FLOWS:

A First-Order Logic Ontology for Web Services

June 10, 2005

Michael Gruninger, Rick Hull, Sheila McIlraith

on behalf of the SWSL Committee

Who is the SWSL Committee?

- Steve Battle (Hewlett Packard)
- Abraham Bernstein (University of Zurich)
- Harold Boley (National Research Council of Canada)
- Benjamin Grosof (Massachusetts Institute of Technology)
- Michael Gruninger (NIST)
- Richard Hull (Bell Labs Research, Lucent Technologies)
- Michael Kifer (State University of New York at Stony Brook)
- David Martin (SRI International)
- Sheila McIlraith (University of Toronto)
- Deborah McGuinness (Stanford University)
- Jianwen Su (University of California, Santa Barbara)
- Said Tabet (The RuleML Initiative)

Situating FLOWS

SWSI – Semantic Web Services Initiative

http://www.swsi.org

SWSA – SWS Architecture Committee

SWSL – SWS Language Committee

Situating FLOWS

SWSI – Semantic Web Services Initiative

http://www.swsi.org

SWSA – SWS Architecture Committee

SWSL – SWS Language Committee

SWSF – SWS Framework

http://www.daml.org/services/swsf/

1) SWSO - Ontology

FLOWS – First-order Logic Ontology for Web Services (SWSO-FOL)

ROWS – Rules Ontology for Web Services (SWSO-Rules)

2) SWSL – Language

SWSL-Rules – Rules language

SWSL-FOL – First order language

3) Use Cases

Situating FLOWS

SWSI – Semantic Web Services Initiative

http://www.swsi.org

SWSA – SWS Architecture Committee

SWSL – SWS Language Committee

SWSF – SWS Framework

http://www.daml.org/services/swsf/

1) SWSO - Ontology

FLOWS – First-order Logic Ontology for Web Services (SWSO-FOL)

ROWS – Rules Ontology for Web Services (SWSO-Rules)

2) SWSL – Language

SWSL-Rules – Rules language

SWSL-FOL – First order language

3) Use Cases

Situating FLOWS

Target uses of SWS Standards

- Person on the street
- Programmer on the street
- Foundations

Focus so far

Your favorite "procedural"-style description formalism can be used to specify services within FLOWS, e.g.,

- Flowchart, BPEL, Golog, ASM, FSM, Petri net, …
- Already done this for Golog-inspired constructs

Our Position

An **unambiguously computer interpretable** description of the service descriptors, the **process model** of a Web service and its **effect on the world** are critical to automating a diversity of tasks, including discovery, invocation, composition, monitoring, verification and simulation

Our Position

An **unambiguously computer interpretable** description of the service descriptors, the **process model** of a Web service and its **effect on the world** are critical to automating a diversity of tasks, including discovery, invocation, composition, monitoring, verification and simulation

The level of detail required of the process model necessitates the use of an expressive language for modeling Web services. **We propose first-order logic**, and in particular a process model built on the **Process Specification Language (PSL)** ISO Standard 18629.

Our position is the result of experience with modeling Web services in other formalisms including OWL, Petri-Nets, FSA, situation calculus, BPEL etc.…

Goal (simple examples)

Automation of:

- Web service discovery

  Find me a shipping service that will transport frozen vegetables from San Francisco to Tuktoyuktuk.

- Web service invocation

  Buy me "The Da Vinci Code" at www.amazon.com

- Web service selection, composition and interoperation

  Make the travel arrangements for my W3C05 workshop.

- Web service execution monitoring

  Has my book been shipped yet?

Web service simulation, verification and exception handling

Representational Desiderata:

- Model-theoretic semantics
- Primitive and complex processes are first-class objects
  (we want to be able to talk about the processes & their properties)
- Taxonomic representation
- Leverages existing service ontologies (e.g., OWL-S)
- Interoperates w/ domain-specific ontologies
- Embraces and integrates with existing and emerging standards and research (BPEL, W3C choreography, ebXML, UML, WSDL, etc.)
- Explicit representation of messages and dataflow
- Captures activities, process preconditions and effects on world.
- Captures process execution history.
- Captures *partial* descriptions of WS behaviour
- Captures exceptions and compensations

Some Lessons Learned

- OWL was not sufficiently expressive to capture the semantics of the process model within the OWL-S language.

- Typical process modeling languages (e.g., Petri Nets, FSMs, pi-calculus) are generally good at defining aspects of the process model, but not things such as
  - (conditional) effects on the world
  - non-functional constraints
  - the relationships between objects in a domain
  - ...

Some Pros/Cons of FOL

+ provides a well-understood model-theoretic semantics

+ rich expressive power (e.g., variables, quantifiers, terms, etc.)
  -- overcomes expressiveness issues that have haunted OWL-S

+ enables characterization of reasoning tasks in terms of classical notions of deduction, consistency, etc.

+ enables exploitation of off-the-shelf systems such as existing FOL reasoning engines and DB query engines.

- semi-decidable and intractable for many tasks (worst case) (tractability is not about the language, but note that many intractable tasks often prove easily solved in practice)

- syntax unsuitable for common man (surface languages under development)

+ provides a theoretical mechanism for preserving semantics and relating different SWS ontologies

+ enables (easy) mapping to lite versions of ontology

+ provides basis for blending results about SWS origins in different methodologies (e.g., automata-based, DL-based, Petri-net based, sitcalc-based, etc)

+ easily incorporate pre-existing work. Can import other ontologies relatively seamlessly

What is FLOWS?

FLOWS is:

a First-order Logic Ontology for Web Services

FLOWS comprises:

- Service Descriptors
- Process Model

FLOWS Process Model

- FLOWS Process Model consists of
  - a subset of the PSL Ontology
  - extensions for service concepts

The bulk of this already exists and has been vetted.

… so here’s an overview of PSL….

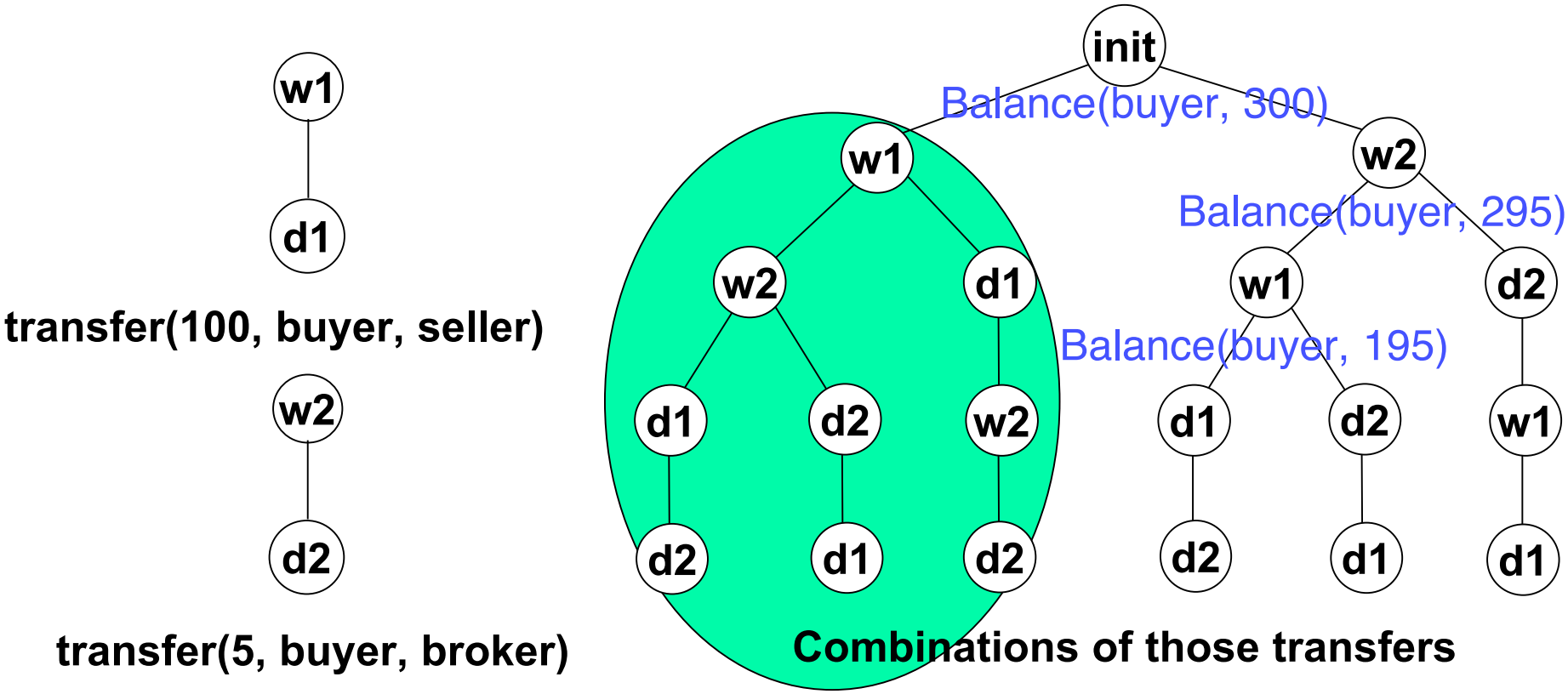
Process Specification Language

- PSL is a modular, extensible first-order logic ontology capturing concepts required for manufacturing and business process specification
  - PSL is an International Standard (ISO 18629)
  - There are currently 300 concepts across 50 extensions of a common core theory (PSL-Core), each with a set of first-order axioms written in Common Logic (ISO 24707)
  - The core theories of the PSL Ontology extend situation calculus
  - PSL is a verified ontology -- all models of the axioms are isomorphic to models that specify the intended semantics

Simple illustration of PSL model theory

Atomic activities:

- **w1 = withdraw (100, buyer)**
- **d1 = deposit (100, seller)**
- **w2 = withdraw (5, buyer)**
- **d2 = deposit (5, broker)**



- Can add constraints, e.g., that w1 must precede w2
- Can use FOL inference or domain-specific reasoning

FLOWS Process Model

FLOWS-Core

- PSL-Core
- Service, AtomicProcess, composedOf, message, channel

FLOWS Extensions

- Control Constraints
  - Split, Sequence, Unordered, Choice, IfThenElse, Iterate, RepeatUntil
- Ordering Constraints
  - OrderedActivity
- Occurrence Constraints
  - OccActivity
- State Constraints
  - TriggeredActivity
- Exception Constraints
  - Exception

FLOWS-core

- Web service
  - Named object
  - Has non-functional properties
  - Has a PSL activity (which describes the internal process of the service)
  - Can have multiple occurrences (instantiations of the service)
- AtomicProcess
  - Domain specific: analogous to OWL-S atomic processes; can impact “the real world”
  - Service specific: mainly for message handling
    - Create message (which can include place into a channel)
    - Read message
    - Destroy message
  - Also service-specific processes for channels
    - Create channel, destroy, add/delete source, add/delete target
- Messages
  - First-class objects that are created and destroyed, can be read
  - Can be placed on channels (as one mechanism to control data flow)

Driving Home Some Points

Two ways to use FLOWS

1) Describe your web services in FLOWS.

2) Use FLOWS to define the semantics of your favourite modeling paradigm (e.g., UML, ASM, FSM, Petri nets).

   – *FLOWS provides an excellent SWS Framework for relating different WS/process modeling paradigms, ensuring semantic interoperation between different modeling paradigms*.

"How might the programmer-on-the-street describe web services in FLOWS?"

- In the current FLOWS ontology, the "Control Constructs" extension on top of FLOWS-Core provides a flowchart-style process model for the "programmer on the street"
- Other "procedural" models can be incorporated into FLOWS in an analogous manner

Driving home some points (cont.)

"Reasoning in FOL is too hard." FLOWS is an ontology. It provides an unambiguous (computer interpretable) specification of a process model. While our driving tasks are characterizable in FOL using entailment and consistency, we are not (necessarily) advocating that they be implemented using a full FOL reasoner. We anticipate the use of highly-optimized special-purpose reasoners.

"Reasoning in FOL is intractable" Problems are intractable, not languages.

"FLOWS/PSL is too hard to learn and write." We don't expect the average user to ever see or write in FLOWS . This is the assembly language that ensures everything works correctly. We anticipate 95% of the users working with a much less expressive high-level syntax that hides all these details.

"There's too much detail in this language." If you don't need it, don't use it, but it's there if you do need it.

Want to learn more?

http://www.daml.org/services/swsf