Semantic Integration Through Invariants

Joseph B. Kopena  
kopena@cme.nist.gov

Michael Grünninger  
gruning@cme.nist.gov

August, 2003
Knowledge and Information Interchange
Presentation Outline

- **Motivating Domain:**
  - Manufacturing, business, and software \textbf{processes}

- **Integration Through Ontology:**
  - \textit{Interlinguas}; the \textbf{Process Specification Language}

- **Application Characterization and Analysis:**
  - \textit{Invariants}; \textit{profiles}; \textit{comparison}
Simplified process model for manufacturing die-cast parts:

Process models may capture:

- **Activity orderings**—sequences, overlaps, alternatives, iterations
- **Resources utilized**—workers, machines, supplies
- **Temporal information**—durations, delay constraints, absolute timestamps
Real-World Applications of Process Modeling

- Military Support
  (streamlining, cost est.)

- Business
  (merging, training)

- Software Services
  (composition, verification)

- Space Operations
  (simulation, analysis)

- Manufacturing
  (design, production)
Integration among cooperating project elements—translation and communication between information and services—is a necessity

- Might not share language, terminology, conventions, software
- Might be geographically, organizationally, even temporally dispersed

Many translators required for integration—worst case $n^2$!

- E.g., with some typical manufacturing process tools/tasks:
Introduce an interlingua—a communication medium between integration targets

- Reduce necessary translators to $n$
- Less work required for integration
- Lower probability of error

E.g., with the same manufacturing process tools/tasks:
Semantically Correct Interchange

Must convey the intended **meaning** of the information—its **semantics**

- **Exchange correct structure and constrain interpretation**
The Process Specification Language

Need to formally characterize the interlingua and targets

- The Process Specification Language (PSL): ontology in first order logic

Capture intended meaning using mathematical structures:

**PSL: Occurrence trees**
(combination of all possible events)

**PSL: Activity trees**
(sequences, iterations, alternatives)

**PSL: Fluents**
(state manipulated by occurrences)
Some invariant properties of geometric shapes:

- Is the shape a polygon with \( n \geq 3 \) sides?
- Is the shape convex?
- Is the symmetry group \( \text{Symm} \) of the shape \( \equiv D_n \), which consists of the rotations \( R_{2k\pi/n} \) for \( k = 0,1,\ldots,n-1 \) and the reflections \( R_{l_1},\ldots,R_{l_n} \) about the lines \( l_1,\ldots,l_n \) connecting the centroid to the vertices and midpoints?

Definition for a class of shapes:

Regular polygons \( \equiv \) convex polygons w/ \( n \geq 3 \) sides and symmetry group \( \equiv D_n \).

<table>
<thead>
<tr>
<th>Shape</th>
<th>( n )</th>
<th>Convexity</th>
<th>Symmetry Group</th>
<th>Regularity</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Shape" /></td>
<td>6</td>
<td>yes</td>
<td>( \text{Symm} \neq D_n )</td>
<td>( \nabla \nabla \nabla ) regular</td>
</tr>
<tr>
<td><img src="image2" alt="Shape" /></td>
<td>8</td>
<td>yes</td>
<td>( \text{Symm} \equiv D_n )</td>
<td>( \nabla \nabla \nabla ) regular</td>
</tr>
<tr>
<td><img src="image3" alt="Shape" /></td>
<td>8</td>
<td>yes</td>
<td>( \text{Symm} \neq D_n )</td>
<td>( \nabla \nabla \nabla ) regular</td>
</tr>
<tr>
<td><img src="image4" alt="Shape" /></td>
<td>3</td>
<td>yes</td>
<td>( \text{Symm} \equiv D_n )</td>
<td>( \nabla \nabla \nabla ) regular</td>
</tr>
</tbody>
</table>
Example PSL activity invariant: state-based precondition constraints

- Any occurrence of the activity depends only on fluents that hold prior to the activity occurrence. 
  \[(\forall a) \text{markov-precond}(a) \equiv ((\forall o_1, o_2) \text{state-equiv}(o_1, o_2) \supset \text{poss-equiv}(a, o_1, o_2))\]

- Some (but not all) occurrences of the activity depend only on fluents that hold prior to the occurrence. 
  \[(\forall a) \text{partial-state}(a) \equiv (\exists o_1) ((\forall o_2) \text{state-equiv}(o_1, o_2) \supset \text{poss-equiv}(a, o_1, o_2)) \land (\exists o_3, o_4) \text{state-equiv}(o_3, o_4) \land \neg \text{poss-equiv}(a, o_3, o_4)\]

- There is no relationship between occurrences of the activity and fluents holding prior to those occurrences. 
  \[(\forall a) \text{rigid-state}(a) \equiv (\forall o_1) (\exists o_2) \text{state-equiv}(o_1, o_2) \land \neg \text{poss-equiv}(a, o_1, o_2)\]

Use invariants to characterize integration targets

- Process Information Exchange (PIE) Profiles: 
  Values for each invariant for each concept 
  - Example: AtomicProcess definition from DAML-S profile
  \[(\forall a) \text{AtomicProcess}(a) \equiv \text{primitive}(a) \land \text{markov-precond}(a) \land \text{markov-effects}(a) \lor \text{context-free}(a)\]
Applications of PIE Profiles

Retrofit formal ontologies onto legacy applications, (semi-formal) ontologies

Analyze application/ontology structure

- Profiles identify important, possibly anonymous concepts
- Subsumption inference improves understanding of concept hierarchy

Compare integration targets and perform translation

- For each \( C^A_i \), \( C^B_j \) derive \( \phi \) such that \( T_{psl} \models (\forall a) \, \phi(a) \supset (C^A_i(a) \supset C^B_j(a)) \)
- Coverage analysis: \( \phi \) describes equivalencies/overlaps/disjointness
- Translation: \( \phi \) defines rules for exchanging information
- Example:

\[
(\forall a) \, C^{alice}_1(a) \equiv \text{unconstrained}(a) \land (\text{markov\_effects}(a) \land \text{context\_free}(a))
\]

\[
(\forall a) \, C^{bob}_1(a) \equiv (\text{unconstrained}(a) \land \text{markov\_precond}(a)) \land \text{context\_free}(a)
\]

\[
T_{psl} \models (\forall a) \, \text{markov\_precond}(a) \supset (C^{alice}_1(a) \supset C^{bob}_1(a))
\]

\[
T_{psl} \models (\forall a) \, \text{markov\_effects}(a) \supset (C^{alice}_1(a) \supset C^{bob}_1(a))
\]
Summary

Semantic integration is a critical problem

- Can be addressed through formal ontology

Invariant properties in an ontology can be used to:

- Characterize applications
- Analyze existing ontologies and legacy software
- Compare and integrate systems
Acknowledgements

This work was supported in part by National Institute of Standards and Technology (NIST) Grant #70NAN33H1026, funded by the National Science Foundation (NSF), as well as the Precision Engineering Project within the Manufacturing Engineering Laboratory at NIST. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and not necessarily the supporting government and other organizations.

This presentation beautifully typeset through the power and magic of \LaTeX.

Thanks Mike & Kathy!!
Questions/More Information

PSL homepage: http://ats.nist.gov/psl/

eMail: jkopena@cme.nist.gov, gruning@cme.nist.gov

(bye-bye!)