificial Intelligence, Saint Paul, MN (1988)
14. MacGregor, R.: A description classifier for the predicate calculus.In Proceedings of the Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA (1994)
15. McIlraith, S., Son, T.C.: Adapting Golog for composition of semantic web services. In Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (2002)
16. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In Proceedings of the International Semantic Web Conference, Sardinia, Italy (2002)

17. Penberthy, J.S., Weld, D.S.: UCPOP: A sound, complete, partial order planner for ADL. In Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA (1992)
18. PowerLoom. The PowerLoom knowledge representation & reasoning system (2003), `www.isi.edu/isd/LOOM/PowerLoom`
19. Pratt, W., Hearst, M.A., Fagan, L.M.: A knowledge-based approach to organizing retrieved documents. In Proceedings of the 16th National Conference on Artificial intelligence, Menlo Park, CA, USA (1999)
20. Sirin, E., Parsia, B., Wu, D., Hendler, J., Nau, D.: HTN planning for web service composition using SHOP2. Web Semantics 1(4), 377–396 (2004)
21. Sivashanmugam, K., Verma, K., Sheth, A., Miller, J.: Adding semantics to web services standards. In Proceedings of the 1st International Conference on Web Services. Las Vegas, NV (2003)
22. Stevens, R.D., Robinson, A.J., Goble, C.A.: myGrid: personalised bioinformatics on the information grid. Bioinformatics 19(1), 302–304 (2003)
23. Szomszor, M., Payne, T., Moreau, L.: Automated syntactic medation for web service integration.In Proceedings of the International Conference on Web Services, Chicago, IL (2006)