Objectives of Tutorial

• Conceptual
  – Conceptual model for WSMO

• Language
  – Present the syntax and semantics for WSML
  – Show the reasoning support for WSML
  – Discuss approach to service discovery
  – Walk through WSML examples for ASG Use Case

• Hands-on
  – Translation of WSML to F-Logic
  – Using Flora to query domain ontologies and service descriptions
  – Using Flora for service discovery

• Tools
  – Show tool support for WSML

• Discussion
  – ASG scenarios and WSML
Contents

• **Part I:**
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants
    -------------- break --------------

• **Part II:**
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
    -------------- lunch 11.50 to 12:50 --------------
  – Reasoning Support using Flora2/XSB

• **Part III:**
  – Hands-on session with Flora2/XSB
    -------------- break --------------

• **Part IV:**
  – Demos: WSMX and Mediation
  – Grounding discussion
  – ASG Use Case discussion
Part I
Grid & Web Services
Convergence of Grid and Web Services (1)

Have been converging

Grid

OGSI

Web

GT1

GT2

WSDL, WS-*

WSDL 2, WSDM

HTTP

Started far apart in apps & tech

However, despite enthusiasm for OGSI, adoption within Web community turned out to be problematic
OGSA

• Open Grid Services Architecture
  – Collection of services with standardized interfaces
    • Infrastructure services
    • Data services
    • Execution management services
    • Resource management services
    • Security services
    • Information services
  – How to build all these Services?
    • Web services
    • Or identify grid specific requirements & evaluate web services standards
OGSI

Introduces semantics and capabilities to Web Services
• GWSDL
• portType inheritance
• Statefulness (SDEs)
• Stateful interactions
• Transient instances
  – Factories, GSH & GSR
• Lifetime management
• Introspection
• Notification on state changes
Grid Service

OGSA
Defines and is based on
specifies
implements
extension of
OGSI
Grid Service
Web Service
Standard interoperable technologies
XML, WSDL, SOAP, ...
GT3
Web Services Concerns with OGSI

- Too much stuff in one specification

- Does not work well with existing Web services tooling
  - Extended WSDL & therefore not compatible with other Web service standards..

- Too “object oriented”
  - Grid services extended Web services – means that Grid services are ON THEIR OWN…
The definition of WSRF means that Grid and Web communities can move forward on a common base. Support from major WS vendors, especially in management space: e.g. HP, IBM.
WSRF
WSRF

- WS-ResourceProperties
- WS-ResourceLifetime
- WS-RenewableReferences
- WS-ServiceGroup
- WS-BaseFault
- WS-Notification
Semantic Web & Web Services
The General Vision

[Butler 2003]
The General Vision

The Semantic Web

• Next generation of the Web where information has machine-processable and machine-understandable semantics
• Bring structure to the meaningful content of Web pages
• Not a separate Web: build on existing Web giving information well-defined meaning
Ontologies

• Core concept of the Semantic Web that allows the representation of data in a machine processable way

• Ontologies also human readable

• Ontologies are introduced to provide machine-understandable semantics
  - “Formal, explicit specification of a shared conceptualization” [Gruber, 1993]
Ontologies – Breakdown

- **Concepts:**
  - Basic elements of the domain, often organized in taxonomies and having attributes

- **Instances:**
  - Instantiation of concepts

- **Relations:**
  - Express relationship between concepts in the domain

- **Function:**
  - Special type of relation that returns an instance of a single concept

- **Axioms:**
  - Logical expressions that help to formalize domain knowledge
Ontologies - Concepts

- **Concept:**
  
  ```
  concept country
  subConceptOf geographicalLocation

  nonFunctionalProperties
  dc:description hasValue "country details"
  endNonFunctionalProperties

  inContinent ofType continent
  population ofType xsd:double
  extension ofType xsd:integer
  isoCode ofType xsd:string
  ```

- **Instance:**
  
  ```
  instance ireland memberOf country

  inContinent hasValue europe
  population hasValue 4000000
  extension hasValue 353
  isoCode hasValue "ie"
  ```
Semantic Web Services
Semantic Web Services

Semantic Web Services =

Semantic Web Technology +

Web Service Technology
Semantic Web Services

Web Services: [Stencil Group]

- Loosely coupled, reusable components
- Semantically encapsulate discrete functionality
- Distributed
- Programmatically accessible over standard internet protocols
Semantic Web Services

Semantic Web:
- Ontologies - basic building block
- Allow machine supported data interpretation

Semantic Web Services:
- Allow the automatic publication, discovery, selection, composition, mediation and execution of inter-organization business logic
- Internet becomes a global common platform to support SWS applications
Semantic Web Services

Usage Process – Phase 1

• Publication:
  – Make available the service description

• Discovery:
  – Locate services suitable for a given task

• Selection:
  – Choose the most appropriate service

• Composition:
  – Combine services to achieve a goal

• Mediation:
  – Solve mismatches (data, protocol, process) among the combined

• Execution:
  – Invoke services following programmatic conventions
Semantic Web Services

Usage Process – Phase 2

• Monitoring:
  – Keep track of state of execution

• Compensation:
  – Need transactional support to undo or mitigate unwanted effects

• Replacement:
  – Facilitate the substitution of services by equivalent ones

• Auditing:
  – Verify that service execution occurred in the expected way
Web Services as Programming Technology

- The web is organized around URIs, HTML, and HTTP
  - URIs provide unique identifiers on the Web
  - HTML provides a standardized way to describe document structures
  - HTTP provides the transport protocol
- Web Services are similarly organised around UDDI, WSDL, and SOAP
Web Services as Programming Technology

- UDDI Registry
  - Points to Description
  - Points to Service
  - Finds Service

- WSDL
  - Describes Service

- Service Consumer
  - Communicates with XML Messages

- SOAP

ASG C1 and C2, Potsdam, Germany, December 13, 2004
Web Services as Programming Technology

- UDDI Registry
  - Points to Description
  - Finds Service

- Service Consumer
  - Communicates with XML Messages
  - SOAP
  - Points to Service

- WSDL
  - Describes Service

ASG C1 and C2, Potsdam, Germany, December 13, 2004
The General Vision

Bringing the web to its full potential

Dynamic

Web Services
UDDI, WSDL, SOAP

Intelligent Web Services

Static

WWW
URI, HTML, HTTP

Semantic Web
RDF, RDF(S), OWL, WSMO

Syntactic

Semantic
Contents

• **Part I:**
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

  -------------- break --------------

• **Part II:**
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

  -------------- lunch --------------

• **Part III:**
  – Hands-on session with Flora2/XSB

  -------------- break --------------

• **Part IV:**
  – Demos: WSMX and Mediation
  – ASG Use Case discussion
WSMO Conceptual Model
WSMO - Introduction

- Why a service specification language?
  - To allow automatic tasks (e.g. discovery, selection, composition, mediation, execution, monitoring, etc.) to be performed with respect to services in the context of web and grid.

- What is needed?
  - A conceptual model that captures core elements of the problem that has to be solved.
  - A formal language that reflects the conceptual model and allows for different proofs.
WSMO

- An ontology derived from and based on the Web Service Modeling Framework (WSMF).
- A meta-model for semantic web services and related aspects.
WSMO Top Level Notions

Objectives that a client may have when consulting a Web Service

- Provide the formally specified terminology of the information used by all other components
- Connectors between components with mediation facilities for handling heterogeneities

Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)

ASG C1 and C2, Potsdam, Germany, December 13, 2004
Non-Functional Properties (1)

- Every WSMO element is described by properties that contain relevant, non-functional aspects of the item.
- Used for management and element overall description.

Core Properties:
- **Dublin Core Metadata Element Set** plus **version** (evolution support) and **owner**
- **W3C-recommendations** for description type.

Web Service Specific Properties:
- Quality aspects and other non-functional information of Web Services.
- used for Service Discovery & Selection.
Non-Functional Properties (2)

- Which non functional properties apply to which WSMO element is specified in the description of each WSMO element.
- We do not enforce restrictions on the range value type, but in some cases we do provide additional recommendations.
Non-Functional Properties - example

• An example of NFPs for the “Find Phone Provider” Service:

```xml
nonFunctionalProperties
  dc:title hasValue "Find Phone Provider Web Service"
  dc:creator hasValue "Titi and loan"
  dc:description hasValue "Service for finding a phone provider."
  dc:publisher hasValue "Example.org"
  dc:contributor hasValues "Adrian"
  dc:date hasValue "2004-10-22"
  dc:type hasValue <<http://www.wsmo.org/2004/d2/#service>>
  dc:format hasValue "text/html"
  dc:language hasValue "en-us"
  dc:coverage hasValues {tc:austria, tc:germany}
  dc:rights hasValue <<http://example.org/privacy.html>>
  owner hasValue inst:Telekom
  version hasValue "$Revision 0.01 $"
  financial hasValue (usageCost memberOf Cost and usageCost.min = 1 and usageCost.max = 3)
endNonFunctionalProperties
```
WSMO Ontologies

Objectives that a client may have when consulting a Web Service

- Provide the formally specified terminology of the information used by all other components

Semantic description of Web Services:
- **Capability** *(functional)*
- **Interfaces** *(usage)*

Connectors between components with mediation facilities for handling heterogeneities
Ontology Specification

- **Non functional properties** (see before)
- **Imported Ontologies** importing existing ontologies where no heterogeneities arise
- **Used mediators**: OO Mediators (ontology import with terminology mismatch handling)

- ‘Standard’ Ontology Notions:
  - **Concepts**: set of concepts that belong to the ontology, incl.
  - **Attributes**: set of attributes that belong to a concept
  - **Relations**: define interrelations between several concepts
  - **Functions**: special type of relation (unary range = return value)
  - **Instances**: set of instances that belong to the represented ontology
  - **Axioms**: axiomatic expressions in ontology (logical statement)
Concepts in Ontologies

- **Concept definition:**

  ```xml
  Class concept
  hasNonFunctionalProperty type nonFunctionalProperty
  hasSuperConcept type concept
  hasAttribute type attribute
  hasDefinition type logicalExpression multiplicity = single-valued
  ```

- **Concept examples:**

  ```xml
  concept mobileCellPhone subConceptOf telephone
  nonFunctionalProperties
  dc:description hasValue "mobileCellPhone subclass of telephone"
  endNonFunctionalProperties
  standard ofType xsd:string
  hasWAPSupport ofType xsd:boolean
  hasMessagingSupport ofType xsd:boolean // SMS or MMS
  concept communicationDevice
  nonFunctionalProperties
  dc:description hasValue "general class - communication device"
  endNonFunctionalProperties
  name ofType xsd:string
  width ofType xsd:integer
  height ofType xsd:integer
  depth ofType xsd:integer
  manufacturer ofType xsd:string
  hasLocation ofType loc:geographicalLocation
  ```
Relations in Ontologies

- Relation definition:

```
Class relation
hasNonFunctionalProperty type nonFunctionalProperty
hasSuperRelations type relation
hasParameters type parameter
hasDefinition type logicalExpression multiplicity = single-valued
```

- Relation examples:

```
relation isInRange

nonFunctionalProperties
dc:description hasValue "test if the mobilePhone is in the range of the radio antenna"

antenna ofType radioAntenna
mobile ofType mobileCellPhone
definedBy
forall ?x ?y {
    isInRange[antenna hasValue ?x, mobile hasValue ?y]
    ?x memberOf radioAntenna and
    ?y memberOf mobileCellPhone and
    (distance(radioAntenna.hasLocation, mobileCellPhone.hasLocation) <= radioAntenna.range))
```
Axioms in Ontologies

- **Axiom definition:**

  ```
  Class axiom
  hasNonFunctionalProperty type nonFunctionalProperty
  hasDefinition type logicalExpression
  ```

- **Axiom examples:**

  ```
  axiom validFrequencyRange
  nonFunctionalProperties
  dc:description hasValue "defines the valid frequency range of mobile telephony"
  endNonFunctionalProperties
  definedBy
  constraint
  ?X <500 and ?X>3000.
  ```
Instances in Ontologies

- **Instance definition:**

- **Instance examples:**

```plaintext
Class instance
  hasNonFunctionalProperty type nonFunctionalProperty
  hasType type concept
  hasAttributeValues type attributeValue

concept myTelephoneNumber memberOf mobileCellNumber

nonFunctionalProperties
  dc:description hasValue "My handy number"

endNonFunctionalProperties
  countryCode hasValue 0043
  networkCode hasValue 664
  userCode hasValue 4932607
  hasProvider hasValue {par:A1}
```
WSMO Goals

Objectives that a client may have when consulting a Web Service

- **Capability** (functional)
- **Interfaces** (usage)

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities

Semantic description of Web Services:
Goals

• De-coupling of Request and Service
  Goal-driven Approach, derived from AI rational agent approach
  - Requester formulates objective independent / without regard to services for resolution
  - ‘Intelligent’ mechanisms detect suitable services for solving the Goal
  - Allows re-use of Services for different purposes

• Usage of Goals within Semantic Web Services
  – A Requester, that is an agent (human or machine), defines a Goal to be resolved
  – Web Service Discovery detects suitable Web Services for solving the Goal automatically
  – Goal Resolution Management is realized in implementations
Goal Specification (1)

- Non functional properties
- Imported Ontologies
- Used mediators
  - OO Mediators: for importing ontologies with integration
  - GG Mediator:
    - Goal definition by reusing an already existing goal
    - Allows specification of Goal Ontologies
- Post-conditions
  Describe the state of the information space that is desired.
  - The result expected from execution a Web Service
  - Expressed as an axiom (unambiguous, based on ontology)
- Effects
  Describe the state of the world that is desired.
  - Expected changes in the world that shall hold after a service execution
  - Expressed as an axiom (unambiguous, based on ontology)
Goal Specification (2)

- **Service definition**
  
  ```
  Class goal
  nonFunctionalProperties type nonFunctionalProperties
  importedOntologies type ontology
  usedMediators type {ooMediator, ggMediator}
  postConditions type axiom
  effects type axiom
  
  goal <https://asg-platform.org/wiki/sub/Internal/0-1/goalFindPhoneProvider.weml>
  
  nonFunctionalProperties
  dct:title hasValue "Finding a phone provider"
  dct:creator hasValue "DERI International"
  dct:description hasValue "Express the goal of finding a phone provider"
  dct:publisher hasValue "DERI International"
  dct:contributor hasValues ("Dumitru Roman")
  dct:date hasValue "2004-10-04"
  dct:type hasValue <http://www.w3.org/2004/d2#goals>
  dct:format hasValue "text/html"
  dct:rights hasValue <http://deri.at/privacy.html>
  owner hasValue "PersonX"
  version hasValue "$Revision: 0.1 $"
  endNonFunctionalProperties
  
  postcondition
  axiom findPhoneProvider
  nonFunctionalProperties
  dct:description hasValue "This goal expresses the general desire of finding a phone provider."
  endNonFunctionalProperties
  definedBy
  (TelephoneProvider memberOf part telephoneProvider)
  ```

- **Service example**
WSMO Web Services

Objectives that a client may have when consulting a Web Service

- Provide the formally specified terminology of the information used by all other components
- Connectors between components with mediation facilities for handling heterogeneities

Semantic description of Web Services:
- **Capability** (functional)
- **Interfaces** (usage)
Service Descriptions (1)

Capability (the functionality of the service)

How the service achieves its capability by means of interactions with its user (communication).

Choreography

Service

Internal logic

(not of interest in Service Description)

Orchestration

How the service achieves its capability by making use of other services (coordination).

Service

Service

Service
Service Descriptions (2)

- **Non Functional Properties**: besides the common properties, contain also service specific properties: security, reliability, performance, robustness, etc.
- **Imported Ontologies**: import ontologies as long as no conflicts are needed to be resolved.
- **Used Mediators**: deal with the ontology, behavior, protocol heterogeneity.
- **Capability**: functional description of the service.
- **Interfaces**: how the functionality of the service can be achieved
  - *Choreography* - the interaction protocol for accessing the service
    - Execution choreography
    - Mata choreography
  - *Orchestration* - how the service makes use of other services in order to achieve its capability.
Service Descriptions (3)

- Service definition
- Service example

webService https://asg-platform.org/twiki/pub/Internal/C-1/getPhoneLocation.wsml

nonFunctionalProperties
dc:title hasValue "Get Phone Location Web Service"
dc:creator hasValue "DERI International"
dc:description hasValue "Service for locating a phone by Vodafone."
dc:publisher hasValue "ASG Project"
dc:contributor hasValue "[Dumitru Roman, Ioan Toma]"
dc:date hasValue "2004-11-15"
dc:type hasValue "http://www.w3.org/2004/02/wselement">
dc:format hasValue "text/html"
dc:language hasValue "en-us"
dc:rights hasValue "http://www.deri.org/privacy.html">
nonFunctionalProperties
capability "https://asg-platform.org/twiki/pub/Internal/C-1/getPhoneLocation/"

interface _#
nonFunctionalProperties
dc:description hasValue "Describes the Interface of get phone location web service."
nonFunctionalProperties

choreography
nonFunctionalProperties
dc:description hasValue "Currently this is a placeholder, however here is where the mapping of this atomic service to a WSDL operation is defined."
nonFunctionalProperties
Service Capability (1)

- **Non functional properties**
- **Imported Ontologies**
- **Used mediators**
  - *OO Mediator*: importing ontologies as terminology definition
  - *WG Mediator*: link to a Goal that is solved by the Web Service
- **Pre-conditions**
  What a web service expects in order to be able to provide its service. They define conditions over the input.
- **Assumptions**
  Conditions on the state of the world that has to hold before the Web Service can be executed
- **Post-conditions**
  describes the result of the Web Service in relation to the input, and conditions on it
- **Effects**
  Conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world)
Service Capability (2)

- Service Capability Definition

- Service Capability Example
WSMO Mediators

Objectives that a client may have when consulting a Web Service

- **Semantic description of Web Services:**
  - **Capability** *(functional)*
  - **Interfaces** *(usage)*

Provide the formally specified terminology of the information used by all other components

Connectors between components with mediation facilities for handling heterogeneities
Mediation

• Heterogeneity …
  – Mismatches on structural / semantic / conceptual / level
  – Occur between different components that shall interoperate
  – Especially in distributed & open environments like the Internet

• Concept of Mediation (Wiederhold, 94):
  – *Mediators* as components that resolve mismatches
  – Declarative Approach:
    • Semantic description of resources
    • ‘Intelligent’ mechanisms that resolve mismatches independent of content
  – Mediation cannot be fully automated (integration decision)

• Levels of Mediation within Semantic Web Services (WSMF):
  (1) Data Level: mediate heterogeneous *Data Sources*
  (2) Protocol Level: mediate heterogeneous *Communication Patterns*
  (3) Process Level: mediate heterogeneous *Business Processes*
WSMO Mediators Overview
Mediator Structure

WSMO Mediator

- as a Goal
- directly
- optionally incl. Mediation

Source Component

1..n

Target Component

Source Component

Mediation Services

Class mediator
hasNonFunctionalProperties type nonFunctionalProperties
importsOntology type ontology
hasSource type [ontology, goal, service, mediator]
hasTarget type [ontology, goal, service, mediator]
hasMediationService type [goal, service, wwmMediator]
GG Mediators

• **Aim:**
  - Support specification of Goals by re-using existing Goals
  - Allow definition of **Goal Ontologies** (collection of pre-defined Goals)
  - Terminology mismatches handled by OO Mediators

• **Example: Goal Refinement**

```
Source Goal
“Buy a ticket”

GG Mediator
Mediation Service

Target Goal
“Buy a Train Ticket”

postcondition:
“aTicket memberof trainticket”
```
Contents

• **Part I:**
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

------------------ break ------------------

• **Part II:**
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

------------------ lunch ------------------

• **Part III:**
  – Hands-on session with Flora2/XSB

------------------ break ------------------

• **Part IV:**
  – Demos: WSMX and Mediation
  – ASG Use Case discussion
WSML – Overview, Syntax and Variants
WSML: Overview

• Introduction to WSML
• Rationale of WSML
• Syntaxes for WSML
• WSML Variants
  – WSML-Core
  – WSML-Flight
  – WSML-OWL
  – WSML-Full
• Conclusions
Web Service Modeling Language

• Four elements of WSMO:
  – Ontologies
  – Goals
  – Web Services
  – Mediators

• WSML provides a formal grounding for the conceptual elements of WSMO, based on:
  – Description Logics
  – Rule Languages
  – First-Order Logic
Rationale of WSML

• Provide a Web Service Modeling Language based on the WSMO conceptual model
  – Concrete syntax
  – Semantics

• Provide a Rule Language for the Semantic Web
• Many current Semantic Web languages have
  – undesirable computational properties
  – unintuitive conceptual modeling features
  – inappropriate language layering
    • RDFS/OWL
    • OWL Lite/DL/Full
    • OWL/SWRL
Syntaxes for WSML

• Human-readable syntax
  – Part of WSMO-Standard
    • WSMO-Standard defines intuitive semantics
  – Modular syntax
    • WSMO-syntax functions as “umbrella”
    • Modules for different WSML variants
  – Syntax:
    • Inspired by OIL/OWL and F-Logic
    • Conceptual syntax
    • Logical Expression Syntax
  – Semantics is fixed in WSML variants
• XML syntax
  – Based on human-readable syntax
• OWL/RDF syntax
  – Based on human-readable syntax
Variants of WSML

WSML-DL \(\rightarrow\) First-Order Logic (with nonmonotonic extensions) \(\rightarrow\) WSML-Full

WSML-Core \(\rightarrow\) WSML-Flight \(\rightarrow\) WSML-Rule

Description Logics

Logic Programming

First-Order Logic (with nonmonotonic extensions)

ASG C1 and C2, Potsdam, Germany, December 13, 2004
Variants of WSML (contd.)

- **WSML-Core**
  - Based on the intersection of Description Logics and Datalog
  - Semantics defined through OWL Lite
  - Has (frame-based) conceptual syntax and logical expression syntax

- **WSML-Flight**
  - Based on OWL Flight
    - Basic meta-class facility
    - Constraints
    - Non-monotonic features (default negation)
  - “Fixes” some of the mistakes in OWL, such as unintuitive modeling constructs
  - Preferred ontology modeling language
Variants of WSML (contd.)

- **WSML-Rule**
  - Based on Logic Programming with default negation and F-Logic/HiLog syntactical extensions
  - Preferred goal/web service modeling language
- **WSML-DL**
  - Based on species of OWL
- **WSML-Full**
  - Combining FOL with minimal models and non-monotonicity
WSML Conceptual Syntax for Ontologies

- Ontologies
- Namespaces
- Imported Ontologies
- Used Mediators

- Concepts
- Relations
- Functions
  - Special kind of relation
- Instances
  - Explicitly defined in ontology
  - Retrieved from external instance store
- Axioms

Extra-Logical declarations

Logical Declarations

Non-Functional Properties
WSML Logical Expressions

- Frame- and first-order-based concrete syntax (BNF Grammar in D2, Appendix B)

- Elements:
  - Function symbols (e.g. \( f() \))
  - Molecules (e.g. Human \( \text{subClassOf} \) Animal, John \( \text{memberOf} \) Human, John[name hasValue ‘John Smith’]).
  - Predicates (e.g. distance(to:?x, from:?y, distance:?z))
  - Logical connectives (or, and, not, implies, equivalent, impliedBy, forall, exists)

- Example:

  \[
  ?x \text{memberOf Human equivalent}
  \]

  \[
  ?x \text{memberOf Animal and } ?x \text{memberOf LegalAgent}.
  \]
WSML Goals and Web Services

• Goal / Web Service
  – assumptions
  – effects
  – pre-conditions
  – post-conditions
  are defined through WSML logical expressions

• Logical expressions are based on ontologies
WSML-Core

- Allows conceptual modeling of ontologies
- Based on OWL Lite- (a subset of OWL Lite, for which a translation exists to Datalog)
  - Efficient query answering
  - Allows to take advantage from optimization techniques developed in database research
  - Many existing implementations (e.g. XSB, OntoBroker, SWI-Prolog, KAON, DLV)
  - Import/export OWL ontologies
- Expressive enough for most current ontologies
- Can be used for limited goal/web service modeling
Conceptual Syntax

```
concept ticket
  origin ofType location
  destination ofType location
  departure ofType timeStamp
  arrival ofType timeStamp
  fare ofType price
```

definedBy

```
?x memberOf ticket impliedBy ?x[origin hasValues ?y, destination hasValues ?z, fare hasValues ?y].
```

Logical Expression Syntax
WSML-Core - Syntax

• Conceptual syntax is slight restriction of WSML (e.g. only binary relations allowed)
  – Small extensions to cover epistemology of OWL Lite- (e.g. transitive, symmetric, inverse properties)

• Logical expression syntax is big restriction of WSML
  – Only patterns that can be translates to OWL Lite-
WSML-Core - Semantics

- Defined through a translation to OWL Lite-Abstract Syntax
- OWL Abstract Syntax & Semantics document provides direct model-theoretic semantics
- WSML-Core semantics falls in intersection Description Logics and Datalog
WSML-Flight

- Based on OWL Flight
  - Extends OWL Full (Datalog subset of OWL Full)
  - Adds UNA
  - Adds constraints
  - Adds non-monotonic features
- Is an extension of WSML-Core
  - Adds limited support for nominals
  - Meta-modeling
  - Intuitive semantics for attributes
  - Extensive datatype support, based on datatype groups [Pan & Horrocks, 2004]
- Limited support for Goal and Web Service descriptions
Conceptual Syntax

```plaintext
concept ticket
  origin ofType location
  destination ofType location
  departure ofType xsd:dateTime
  arrival ofType xsd:dateTime
  fare ofType price
```

Axiom validDates

```plaintext
axiom validDates
  definedBy
    <- ?xmemberOf ticket[arrival hasValue ?y,
      departure hasValue ?z] and ?y < ?z.
```

Logical Expression Syntax
WSML-Rule

- Based on Logic Programming-variant of F-Logic and HiLog
- Minimal model semantics
- Implements default negation
- Allows unrestricted use of function symbols
- Full support for goal/web service modeling
WSML-Rule - Example

Logical Expression Syntax

effect ticket
definedBy
   (itinerary(?Req)[from hasValue ?From, to hasValue ?To]
      impliedBy
         ?Input = search(?Req,
            ?From memberOf (france or germany),
            ?To memberOf austria)
   and
   (ticket(?Req)[confirmation hasValue ?Num,
         from hasValue ?From, to hasValue ?To, date hasValue ?Date]
      impliedBy
         ?Input = contract(?Req,?From,?To,?Date,_CCard) and
generateConfNumber(?Num)).
WSML-DL

• WSML syntax – OWL semantics
• (to be developed)

• OWL epistemology:
  – Complete class definitions
  – Range/cardinality restrictions
Conceptual Syntax

```
completeConcept ticket
    origin ofType (all,some) location
destination ofType (all,some) location
departure ofType (some) xsd:dateTime
arrival ofType xsd:dateTime
fare ofType (some, <=1) price

relation origin subRelationOf hasLocation
    functional

relation destination subRelationOf hasLocation
    functional
```
WSML-Full

- Based on a combination of First-Order Logic and minimal model semantics and default negation
- Unifies rule language with first-order based language (e.g. OWL)
- For now only theoretical language
WSML Summary

• Formal languages for WSML

• Variants:
  – WSML-Core
  – WSML-Flight
  – WSML-Rule
  – WSML-DL
  – WSML-Full

• Modular, Frame-based

• Conceptual syntax vs. Logical Expressions

• Syntaxes:
  – Human readable
  – XML
  – OWL/RDF
Contents

- **Part I:**
  - Grid and Web Services
  - Semantic Web Services
  - WSMO Conceptual Model
  - WSML Overview, Syntax and Variants

  -------------- break --------------

- **Part II:**
  - WSML Service and Ontology descriptions (use case example)
  - Approaches for Reasoning Support for WSML and Discovery
  - Reasoning Support using Flora2/XSB

  -------------- lunch --------------

- **Part III:**
  - Hands-on session with Flora2/XSB

  -------------- break --------------

- **Part IV:**
  - Demos: WSMX and Mediation
  - ASG Use Case discussion
Part II
WSML Service and Domain Ontology
Examples from Buddy Use Case Scenario
Contents

• **Part I:**
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

  -------------- break --------------

• **Part II:**
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

  -------------- lunch --------------

• **Part III:**
  – Hands-on session with Flora2/XSB

  -------------- break --------------

• **Part IV:**
  – Demos: WSMX and Mediation
  – ASG Use Case discussion
A Conceptual Model for Service Description and Discovery in WSMO
Why automated discovery of WS?

- Web services are an attractive technology for integrating components within a software system

- Currently, web services have to be selected and hard-wired at design time
  - No **dynamic reconfiguration** of the system at runtime is possible
  - Human intervention is needed and costly

- Semantic descriptions of WS can enable the automatic location of web services providing particular functionality

- Autom. Disc. is a basic task for doing composition efficiently
Web Service vs. Service

- Notions of **Web Service** & **Service** are often interpreted in various ways in the literature
- We use the following **terminology & interpretation** here
  - **Service**
    - A **provision of value in some domain** (not necessarily monetary, independent of how service provider and requestor interact)
  - **Web Service**
    - **Computational entity** accessible over the Internet (using Web Service Standards & Protocols), **provides access to (concrete) services** for the clients.

- Thus, we have the following **relation between the notions**:
  - **Service** corresponds to a **concrete execution of a web service** (with given input values)
  - **Web Service** provides a **set of services** to ist client; one service for each possible input value tuple
Web Service vs. Service (II)

Abstract State Space

Web Service

Abstract State Space
Automated Discovery

• **The task**
  – Client is interested in getting a **specific service** $S$ (descr. by Goal $G$)
  – Identify possible **web services** $W$ which are able to provide the requested service $S$ for its clients

• **Discovery** :=
  – Given a goal, possibly some input data – **In general**: Disc.-Request $R$
  – Process of identifying suitable web services $W$ that can provide the requested service (relative to some Web Service repository $RP$)

• **Fundamental Aspect of Discovery**: *Matching*
  Along different **possible dimensions** $D$: $\text{match}_D(W,R)$
  - **Capability** *(service semantics)*
  - **Non-functional properties** *(Quality of Service Parameters, Provider)*
  - **Choreography** *(how to interact with the web service)*
Descriptions & Discovery

• The task
  – Identify possible web services \( W \) which are able to provide the requested service \( S \) for its clients

• An important issue …
  – "being able to provide a service" has to be determined based on given descriptions only (\( WS, \) Goal, Ontos)
  – Discovery can only be as good as these descriptions

  • Very detailed WS descriptions: are precise, enable highly accurate results, are more difficult to provide; in general, requires interaction with the provider (outside the pure logics framework)
  • Less detailed WS descriptions: are easy to provide for humans, but usually less precise and provide less accurate results
• We aim at supporting a wide-variety of clients and applications
  – Support different description techniques for clients
  – Support a wide-variety of applications wrt. needed accuracy
  – **Main focus here**: Capability – What does the service deliver?

• **Basic possibilities for the description of web services:**
  – **Syntactic approaches**
    • Keyword-based search, natural language processing techniques, Controlled vocabularies
  – **Lightweight semantic approaches**
    • Ontologies, What does W provide (not how)?, Action-Object-Modelling, Coarse-grained semantic description of a service
  – **Heavyweight semantic approaches**
    • Describes the service capability in detail, Pre/Post-Cond, takes “in-out“ relationship into account, Fine-grained web service description
A brief excursion …

„Heavyweight“ Semantic Description of Web Services
Basic Web Service Model

- Web Service as a computational entity
  - Takes input values I1, ..., IN
  - that fulfill certain properties (precondition)
- that determine
  - Outputs O(I1, ..., IN)
  - Effects E(I1, ..., IN)

- Semantics
  - State-based: **Web Service as a state-relation** (transformation)
  - Captured by:
    - Precondition/Assumptions
    - Postcondition/Effects
  - **Hoare-Semantics**: \{Pre and Ass\} W \{Post and Eff\}
Basic Discovery Request Model

- A user has a specific desire and wants to use a web service to resolve this desire

- Hence, a service requester specifies
  - his desire by means of a goal (the state to be reached by execution of a web service)
  - Possibly data that he is able to provide as an input to a service
    - User-specific (and thus reusable across different goals)
    - Example: Creditcard number, Name, Address information, Login Information …
  - Perhaps policies for disclosing this information as input to a web service
Basic Matching Model

• Check whether
  – it is possible to execute the service s. t. a state is reached where the user desire is resolved (goal satisfies)

• That means:
  – Are there input values I1, …, IN for the (specific) web service W that the requester can provide such that
    • the precondition of W is fullfilled (that means the W halts in a state which satisfies the postcondition)
    • And in this state the goal (postcondition & effect) is satisfied

• Possible implementation: Simulate the execution of the service on a logical (symbolic) level (TA-Logic/Flora2)
Basic Matching Model – TA/Flora2

• Possible implementation
  – State == model of a logical theory (Knowledge Base)
    • Facts & Rules in the Flora2 system
  – State-Transitions == Update of the logical theory
    • Insertion & Deletion of Facts and/or Rules
    • Flora2 supports Transaction Logics (sequential operator, insert/delete)
    • Service Capability modelled as a set of rules (Flora2-Logic Programs are Turing complete)
    • Goal represented as a query (on the final state of the execution of W)
  – Basic procedure ...

Query against KB -> Get Input I

Query against updated KB
... and back to overview on the discovery model

Levels of Abstraction of Web Services Descriptions
Descriptions & Discovery (III)

- **Service provider side:**
  - Capability description & levels of abstraction

  - **What do I provide?** (Syntactically)
    - {Keyword}
    - Syntactic
  
  - **What do I provide?** (Semantically)
    - WS
    - Semantic („Light“)
  
  - **What do I provide & When (for what input)?** (Semantically)
    - Semantic („Heavy“)
• **Service requester side**: Goal description

```
What do I want? (Syntactically)  {Keyword}  K1 ... Kn
What do I want? (Semantically)  Semantic („Light“)
What do I want & What (input) can I provide? (Semant.)  Semantic („Heavy“)
```

Syntactic
• Basic idea for Matching on the single levels

Common keywords

{Keyword}

W1 ... WL K1 ... Kn

Set-theoretic relationship

Adequate (common) execution/state-transition

Syntactic

Semantic („Light“)

Semantic („Heavy“)

WS
Descriptions & Discovery (VI)

- Capability descriptions: **Layers of Capabilities**
  - How to combine various levels of abstraction?
Descriptions & Discovery (VII)

- Capability descriptions:
  - Levels of abstraction & possible accuracy?

  **What? (Syntactically)**
  - Syntactic capability
    - perhaps complete & perhaps correct

  **What? (Semantically)**
  - Abstract capability
    - complete & perhaps correct

  **What & When? (Semant.)**
  - Concrete capability
    - complete & correct (if user input known & interaction)
Descriptions & Discovery (VIII)

- Possible approaches for checking Matches and their complexity

Information Retrieval: very efficient

DL-based reasoning/deductive databases: more or less efficient

Deductive databases with TA-Logic support/ Theorem-Proving: less efficient/no guarantees

Syntactic

Semantic ("Light")

Semantic ("Heavy")
(Web) Service Discovery

• **Distinguish further between**
  – Web Service Discovery
  – Service Discovery

• **Web Service Discovery**
  – **No interaction with the provider**, matches are only based on **static capability descriptions**
  – Matching is less accurate (we can only return web services which might be able to deliver a requested service)
  – Possibly ignore preconditions and inputs in service capabilities
  – Most likely with abstract capabilities

• **Service Discovery**
  – Interaction with the provider with concrete input from user (**dynamic capabilities**)
  – Only with heavyweight descriptions of service capabilities possible (Input has to be considered)!
  – Matching is can be as accurate as possible
  – The more interaction, the less efficient becomes checking a match
Overall Discovery Process

The process envisioned at present …

- Requester Desire
  - Predefined formal Goal
  - Goal Discovery
  - Requester Goal
    - Web Service Discovery
      - Concrete Capability (possibly dynamic)
      - Still relevant WS
        - Web Service (Service Discovery)
          - Service to be returned

Ease of description
Efficient Filtering
Accuracy
Contents

• Part I:
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

  -------------- break --------------

• Part II:
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

  -------------- lunch --------------

• Part III:
  – Hands-on session with Flora2/XSB

  -------------- break --------------

• Part IV:
  – Demos: WSMX and Mediation
  – ASG Use Case discussion
Reasoning Support Using Flora2/XSB
Content

• Overview on Flora2
  • F-logic, Hilog, Transaction Logic, Top-Down Execution and Tabling

• Why Flora2?
  • Relation with WSML variants

• Using Flora2
  • Getting started, Low level details, Advanced features, Updating the knowledge base

• WSML-Core to Flora
Overview on Flora2

Introduction

- **F-LOgic tRAnslator**
- Knowledge-base programming with frames and logic
- Founded on:
  - F-logic
  - Hilog
  - Transaction Logic
- **Features:**
  - Declarative
  - Object Oriented
  - Logic Programming Style
Overview on Flora2

Problem and Solution

- Flat data structures
  - \texttt{person(john, '123 Main St.', 1)}
    - \textit{[F-logic]}

- Awkward meta-programming
  - What predicates mention John?
    - \textit{[Hilog + F-logic]}

- Modelling side effects
  - State changes, I/O
    - \textit{[Transaction Logic]}
Overview on Flora2

Applications

- Ontology Management
- Information Integration
- Software engineering
- Agents
- Applications requiring manipulations of (semi-)structured data
Overview on Flora2

F-Logic

• Basic ideas:
  • Complex data types as in the oo data bases
  • Combines them with the logic
  • Uses the result as a programming language

• Features:
  • Objects with complex internal structure
  • Class hierarchies and inheritance
  • Typing
  • Encapsulation
Overview on Flora2

F-Logic – Relation with Standard Logic

F-logic programming

F-logic
Overview on Flora2

F-Logic - Relation with Description Logic

F-logic

Description Logic

Predicate logic

Description F-Logic
Overview on Flora2

F-Logic – Examples (I)

• Object Description

  john[name->’John Doe’, phones->{1234, 9876},
  children->{bob, mary}]

• Nested Structures

  sally[spouse->john[address->’123 Main St.’]]

• ISA Hierarchy

  john::person
  alice::student
  student::person
Overview on Flora2

F-Logic – Examples (II)

• Methods

\[
P[\text{ageAsOf}(\text{Year}) \rightarrow \text{Age}] :-
P: \text{person}, P[\text{born} \rightarrow B], \text{Age is Year-B}.
\]

• Type Signature

\[
\text{person}[\text{born} => \text{integer},
\text{ageAsOf} (\text{integer}) => \text{integer},
\text{name} => \text{string}, \text{children} =>> \text{person}].
\]

• Queries

\[
?- \text{john}[\text{born}=>Y, \text{children}=>>C], C[\text{born}=>>B],
\text{Z is Y + 30, B}>Z.
\]

\[
?- \text{john}[\text{ageAsOf}(Y) \rightarrow 30, \text{children} =>> C],
\text{C[born} \rightarrow B], B>Y.
\]

\[
?- \text{person}[\text{name}=>\text{Type}].
\]
Overview on Flora2

F-Logic – Syntax

• Object Ids:
  • Terms like in Prolog – \textit{john, abc, f(john, 34)} etc.
  • \textit{O, C, M, Tobj, …} - usual first order function terms

• IsA Hierarchy:
  • \textit{O:C \ or \ O::C}

• Structure
  • \textit{O[Method -> Value] \ or \ O[Method ->>> Value]}

• Type (signatures)
  • \textit{Class[Method => Class] \ or \ Class[Method =>>> Class]}

• Combination of the above:
  • \textit{\lor, \land, negation, quantifiers}
Overview on Flora2

HiLog

- Allows certain forms of logically clean meta-programming
- Syntactically appears to be higher-order
- Semantically is first-order and tractable
- Sound and complete proof theory
Overview on Flora2

*HiLog - Examples*

- **Variables over predicates and function symbols:**
  
  \[ p(X, Y) :- X(a, Z), Y(Z(b)). \]

- **Variables over atomic formulas**
  
  \[ \text{call}(X) :- X. \]

- **Use of HiLog in Flora2:**
  
  \[ O[\text{unaryMethods}(\text{Class}) \rightarrow\rightarrow M] :- \]
  
  \[ O[M(_) \rightarrow V; M(_)
  \rightarrow\rightarrow V], V:\text{Class}. \]

  - **Reification**
    
    - \[ \text{john}[\text{believes} \rightarrow\rightarrow \{\text{mary}[\text{likes} \rightarrow\rightarrow \text{bob}]\}] \]
Overview on Flora2

HiLog - Syntax

• HiLog term:
  • $X$ and $f$ ($X$ variable, $f$ a constant)
  • $F(A_1,\ldots,A_n)$ if $F$, $A_1,\ldots,A_n$ are HiLog terms
  • Any Prolog term is a HiLog term
  • $X(a,f(Y))$, $f(f(f,g),Y(Y,Y))$, $h$, $Y$
  • $X(a,f(Y))(f(f(f,g),Y(Y,Y)))$, $h$, $Y$
  • $X(a,f(Y))(X(a,f(Y)))(f(f(f,g),Y(Y,Y)))$, $h$, $Y$

• HiLog formulas:
  • Any HiLog term
  • $A \lor B$, $A \land B$, $\neg A$, $\forall X \ A$, etc., if $A$, $B$ are HiLog formulas
Overview on Flora2

The Role of HiLog in Flora

- Allows high degree of meta-programming purely in logic
- Variables can be bound to predicate and function symbols
- Formulas can be represented as terms, decomposed, composed, and manipulated with in flexible ways
- Frame syntax and predicate syntax in the same query/program
Overview on Flora2

Transaction Logic

• A logic of change
• A logic for both programming and reasoning
• Main ideas:
  • Execution paths
  • Truth over a path = execution over that path
  • Elementary state transition
• Reasoning about the effects of actions
Overview on Flora2

Transaction Logic - Syntax

- Serial conjunction, $\bigotimes$
  - $a \bigotimes b$ - do $a$ and than do $b$

- $\land, \lor, \neg, \forall, \exists$ (but with a different semantics)
  - Example: $a \lor (b \bigotimes c) \land (d \lor \neg e)$

- $a :\neg b \equiv a \lor \neg b$
  - Means: to execute $a$ one must execute $b$ (i.e., $a$ is a subroutine)
Overview on Flora2

Top-down Execution and Tabling

• Top-down execution based on the table mechanism implemented in XSB
• SLD at the core of any top-down execution engine
• XSB is a complete implementation for SLG (SLD with negation)
• SLG is complete for Horn clauses
• Molecules and HiLog are automatically tabled by Flora
  - `flora ?- refresh{...}`
  - `?- abolish_all_tables`
Why Flora2?

Relation with WSML Variants (I)

- **WSML-Core**
  - Subsumption reasoning
  - Integrity constraints
  - Built-in functions
  - Query answering

- **WSML-Flight**
  - Equality and Stratified Negation
  - Meta-Modelling

- **WSML Rule**
  - Function Symbols
  - Unstratified Negation
  - Well founded semantics for negation
  - Not stable model semantics
  - Direct support for Hilog
Why Flora2?

Relation with WSML Variants (II)

• WSML-DL
  – Cannot handle disjunction
  – Classical Negation
  – Subsumption Reasoning

• WSML-Full
  • For the extensions added to WSML-Core
    + Good support in the direction of Logic Programming
    – Limitations in the directions of Description Logic
    – Minimal model semantics, not first order semantics
Using Flora2
Getting Started (I)

• Flora2 officially part of the XSB distribution
• After installation: runflora
• Loading a file
  flora2 ?- [myFile]. or flora2 ?- [myFile].
  flora2 ?- ['fullpath\myFile']. in Windows
  flora2 ?- ['fullpath/myFile']. in Unix
• Asking Queries
  flora2 ?- mary[works ->> where].
• Commands
  flora2 ?- flHalt. (flOne, flAll, flHelp)
Using Flora2

Getting Started (II)

• Variables
  • Uppercased alphanumeric, _, _Alphanumeric

• Atoms
  • Constants, Lowercased alphanumeric, ‘other_characters’

• Strings
  • List of characters enclosed by “ ”; e.g. “john”

• Numbers
  • Integers, Floats

• Comments
  • %, //, /*…multi line…*/
Using Flora2
Getting Started (III)

• Logical Expressions
  • In rule bodies literals combined with , and ;
  • Inside molecules , and ; are also allowed
  • Negation tnot (inside or outside molecules)

• Arithmetic Expressions
  • The variables have to be instantiated

• Modules
  • User modules
  • Flora2 system modules
  • Prolog XSB modules
  • Flora ?- [myFile >> moduleName]
Using Flora2
Low-level Details

- **Tabling**
  - **Tabled:** \( p(X,a), a[m \rightarrow X], X[p(a,b)] \)
  - **Non-tabled:** \( \#p(X,a), X[\#p(a,b)] \)

- \# applicable only to boolean methods

- Queries should use tabled methods/predicates

- Actions that have side effects should not be tabled
• Path Expressions

\[ \text{X.Y \ stands for } Z \text{ in } X[Y->Z] \]
\[ \text{X..Y \ stands for } Z \text{ in } X[Y->>Z] \]
\[ \text{X!Y \ stands for } Z \text{ in } X[Y*->Z] \]
\[ \text{X!!Y \ stands for } Z \text{ in } X[Y*->>Z] \]

\[ X[m \rightarrow Z].Y..Z[abc ->> Q] \]

is

\[ X[m \rightarrow Z], X[Y \rightarrow V], V[Z \rightarrow> W], W[abc \rightarrow> Q] \]

or

\[ X[m \rightarrow Z, Y \rightarrow V[Z \rightarrow> W[abc \rightarrow> Q]]] \]

(nested molecules)
Using Flora2
Advanced Features (II)

• Anonymous OIDs
  • Similar with blank nodes in RDF
  • Unnumbered: _.#
    – Different occurrences, different IDs
  • Numbered: _.#1, _.#2,…
    – _.#n in the same clause, the same ID
  • Plain symbols - _.#(_.#1, _.#, _.#2)
  • Can appear only in facts and rules heads

• Equality :=:
  – none, basic, flogic

Flogic equality:
john[father->president(acmeInc)]
john[father->peter]
Should derive
peter :=: president(acmeInc)
Using Flora2

Updating the Knowledge Base (I)

• Insert/delete facts in a module
  • Non-logical updates (non-backtrackable): insert, delete, insertall, deleteall, erase, eraseall
  • Logical updates (backtrackable): btinsert, btdelete, btinsertall, btdeleteall, bterase, bteraseall
    • updateOp{Literals} or updateOp{Literals|Query}

• Insert/delete rules in a module
  • Static and dynamic rules
  • insertrule_a, insertrule_z, deleterule
    • Not backtrackable

• Create a new module
WSML-Core to Flora2

Concepts

WSML

```
concept mobileCellPhone subConceptOf telephone
    nonFunctionalProperties
        dc:description hasValue "mobileCellPhone subclass of telephone"
    endNonFunctionalProperties
    standard ofType xsd:string
    hasWAPSupport ofType xsd:boolean
    hasMessagingSupport ofType xsd:boolean
```

Flora2

```
mobileCellPhone::telephone.
mobileCellPhone[ nonFunctionalProperties ->
    _#[description -> "mobileCellPhone subclass of telephone"],
    standard => string,
    hasWAPSupport => boolean,
    hasMessagingSupport => boolean
].
```
WSML-Core to Flora2

Relations

WSML

relation contains
  nonFunctionalProperties
    dc:description hasValue "(Interval, X) is a tuple of the binary relation corresponding to this function iff Interval contains X and X is an instant or an interval"
  endNonFunctionalProperties
  interval ofType interval
  interval0rInstant ofType instant0rInterval

Flora2

contains[
  nonFunctionalProperties ->
    #[(description -> "(Interval, X) is a tuple of the binary relation corresponding to this function iff Interval contains X and X is an instant or an interval"),
      param1 => interval,
      param2 => interval0rInstant
  ].
WSML-Core to Flora2

Functions

WSML

```
function secondsFromMidnight
    nonFunctionalProperties
        dc:description hasValue "(Time, SecondsFromMidnight) is a tuple of the binary relation corresponding to this function iff SecondsFromMidnight are the seconds elapsed from 00:00:00 of the same day. This simplifies the axiomatization of the difference between two given times"
    endNonFunctionalProperties
    time ofType time
    range ofType xsd:integer
```

Flora2

```
secondsFromMidnight[
    nonFunctionalProperties ->
        _# [description -> "(Time, SecondsFromMidnight) is a tuple of the binary relation corresponding to this function iff SecondsFromMidnight are the seconds elapsed from 00:00:00 of the same day. This simplifies the axiomatization of the difference between two given times"],
        param1 => time,
        range => integer
].
```
WSML-Core to Flora2

Instances

WSML

instance Vodafone memberOf telephoneProvider

instance HPI memberOf institution

Flora2

'Vodafone':telephoneProvider.

'HPI':institution.
WSML-Core to Flora2

Axioms

WSML

axiom validPDAInputMethod
    nonFunctionalProperties
    dc:description hasValue "enumerate the valid PDA input methods"
    endNonFunctionalProperties
    definedBy
        ?X hasValue "keyboard" or
        ?X hasValue "touch screen" or
        ?X hasValue "glidepad" or
        ?X hasValue "trackpoint".

Flora2

validPDAInputMethod[nonFunctionalProperties -> _#(description -> "enumerate the valid PDA input methods")].
validPDAInputMethod(X) :- X='keyboard';X='touch screen';X='glidepad';X='trackpoint'.
Reasoning Support Using Flora2/XSB
Contents

• **Part I:**
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

  -------------- break --------------

• **Part II:**
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

  -------------- lunch --------------

• **Part III:**
  – Hands-on session with Flora2/XSB

  -------------- break --------------

• **Part IV:**
  – Demos: WSMX and Mediation
  – ASG Use Case discussion
Part III
Hands-on Session
Hands-On

• Part 1 – 30 minutes
  – Create a WSML description of a domain concept
  – Translate this to F-Logic
  – Load in Flora
  – Query
• Part 2 – 30 minutes
  – Create WSML description of service
  – Load F-Logic translation into Flora
  – Query the service description
• Part 3 – 30 minutes
  – Load domain ontologies and service descriptions to Flora
  – Run query to simulate discovery
  – Discuss discovery in context of Flora
• http://sw.deri.ie/~adrian/Resources/XSBFlora2_win.zip
• http://sw.deri.ie/~adrian/Resources/XSBFlora2_cygwin.zip
Hands-On: Part 1

- Write the WSML for the communicationsDevice concept with
  - Non functional property
    - description
  - Attributes
    - name: type = string
    - width: type = integer
    - height: type = oienteger
    - depth: type = integer
    - manufacturer: type = string
    - hasLocation: type = geographicalLocation
Hands-On: Part 1

- Write the WSML for the communicationsDevice concept with
  - Non functional property
    - description
  - Attributes
    - name: type = string
    - width: type = integer
    - height: type = ointeger
    - depth: type = integer
    - manufacturer: type = string
    - hasLocation: type = geographicalLocation

```
concept communicationDevice
  nonFunctionalProperties
    dc:description hasValue "general class - communication device"
  endNonFunctionalProperties

  name ofType xsd:string
  width ofType xsd:integer
  height ofType xsd:integer
  depth ofType xsd:integer
  manufacturer ofType xsd:string
  hasLocation ofType loc:geographicalLocation
```

WSML
FindPhoneNumber {  
  Provider: HPI.  
  Input: Name n.  
  Output: PhoneNumber p.  
  Variable: Person x.  
  Precondition | Effect  
  has(x, n) | has(x, p).  
}.  

namespace <https://asg-platform.org/findPhoneNumber#>  
com:<https://asg-platform.org/C-1/communication.wsml#>  
par:<https://asg-platform.org/C-1/participant.wsml#>  

webservice <https://asg-platform.org/findPhoneNumber.wsml>>  

nonFunctionalProperties  
dc:title hasValue "Find Phone Number Web Service"  
endNonFunctionalProperties  

importedOntologies {  
  <https://asg-platform.org/communication.wsml>>,  
  <https://asg-platform.org/participant.wsml>>  
}

capability <https://asg-platform.org/findPhoneNumber/>>  

precondition  
  axiom #  
  definedBy  
  ?name memberOf xsd:string.  

postcondition  
  axiom #  
  definedBy  
  (?telephoneNumber memberOf com:telephoneNumber) and  
  (?person.telephoneNumber = ?telephoneNumber).  

WSML
Hands-On: Part 2

FindPhoneProvider {
  Provider: Telekom.
  Cost: min: 1, max: 3.
  Input: PhoneNumber p.
  Output: Provider v.
  Precondition | Effect
               | providerof(v, p).
}
Hands-On: Part 2

FindPhoneProvider {
  Provider: Telekom.
  Cost: min: 1, max: 3.
  Input: PhoneNumber p.
  Output: Provider v.
  Precondition | Effect
    | providerof(v, p).
}.

webservice <<https://asg-platform.org/findPhoneProvider.wsml>>

  nonFunctionalProperties
    dc:title hasValue "Find Phone Provider Web Service"
    owner hasValue par:HPI
  endNonFunctionalProperties

importedOntologies {
  <<https://asg-platform.org/communication.wsml>>,
  <<https://asg-platform.org/participant.wsml>>
}

capability <<https://asg-platform.org/findFhoneProvider/>>

  precondition
    axiom _#
    definedBy
      (?telephoneNumber memberOf com:telephoneNumber).

  postcondition
    axiom _#
    definedBy
      (?telephoneProvider memberOf par:telephoneProvider) and
      (?telephoneNumber.hasProvider = ?telephoneProvider).
Hands-On: Part 3

• Part 3
  – Load domain ontologies and service descriptions to Flora
  – Run query to simulate discovery
  – Discuss discovery in context of Flora
Contents

• Part I:
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

----------- break -----------

• Part II:
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

----------- lunch -----------

• Part III:
  – Hands-on session with Flora2/XSB

----------- break -----------

• Part IV:
  – Demos: Editor, Theorem Prover, Reference Implementation
  – ASG Use Case discussion
Part IV
Demos, Tools and Discussions
Tools – Demo

- WSML Editor
  - SWWS Studio [SWWS]
- Vampire – theorem prover
- Execution framework – WSMX
- Data Mediation
SWWS Studio
developed by Ontotext Lab
(http://www.ontotext.com/)
WSMO Editor

- Create WSMO descriptions
  - Service descriptions
  - Domain ontologies
  - Goals
  - Mediators
- Exports in F-Logic, WSML and OWL
- Next release will be Eclipse plug-in using WSMO API and WSMO4J
Vampire Theorem Prover
WSMX: Reference Implementation

Open source available at:
http://www.sourceforge.net/projects/wsmx
Architecture: Compilation

ASG C1 and C2, Potsdam, Germany, December 13, 2004
WSMX Overview

- Reference architecture & implementation
- Conceptual model is WSMO
- End to end functionality for executing SWS
- Multiple execution semantics
- Open source code base at Source Forge
- Event based component architecture
WSM* and Adaptive Service Grid
WSM* and ASG

Semantic Web Services

<table>
<thead>
<tr>
<th>WSMO: Conceptual Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSML: Language</td>
</tr>
<tr>
<td>WSMX: Architecture &amp; Ref. Implementation</td>
</tr>
</tbody>
</table>
WSM* and ASG

Semantic
Web Services

<table>
<thead>
<tr>
<th>Ontologies</th>
<th>WSMO: Conceptual Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>WSML: Language</td>
</tr>
<tr>
<td></td>
<td>WSMX: Architecture &amp; Ref. Implementation</td>
</tr>
</tbody>
</table>

Event-based Framework
Discovery
Parser
WSM* and ASG

Semantic Web Services

Ontologies

WSMO: Conceptual Model

Syntax

WSML: Language

WSMX: Architecture & Ref. Implementation

ASG

Event-based Framework

Discovery

Parser
WSM* and ASG

Semantic Web Services

WSMO: Conceptual Model

Ontologies

WSML: Language

Syntax

WSMX: Architecture & Ref. Implementation

ASG

WSRF

WS-Addressing

Event-based Framework

Discovery

Parser

Negotiation

SLA

ASG C1 and C2, Potsdam, Germany, December 13, 2004
WSM* and ASG

Semantic Web Services

WSMO: Conceptual Model

Ontologies

WSML: Language

Syntax

WSMX: Architecture & Ref. Implementation

Event-based Framework

Discovery

Parser

ASG

WSRF

WS-Addressing

Negotiation

SLA

Globus Toolkit
The definition of WSRF means that Grid and Web communities can move forward on a common base.
Support from major WS vendors, especially in management space: e.g. HP, IBM.
Contents

• Part I:
  – Grid and Web Services
  – Semantic Web Services
  – WSMO Conceptual Model
  – WSML Overview, Syntax and Variants

  --------- break ---------

• Part II:
  – WSML Service and Ontology descriptions (use case example)
  – Approaches for Reasoning Support for WSML and Discovery
  – Reasoning Support using Flora2/XSB

  --------- lunch ---------

• Part III:
  – Hands-on session with Flora2/XSB

  --------- break ---------

• Part IV:
  – Demos: Editor, Theorem Prover, Reference Implementation
  – Demos: WSMX and Mediation
  – Grounding discussion
  – ASG Use Case discussion
ASG Use Case Discussion
Wrap-up
Wrap-Up

• Thanks
• WSMO/WSML a good start for ASG service description
  – Can and should be extended for missing concepts
• Reasoning now: WSML translates to F-Logic
  – WSMO API & WSMO4J libraries available
• Editor
  – SWWS Editor first cut
  – WSMO Editor based as Eclipse Plug-In under development
• WSMX a reference architecture and implementation
• Globus toolkit may be good fit for Grid functionality
References

• The central location where WSMO work and papers can be found is WSMO Working Group: http://www.wsmo.org
• In regard of WSMO languages: WSML Working Group: http://www.wsml.org
• WSMO implementation: WSMX working group can be found at: http://www.wsmx.org
• WSMX open source can be found at: https://sourceforge.net/projects/wsmx/
References

• Flora2: http://flora.sourceforge.net/
• XSB: http://xsb.sourceforge.net/
• Guizhen Yang1, Michael Kifer2 and Chang Zhao: Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web. ODBASE 2004
• SWWS: the SWWS Studio, developed by Ontotext Lab (http://www.ontotext.com/)
References


