Hybrid-Type Extensions for Actor-Oriented Modeling
(a.k.a. Semantic Data-types for Kepler)

Shawn Bowers & Bertram Ludäscher
University of California, Davis
Genome Center & CS Dept.
May, 2005

Outline

1. Hybrid Types
2. Hybrid Types and Scientific Workflow Design
3. Super Rapid Prototyping: The “Sparrow Family of Languages”
4. Next Steps: Adding a Hybrid-Type System to Kepler
Hybrid Types

Hybrid Types: Superimposing Semantics

Separation of Concerns:

**Conventional** Data Modeling (*Structural Data Types*)
- E.g., XML Schema / DTD, etc.

**Conceptual** Data Modeling (*Semantic Types*)
- Drawn from ontologies (expressed in Description Logic)
- Capturing domain knowledge (e.g., biodiversity, ecology)

Explicit (external) **linkages** (*Semantic Annotations*)
- Simple links (one concept per item)
- Links expressed as constraints (logical mappings)
Hybrid Types: Superimposing Semantics

Datatypes

T1 := table
    site plot spp bm
    string string string double
    a relational table of measurements

T2 := list
    spp
    string
    a list of strings

Semtypes

SpeciesBiomass ⊑ Measurement ⊓ ∃item.Species ⊓ ∃prop.Biomass ⊓ ∃loc.Location

"Species biomass is a measure of the amount of biomass of a particular species within a location."

SpeciesCommBiomass ⊑ SpeciesBiomass ⊓ ∃loc.Community

"Species community biomass is a species biomass within a community."

table/X:meas => X:SpeciesCommBiomass

each table/meas instance is a measurement
SpeciesBiomass ⊑ Measurement □ item.Species □ prop.Biomass □ loc.Location

"Species biomass is a measure of the amount of biomass of a particular species within a location."

SpeciesCommBiomass ⊑ SpeciesBiomass □ loc.Community

"Species community biomass is a species biomass within a community."

T1 := table
    Community
        meas
    site
table/X:meas, X/Y:site, X/Z:plot ⇒
X:SpeciesCommBiomass, C=f(Y,Z),
(X,C):loc, C:Community,
**Hybrid Types: Superimposing Semantics**

SpeciesBiomass $\subseteq$ Measurement $\sqcap$ $\exists$Item.Species $\sqcap$ $\exists$prop.Biomass $\sqcap$ $\exists$loc.Location

"Species biomass is a measure of the amount of biomass of a particular species within a location."

SpeciesCommBiomass $\subseteq$ SpeciesBiomass $\sqcap$ $\exists$loc.Community

"Species community biomass is a species biomass within a community."

T1 := table

SpeciesCommBiomass

<table>
<thead>
<tr>
<th>Community</th>
<th>meas</th>
<th>Species</th>
<th>Biomass</th>
</tr>
</thead>
<tbody>
<tr>
<td>site</td>
<td>plot</td>
<td>spp</td>
<td>double</td>
</tr>
</tbody>
</table>

SpeciesBiomass $\subseteq$ Measurement $\sqcap$ $\exists$Item.Species $\sqcap$ $\exists$prop.Biomass $\sqcap$ $\exists$loc.Location

"Species biomass is a measure of the amount of biomass of a particular species within a location."

SpeciesCommBiomass $\subseteq$ SpeciesBiomass $\sqcap$ $\exists$loc.Community

"Species community biomass is a species biomass within a community."

**Hybrid Types: Superimposing Semantics**

**Searching**
- Concept-based, e.g., “find all datasets containing biomass measurements”

**Merging/Integrating**
- Combining heterogeneous sources based on annotations
- Concatenate, Union (merge), Join, etc.

**Transforming**
- Construct mappings from schema S1 to S2 based on annotations

**Semantic Propagation**
- “Pushing” semantic annotations through transformations/queries
Semantic Annotation Propagation

Capture I/O constraints
- Similar to unit type constraints
- Can enable automated metadata creation (annotation "propagation")
- Can help refine ontologies and existing annotations

![Diagram](image)

Figure 1. A workflow step whose semantic annotation \( \alpha \) has been propagated via \( q \) to a new semantic annotation \( \alpha' \).

Semantic Annotation Propagation

\[
\alpha \xrightarrow{p} \alpha' = q(\alpha)
\]

Port 1 Annotation

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{table} & \text{meas} & \text{site} & \text{spp} & \text{bm} \\
\hline
\text{string} & \text{double} & \text{string} & \text{string} & \text{double} \\
\hline
\end{array}
\]

T1 := table

\[
p1::T1 \quad p2::T2 \quad p3::T3
\]

p1.meas(site, plot, spp, bm) :=

p2.spp(spp).

Port 2 “Chased” Annotation

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{table} & \text{obs} & \text{site} & \text{plot} & \text{bm} \\
\hline
\text{string} & \text{string} & \text{string} & \text{double} & \\
\hline
\end{array}
\]

T3 := table

\[
p3.obs(site, plot, spp, bm) :=
\]

p1.meas(site, plot, spp, bm),
p2.spp(spp).

Port 1 Annotation

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{table} & \text{meas} & \text{X/Y:site} & \text{X/Z:plot} & \text{X/U:spp} & \text{X/B:bm} \\
\hline
\text{string} & \text{string} & \text{double} & \text{string} & \text{double} \\
\hline
\end{array}
\]

\[
\begin{align*}
(\text{X,C}:\text{loc}, \text{C:Community}), \\
(\text{X,U}:\text{item}, \text{U:Species}), \\
(\text{X,B}:\text{prop}, \text{B:Biomass}).
\end{align*}
\]

Actor I/O constraint (approx.)

\[
p3.obs(site, plot, spp, bm) :=
p1.meas(site, plot, spp, bm),
p2.spp(spp).
\]

Port 2 “Chased” Annotation

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{table} & \text{obs} & \text{X/Y:site} & \text{X/Z:plot} & \text{X/U:spp} & \text{X/B:bm} \\
\hline
\text{string} & \text{string} & \text{string} & \text{double} & \\
\hline
\end{array}
\]

\[
\begin{align*}
(\text{X,C}:\text{loc}, \text{C:Community}), \\
(\text{X,U}:\text{item}, \text{U:Species}), \\
(\text{X,B}:\text{prop}, \text{B:Biomass}).
\end{align*}
\]
Hybrid Types and Scientific Workflow Design

Workflow Design Primitives

End-to-End Workflow Design and Implementation

- Viewed as a series of primitive “transformations”
- Each takes a WF and produces a new WF
- Can be combined to form design “strategies”
Workflow Design Primitives

Semantic types and Actor Oriented Modeling:

- **Actors and Workflows**: can have semantic types conceptually describing their “function”
- **Ports**: can have semantic types conceptually describing what they consume and produce
- **I/O Constraints**: a general form of constraint between input and output (e.g., like unit constraints) ... approximating the function of an actor

<table>
<thead>
<tr>
<th>Basic Transformations</th>
<th>Starting Workflow</th>
<th>Resulting Workflow</th>
<th>Resulting Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 ): Entity Introduction (actor or data connection)</td>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>( t_2 ): Port Introduction</td>
<td><img src="image4" alt="Diagram" /></td>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>( t_3 ): Datatype Refinement ((s', \pi, t' \geq t))</td>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
<td><img src="image9" alt="Diagram" /></td>
</tr>
<tr>
<td>( t_4 ): Hierarchical Abstraction</td>
<td><img src="image10" alt="Diagram" /></td>
<td><img src="image11" alt="Diagram" /></td>
<td><img src="image12" alt="Diagram" /></td>
</tr>
<tr>
<td>( t_5 ): Hierarchical Refinement</td>
<td><img src="image13" alt="Diagram" /></td>
<td><img src="image14" alt="Diagram" /></td>
<td><img src="image15" alt="Diagram" /></td>
</tr>
<tr>
<td>( t_6 ): Data Connection</td>
<td><img src="image16" alt="Diagram" /></td>
<td><img src="image17" alt="Diagram" /></td>
<td><img src="image18" alt="Diagram" /></td>
</tr>
<tr>
<td>( t_7 ): Director Introduction</td>
<td><img src="image19" alt="Diagram" /></td>
<td><img src="image20" alt="Diagram" /></td>
<td><img src="image21" alt="Diagram" /></td>
</tr>
</tbody>
</table>
## Additional (Planned) Design Primitives for Semantic Types

<table>
<thead>
<tr>
<th>Extended Transformations</th>
<th>Starting Workflow</th>
<th>Resulting Workflow</th>
<th>Resulting Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Actor Semantic Type Refinement ( T \rightarrow T' )</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>2. Port Semantic Type Refinement ( C \rightarrow C', D \rightarrow D' )</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>3. Annotation Constraint Refinement ( \alpha \rightarrow \alpha' )</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>4. I/O Constraint Strengthening ( \psi \rightarrow \phi )</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>5. Data Connection Refinement</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>6. Adapter Insertion</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>7. Actor Replacement</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
<tr>
<td>8. Workflow Combination (Map)</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
<td>![Diagram]</td>
</tr>
</tbody>
</table>

### Adapters for Semantic and Structural Incompatibility

Adapters may:
- be abstract (no impl.)
- be concrete
- bridge a semantic gap
- fix a structural mismatch
- be generated automatically (e.g., Taverna’s “list mismatch”)  
  - be reused components  
    (based on signatures)
Applying the Replacement Primitive

- General replacement doesn’t consider surrounding connections
- Context-sensitive replacement gives more “wiggle room” by “tuning” the actors semtypes based on connections

Workflow Elaboration

Adapter insertion, replacement, and search provide a powerful mechanism for workflow “elaboration”:

1. Given an initial, user specified set of connected “abstract” actors
2. Repeatedly search for replacement “concrete” actors (atomic and composite)
3. At each step, insert adapters when necessary
4. Allow user to select returned workflows to be combined
Super Rapid Prototyping: The Sparrow “Family of Languages”

The “Sparrow Family of Language”

Basic Idea: Have both Machine and Human readable syntax

Sparrow-DL
Description logic

Sparrow-DTD
Datatypes, variant of XML DTDs

Sparrow-Annotate
Configuring concepts; linking datatypes and ontologies

Sparrow-SWF
KSW-Based MoML Metadata

Sparrow-Rule
Fancy stuff, like type constraints (a la unit types), function approximation, and misc. other constraints
The “Sparrow Family of Languages”

`Sparrow-DTD`

**Sparrow-SWF**

**Sparrow-DL**

**Sparrow-Toolkit Operations**

**Sparrow-Toolkit (example) Operations**

- Is `w1` semantically and/or structurally well typed?
- What can be semantically connected to `a3`?
- Insert “abstract” adapter between `a3` and `a4`
- What can replace (e.g., implement) the adapter?
- ...

---

<table>
<thead>
<tr>
<th>p1</th>
<th>p2</th>
<th>p3</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>int</td>
<td>string</td>
</tr>
</tbody>
</table>

---

% a badly formed workflow:

workflow `w1` :=

actors [auth ref `a1`, book ref `a3`, vend ref `a4`],
connection [auth:'p1', book:'p1']
connection (book:'p2', vend:'p1')
signature [vend:'p2']

...
Future Steps: Adding a Hybrid-Type System to Kepler

Concept-based Actor Search
- Implemented as proof-of-concept
  - About to undergo major revision
  - Additional operations slated for next Kepler Release (data search, actor-based port search, etc.)

Biggest Challenges
- Building/searching a repository ...
- Making changes to MoML (see KSW)
- GUI changes
- Ontology management