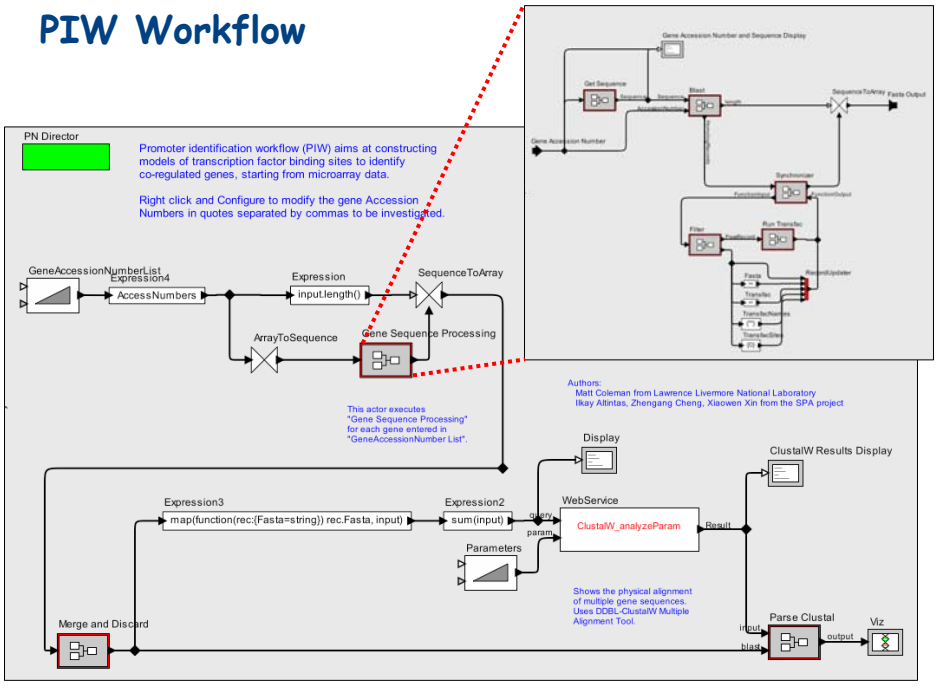
Where we are so far …

- Intro to Data Integration (Datalog, mediators, …)
  - more to come (your projects!): schema matching, simple query rewriting
- Intro to Knowledge Representation & Ontologies
  - description logic, first-order logic, …
  - more to come: FCA, biological pathways & ontologies
- Intro to Scientific Workflows and Kepler
  - more to come: lectures and your projects…
- Today: Linking Scientific Workflows (Process Integration) and "Semantics"
  - support for workflow modeling and design
  - data discovery
  - service/actor discovery
  - data ⇔ actor binding
  - actor ⇔ actor binding

PIW Workflow

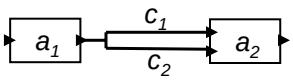


A Simplified Scientific Workflow Model

- A SWF consists of:
  - a set $A = \{a_1, a_2, \ldots, a_n\}$ of *actors*

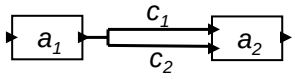
    [[NOTE: we often but not always use "ptolemaic" (ie Ptolemy II terminology); other terms are *steps, components, services, processes, ...* we will say more about some of these]]
    - each actor *a* can have a number of (named) *ports* $\{p_1, \ldots, p_k\}$
    - a port is either an *input port* or an *output port*
  - a set $C = \{c_1, c_2, \ldots, c_m\}$ of *channels*, i.e., directed edges linking outputs ports to input ports

    [[NOTE: more will be said about
    - Composite actors/subworkflows: *hierarchical models/workflows*
    - How these actors are executed: *models of computation* and *directors*]]



A Simplified Scientific Workflow Model

- We can capture the topological structure of a SWF in different forms; simple ones are as relations channel/4 or even channel/2:
  - channel(from_actor, out_port, to_actor, in_port)
    - channel(a_1,p_1, a_2, p_1).
    - channel(a_1,p_1, a_2, p_2).
  - channel(from_actor_out_port, to_actor_in_port)
    - channel($a_1\_p_1$, $a_2\_p_1$).
    - channel($a_1\_p_1$, $a_2\_p_2$).
    - … plus a table associating ports with actors



Port Annotations & Types

- Port Name
  - for identification purposes
- Port I/O type
  - to say whether tokens flow in or out of the port
- Structural type
  - to describe the data/object structure of the port
  - … can be captured via
    - a programming language/data type
      - array(0..n) of record of (a:int, b:float)
    - an XML Schema (…)
    - relational schema
      - employee_record(SSN, Name, DeptNo)
- Storage type
  - to describe details of the physical representation (int16, …)
- **Semantic type**
  - to describe the "semantics" of the tokens being transferred on channels
    - a concept name, concept expressions, … or a more general constraint

Semantic Types and Constraints

- The goal of the semantic type is to provide a link between the structural type and concept expressions from an ontology
- In its general form, a semantic type is given as a *semantic annotation* over two schemas:
  - the (conventional) *structural schema S*
  - the *semantic schema* (ontology) *O*
- A semantic annotation is denoted as a logic formula (constraint) α involving symbols from the [structural] schema *S* and the ontology [schema] *O*
- Here we focus on α having the form α : *S* → *O*
  ➔ "virtually populate" ontology structure *O* with instances of *S*

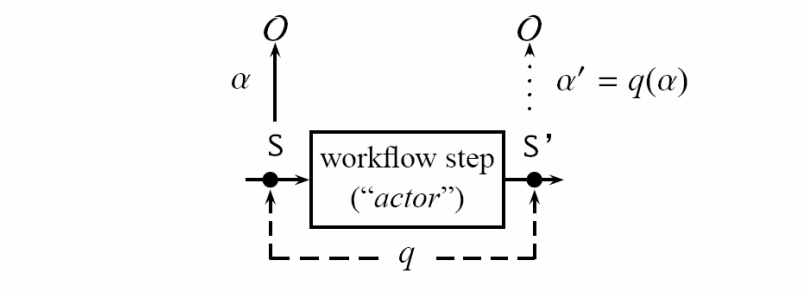
A Semantic Annotation α : *S* → *O*

(1)	x : biom	==>	x : OBSERVATION
(2)	x : biom[yr = y]	==>	x [TEMPORALCONTEXT = y : YEAR]
(3)	x : biom[seas = y], y = 'W'	==>	x [TEMPORALCONTEXT = y :WINTER]
(4)	x : biom[seas = y], y = 'S'	==>	x [TEMPORALCONTEXT = y : SPRING]
(5)	x : biom[seas = y], y = 'F'	==>	x [TEMPORALCONTEXT = y : FALL]
(6)	x : biom[plt = y]	==>	x [SPATIALCONTEXT = y : PLOT]
(7)	x : biom[qd = y]	==>	x [SPATIALCONTEXT = y : QUADRAT]
(8)	x : biom[spp = y]	==>	x [ITEM = y : SPECIES]
(9)	x : biom[bm = y]	==>	x [PROPERTY = y : BIOMASSGSM]

Schema elements/ Structural type S

Ontology / Semantic type O

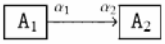
Propagating Semantic Annotations



- **Given**:
  - structural schemas S (input) and S' (output), and an ontology O
  - a *semantic annotation* α: S → O
  - a *query annotation q:* S→ S'
- **Problem**: compute α'

Applications

Example 1 (Ontology Augmentation) Consider an actor A_2 having as input species richness data. The developer of A_2 might have provided a semantic annotation α_2 for A_2's input, explicitely stating that it was designed to "consume" (species) RICHNESS[3] data. Now assume a workflow designer connects the output of another actor A_1 to the input of A_2, and assume also that for the output of A_1 a semantic annotation α_1 has been derived via propagation, indicating that A_1's output is of type sum(OCCURRENCE), i.e., the arithmetic sum (resulting from a corresponding aggregate operation) of OCCURRENCE data. In order for the link



to be semantically type correct, we must have that α_1 is "compatible" with α_2, i.e., sum(OCCURRENCE) $\sqsubseteq$ RICHNESS. If we choose to run our propagation system in "automatic mode", it will augment the given ontology O with this additional axiom. Conversely, the system can be run in "interactive mode", asking the user to determine whether the inferred axiom is correct and can be included in O, or whether there is something wrong with the connection. □

- WF design time:
  - Actor⇔ Actor connections
- Data binding time:
  - Actor⇔ Data connections ("data binding")
- WF runtime:
  - "semantic tagging" of derived data products

B. Ludaescher, ECS289F-W05, Topics in Scientific Data Management

Ontology *O*

(1)	OBSERVATION	⊑	∀ PROPERTY.OBSPROPERTY ⊓ =1 PROPERTY.OBSPROPERTY ⊓ ∀ ITEM.OBSERVABLEITEM ⊓ =1 ITEM.OBSERVABLEITEM ⊓ ∀ SPATIALCONTEXT.SPATIALCONTEXT ⊓ ∃ SPATIALCONTEXT.SPATIALCONTEXT ⊓ ∀ TEMPORALCONTEXT.TEMPORALCONTEXT ⊓ ∃ TEMPORALCONTEXT.TEMPORALCONTEXT
(2)	OCCURRENCE	⊑	OBSPROPERTY
(3)	ABUNDANCE	⊑	OCCURRENCE
(4)	BIOMASS	⊑	ABUNDANCE
(5)	BIOMASSGSM	≡	BIOMASS ⊓ GRAMSSQUAREMETER
(6)	BIOMASSPPH	≡	BIOMASS ⊓ POUNDSPERHECTARE
(7)	RICHNESS	⊑	ABUNDANCE
(8)	PRODUCTIVITY	⊑	BIOMASS
(9)	PRODUCTIVITYGSM	⊑	BIOMASSGSM ⊓ GRAMSSQUAREMETER
(10)	PRODUCTIVITYPPH	⊑	BIOMASSPPH ⊓ POUNDSPERHECTARE
(11)	SPECIES	⊑	OBSERVABLEITEM
(12)	PLOT	⊑	SPATIALCONTEXT
(13)	QUADRAT	⊑	SPATIALCONTEXT
(14)	YEAR	⊑	TEMPORALCONTEXT
(15)	SEASON	⊑	TEMPORALCONTEXT
(16)	WINTER	⊑	SEASON
(17)	SPRING	⊑	SEASON
(18)	SUMMER	⊑	SEASON
(19)	UNITTYPE	⊑	∀FORUNIT.UNIT ⊓ ≤1 FORUNIT
(20)	GRAMSSQUAREMETER	≡	UNITTYPE ⊓ ∀FORUNIT.{GSMUNIT}
(21)	POUNDSPERHECTARE	≡	UNITTYPE ⊓ ∀FORUNIT.{PPHUNIT}
(22)	SIUNIT	⊑	UNIT ⊓ ...

Figure 3. A simplified measurement ontology with ecological concepts.

B.

Scientific Workflow w/ Query Annotations *q*

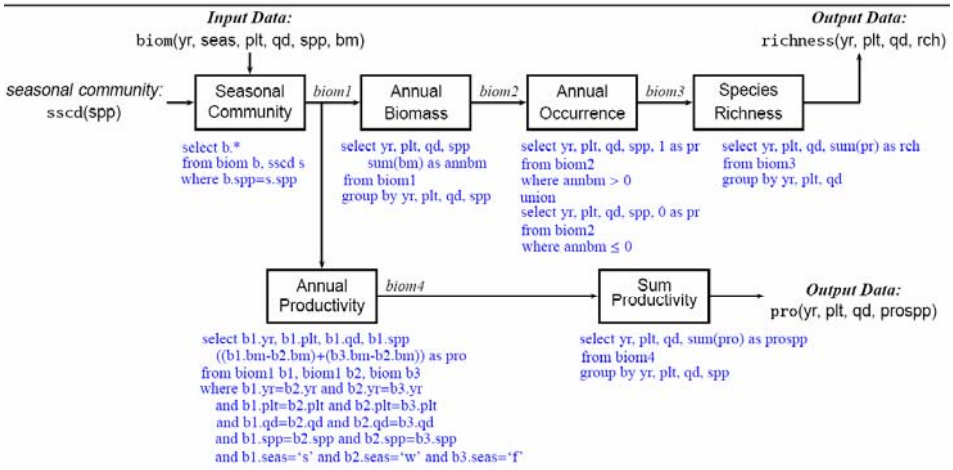


Figure 2. A scientific workflow with *query annotations q* (in SQL) for computing species richness and productivity [14] (see Figure 4 for semantic annotations α).

Annotation Constraint α : *S* → *O*

$\alpha = \forall \mathbf{x}\ (\alpha_s(\mathbf{x}) \wedge \alpha_c(\mathbf{x})) \rightarrow \exists \mathbf{y}\ \alpha_o(\mathbf{z})$ % $z = x \cup y$

- α_s links the variables **x** to schema elements of *S*
- α_c is conjunction of comparisons over **x** and constants
- α_o "populates" the ontology structure O

X : biom[seas=S], S = 'w' → X : observation[temporalContext = S : WinterSeason]

$\alpha_s(\mathbf{x})$ $\alpha_c(\mathbf{x})$ $\alpha_o(\mathbf{z})$

Semantic Annotations

We consider query annotations q and semantic annotations α of the form:

- $q = \forall \bar{u} \exists \bar{v}\ P_{S'}(\bar{u}) :\!- \varphi_S(\bar{u}, \bar{v})$
- $\alpha = \forall \bar{x} \exists \bar{y}\ \alpha_S(\bar{x}) \rightarrow \alpha_O(\bar{x}, \bar{y})$

Here, $P_{S'}$ is a logic atom over the output schema S' and φ_S is a query over the input schema(s) S. Similarly, in a semantic annotation α we relate instances of a schema S with those of an ontology O via subformulas α_S and α_O.

The basic idea of computing $\alpha' = q(\alpha)$ is as follows: We would like to relate instances of the output schema S' with instances of O. Assume we can find instances of φ_S (in q) that imply certain instances of α_S (in α) to be true. For these we then have established the desired relation between S' and O; we can denote this abstractly as $S' \overset{q}{\rightsquigarrow} S \overset{\alpha}{\rightsquigarrow} O$.

More precisely, consider q as a constraint of the form

$$q(\bar{u}) = P_{S'} \rightarrow \underbrace{Q_1 \wedge \cdots \wedge Q_n \wedge \psi}_{\varphi_S}$$

and α of the form

$$\alpha(\bar{x}) = \underbrace{A_1 \wedge \cdots \wedge A_k}_{\alpha_S} \rightarrow \alpha_O$$

Here, $P_{S'}$ is a logic atom over the output schema S'. Similarly, the Q_i and A_j are logic atoms over the input schema(s) S; ψ and α_O are first-order formulas. Note that we assume that all $\exists$-quantified variables ($\bar{v}$ and $\bar{y}$ above) have been eliminated through Skolemization[6], so that q and α can be seen as (implicitely) $\forall$-quantified formulas with variables $\bar{u}$ and $\bar{x}$, respectively. In particular, we can assume that the variables $\bar{u}$ in $q(\bar{u})$ are disjoint from the variables $\bar{x}$ in $\alpha(\bar{x})$.

Observe that q can be written as a conjunction $q_1 \wedge \cdots \wedge q_n \wedge q_\psi$ with $q_i = P_{S'} \rightarrow Q_i$, and $q_\psi = P_{S'} \rightarrow \psi$. Now assume that there is a substitution σ that unifies some atom Q_{i_0} and some A_{j_0}, i.e., $Q^\sigma_{i_0} = A^\sigma_{j_0}$.[7] Then we can infer from $q_{i_0} = P_{S'} \rightarrow Q_{i_0}$ and α a new annotation constraint α'_{i_0} of the form:

$$\alpha'_{i_0} = P^\sigma_{S'} \wedge (\alpha_S \setminus A_{j_0})^\sigma \rightarrow \alpha^\sigma_O$$

where $(\alpha_S \setminus A_{j_0})$ is the conjunction $A_1 \wedge \cdots \wedge A_k$ with A_{j_0} removed. It is easy to show that the annotation α'_{i_0} is implied by q_i and α. In this way, by successively "resolving away" atoms A_j from α_S with matching atoms Q_i from φ_S, we can obtain new semantic annotations α' that relate elements of the output schema S' to those in the ontology O.

Example (Biodiversity Workflow)

Example 6 Consider the following query annotation expressed as a skolemized first-order formula for the *Seasonal Community* actor in Figure 2

(1) a:`biom1`[`yr`=r, `seas`=s, `plt`=t, `qd`=q, `spp`=p, `bm`=b] →
a:`biom`[`yr`=r, `seas`=s, `plt`=t, `qd`=q, `spp`=p, `bm`=b]
$\wedge$ $f(a)$:`sscd`[`spp`=$g(a)$] $\wedge$ $g(a)$=p

and semantic annotation (9) in Figure 4, expressed as the first-order formula

(2) x:`biom`[`bm`=y] → x[PROPERTY=y:BIOMASS]

We can resolve (1) with (2) using the substitution $\sigma = \{x \mapsto a, y \mapsto b\}$[8], which results in the formula

(3) a:`biom1`[`yr`=r, `seas`=s, `plt`=t, `qd`=q, `spp`=p, `bm`=b] →
a[ITEM=b:BIOMASS] $\wedge$ $f(a)$:`sscd`[`spp`=$g(a)$] $\wedge$ $g(a)$=p

Observe that we now have `bm` values for the output schema `biom1` semantically annotated as BIOMASS instances, linked through the PROPERTY role to corresponding tuples. We also get some additional information, which can be viewed as constraints over the input schema. These additional constraints can be ignored, bringing (3) into our standard form for semantic annotations, which results in

(4) a:`biom1`[`yr`=r, `seas`=s, `plt`=t, `qd`=q, `spp`=p, `bm`=b] →
a[PROPERTY=b:BIOMASS]

We can further simplify (4) by dropping the attribute variables not used elsewhere in the annotation, giving

(5) a:`biom1`[`bm`=b] → a[PROPERTY=p:BIOMASS] □

Family Example

Conjunctive Queries. We start with a semantic annotation with the signature $\alpha : S \to O$

- $\mathtt{p}(c, p) \to \textsc{Child}(c) \wedge \textsc{Parent}(p)$ (α)

... and a query annotation with the signature $q : \ S' :\!- S$

- $\mathtt{gc}(y, x) :\!- \mathtt{p}(x, z), \mathtt{p}(z, y)$ (q)

To propatate the semantic type α through q, we need the "backward" (i.e., left-to-right) direction of q[4]:

- $\mathtt{gc}(y, x) \to \exists z.\mathtt{p}(x, z) \wedge \mathtt{p}(z, y)$ (q_b)

Before we chase q_b with α, we eliminate the $\exists$-quantifier via Skolemization:

- $\mathtt{gc}(y, x) \to \mathtt{p}(x, f_z(x, y)) \wedge \mathtt{p}(f_z(x, y), y)$ (q_b')

We can resolve (α) and (q_b') via the substitution $\sigma_1 = \{c \mapsto x, \ p \mapsto f_z(x, y)\}$ resulting in:

- $\mathtt{gc}(y, x) \to \textsc{Child}(x) \wedge \textsc{Parent}(f_z(x, y)) \wedge \mathtt{p}(f_z(x, y), y)$ $(res_{\sigma_1}(\alpha, q_b'))$

We can apply α again, now with $\sigma_2 = \{c \mapsto f_z(x, y), \ p \mapsto y\}$ resulting in:

- $\mathtt{gc}(y, x) \to \textsc{Child}(x) \wedge \textsc{Parent}(f_z(x, y)) \wedge \textsc{Child}(f_z(x, y)) \wedge \textsc{Parent}(y)$ $(res_{\sigma_2}(\alpha, res_{\sigma_1}(\alpha, q_b')))$

Observe how this result is in the desired form $\alpha' : S' \to O$. We now know that x and y are instances of Child and Parent, respectively. Indeed x is the grandchild, y is the grandparent. We also get some additional information (that we might choose to ignore), i.e., that there is some instance $f_z(x, y)$ who is both a Child and a Parent (as it turns out, this is the intermediate parent, establishing the link between the grandchild and grandparent).