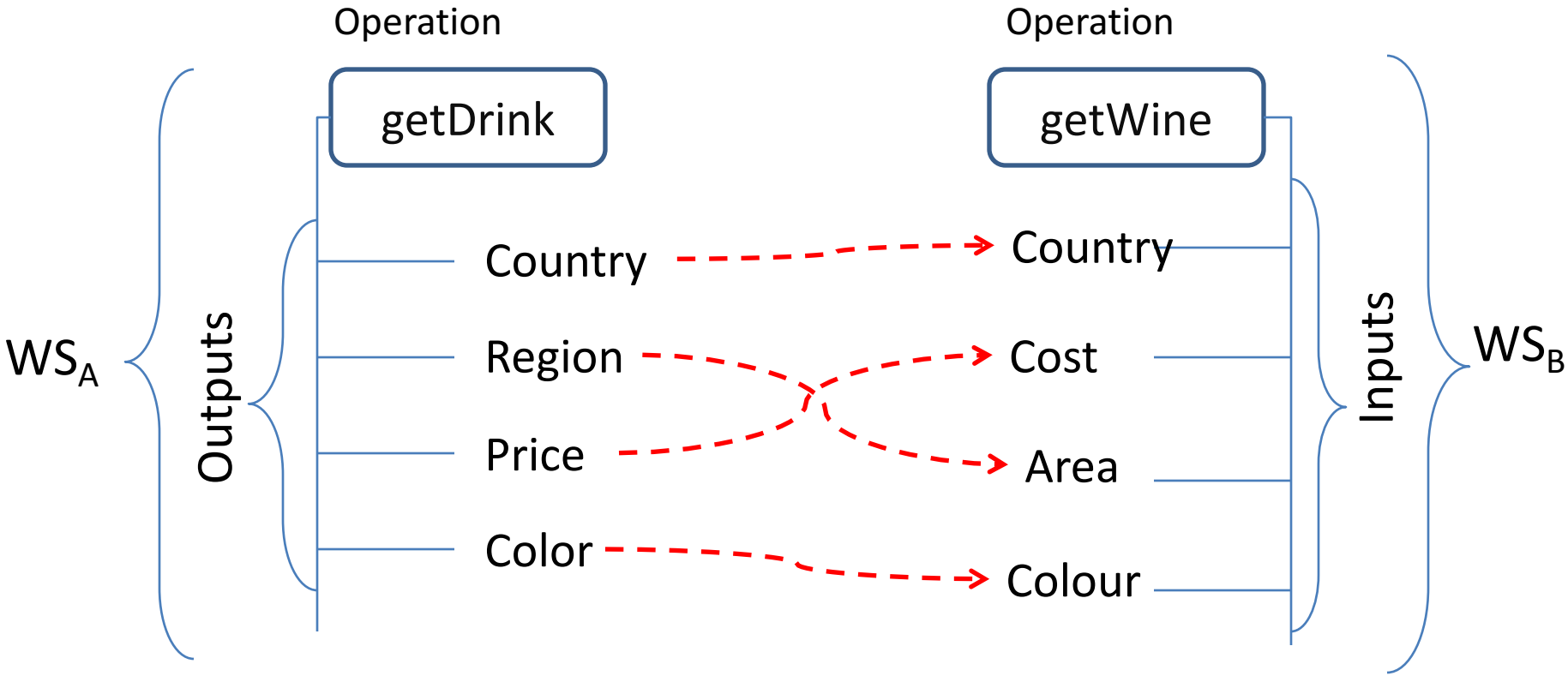
Programming Web Services
ID2208

Project

Mihhail Matskin (misha@kth.se)
Hooman Peiro Sajjad (shps@kth.se)
KTH – ICT School
VT 2017

Web Services Match-Making

- The idea is to automatically match output of an operation of a Web service with inputs of an operation of another Web service.



Web Service Matching

- The matching can be Syntactically or Semantically.
- By matching we mean, finding the following pairs:
  - <Region, Area>
  - <Price, Cost>
  - <Color, Colour>

- Only Consider basic elements (those with built-in types such as *int, double, string, date*, …) for matching
- For the time being, we are looking only at Syntactic matching.
- Then, we extend this to Semantic Matching, where we use ontology.

Syntactic Matching

- So how to do syntactic matching ?
  - Use Edit-Distance (Given two strings s_1 and s_2, the edit distance between s_1 and s_2 is the minimum number of operations required to convert string s_1 to s_2.)
    - http://www.algorithmist.com/index.php/Edit_Distance
  - Use WordNet: is a lexical database which is available online, and provides a large repository of English lexical items.
    - http://wordnet.princeton.edu/

WordNet

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: Country [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) state, nation, **country**, land, commonwealth, res publica, body politic (a politically organized body of people under a single government) *"the state has elected a new president"; "African nations"; "students who had come to the nation's capitol"; "the country's largest manufacturer"; "an industrialized land"*
- S: (n) **country**, state, land (the territory occupied by a nation) *"he returned to the land of his birth"; "he visited several European countries"*
- S: (n) nation, land, **country** (the people who live in a nation or country) *"a statement that sums up the nation's mood"; "the news was announced to the nation"; "the whole country worshipped him"*
- S: (n) **country**, rural area (an area outside of cities and towns) *"his poetry celebrated the slower pace of life in the country"*
- S: (n) area, **country** (a particular geographical region of indefinite boundary (usually serving some special purpose or distinguished by its people or culture or geography)) *"it was a mountainous area"; "Bible country"*

Your Task - 1

- Develop a program, which takes two Web services (WSDL documents) WS_1 and WS_2 as inputs and measure syntactic matching between outputs of operations of the first Web service with the inputs of operations of the second Web service.
- Put results in the format shown in Output.XML,
- We provide you the WSDL files that you need to match

Your Task - 2

- You need also the compute matching score between of operations, OP_i from WS_1 and OP_j from WS_2,

- Element Score = Edit Distance
- Operation Score = Average of all element matching scores
- Service Score = Average of all its operation scores

- Only takes into account element matching with : score > 0.8

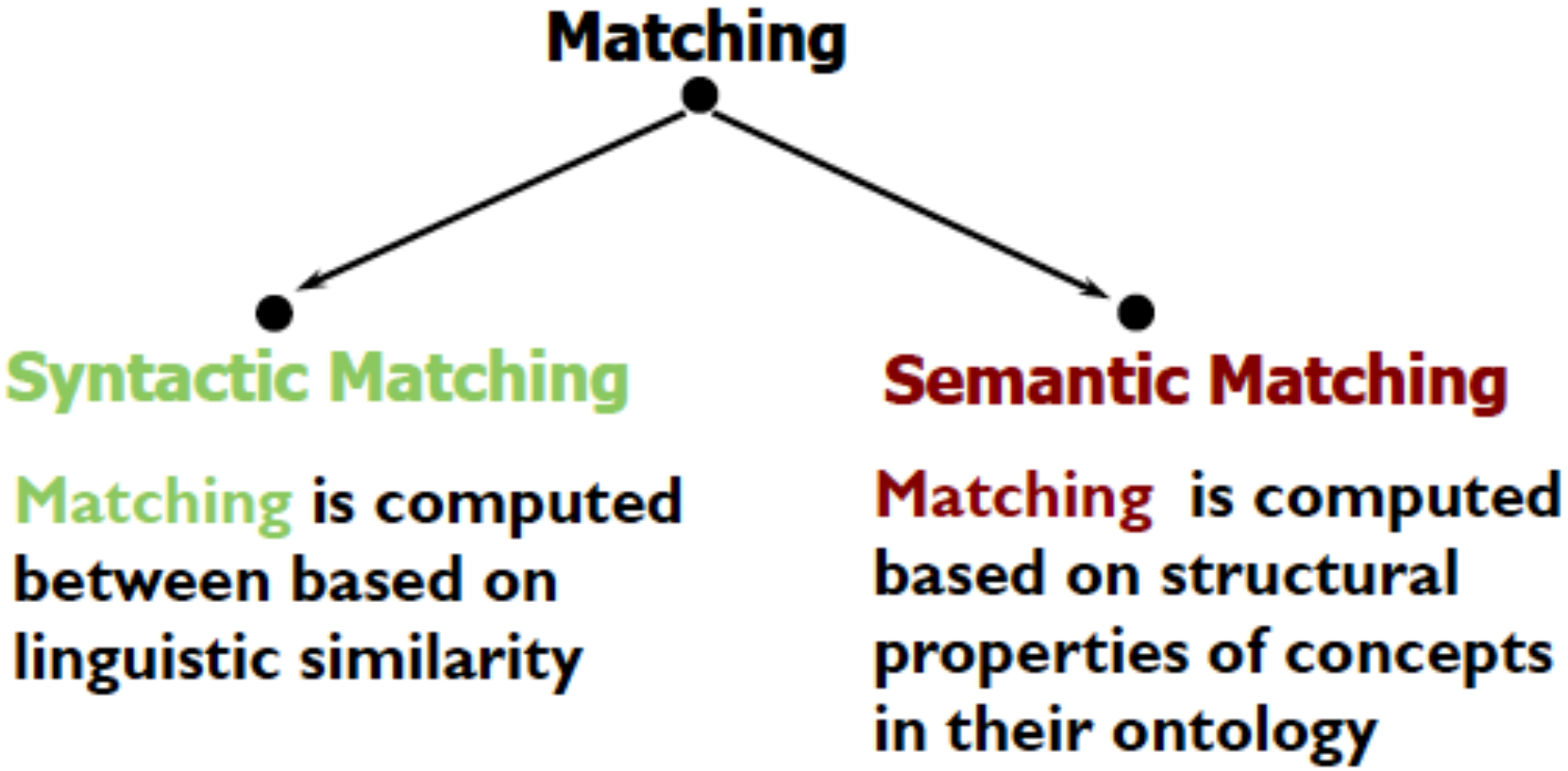
Output format

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:WSMatching xmlns:tns="http://www.kth.se/ict/id2208/Matching" xml
  <tns:Macthing>
    <tns:OutputServiceName>WS_A</tns:OutputServiceName>
    <tns:InputServiceName>WS_B</tns:InputServiceName>
    <tns:MacthedOperation>
      <tns:OutputOperationName>getDrink</tns:OutputOperationName>
      <tns:InputOperationName>getWine</tns:InputOperationName>
      <tns:OpScore>0.9375</tns:OpScore>
      <tns:MacthedElement>
        <tns:OutputElement>Country</tns:OutputElement>
        <tns:InputElement>Country</tns:InputElement>
        <tns:Score>1.0</tns:Score>
      </tns:MacthedElement>
      <tns:MacthedElement>
        <tns:OutputElement>Price</tns:OutputElement>
        <tns:InputElement>Cost</tns:InputElement>
        <tns:Score>0.9</tns:Score>
      </tns:MacthedElement>
      <tns:MacthedElement>
        <tns:OutputElement>Region</tns:OutputElement>
        <tns:InputElement>Area</tns:InputElement>
        <tns:Score>0.85</tns:Score>
      </tns:MacthedElement>
```

We provide Java code for :

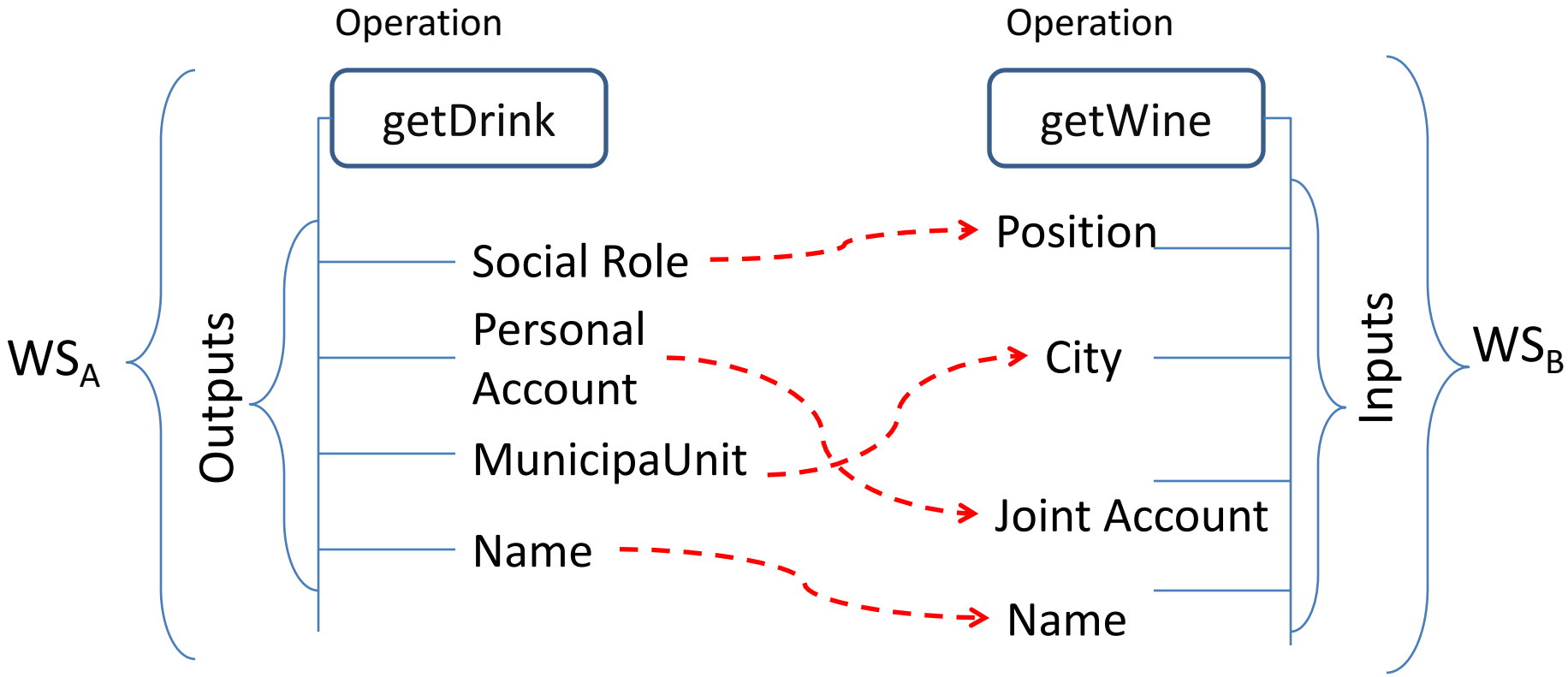
- Edit-Distance
- WSDL files to match
- XML Schema for Output file

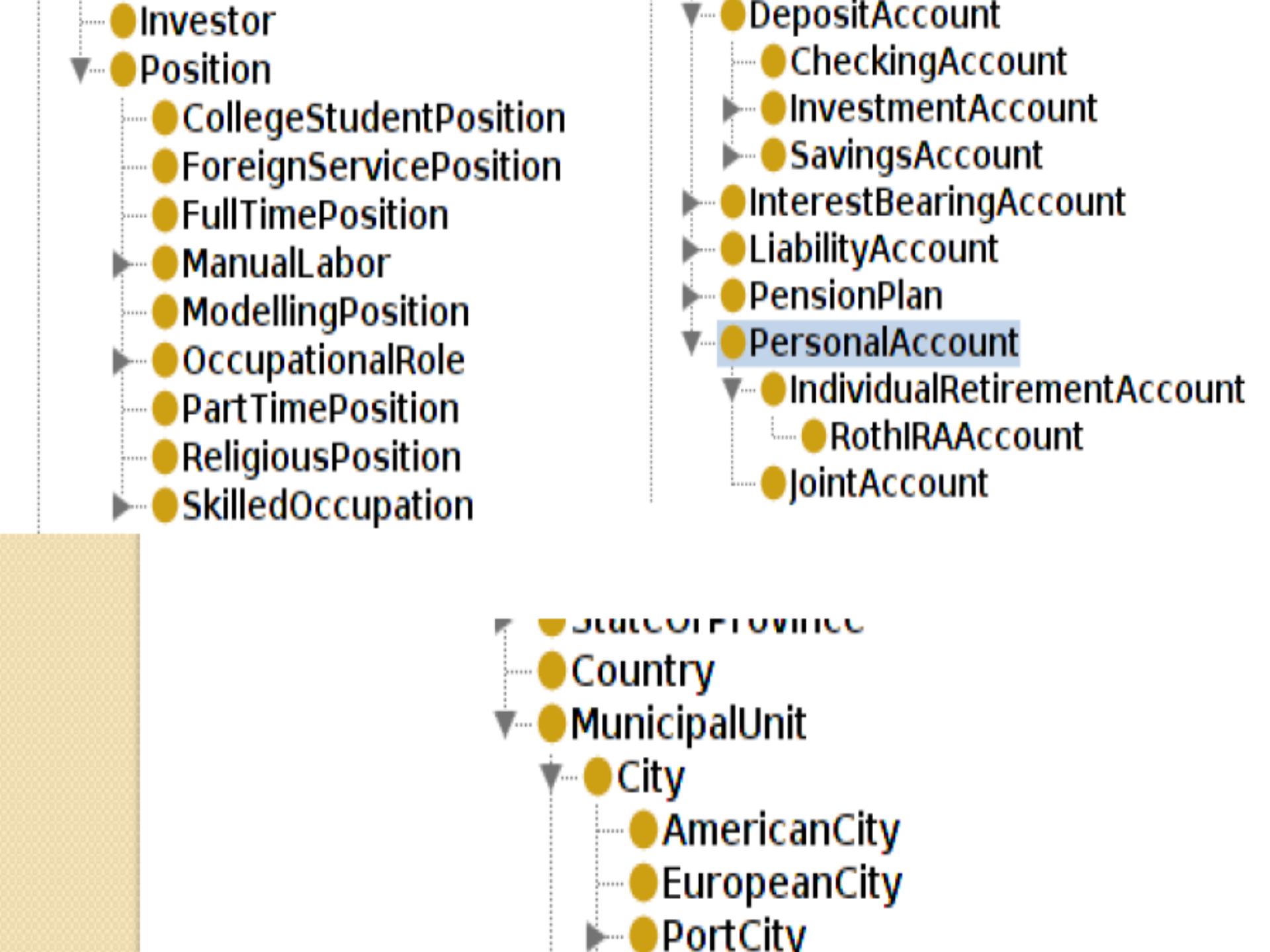
Matching: Syntactic AND Semantic



Web Services Match-Making

- The idea is to automatically match output of a Web service operation with inputs of operation of another Web service based on their semantic annotation.




Investor
Position
CollegeStudentPosition
ForeignServicePosition
FullTimePosition
ManualLabor
ModellingPosition
OccupationalRole
PartTimePosition
ReligiousPosition
SkilledOccupation
DepositAccount
CheckingAccount
InvestmentAccount
SavingsAccount
InterestBearingAccount
LiabilityAccount
PensionPlan
PersonalAccount
IndividualRetirementAccount
RothIRAAccount
JointAccount
Country
MunicipalUnit
City
AmericanCity
EuropeanCity
PortCity

Different Matching Degrees

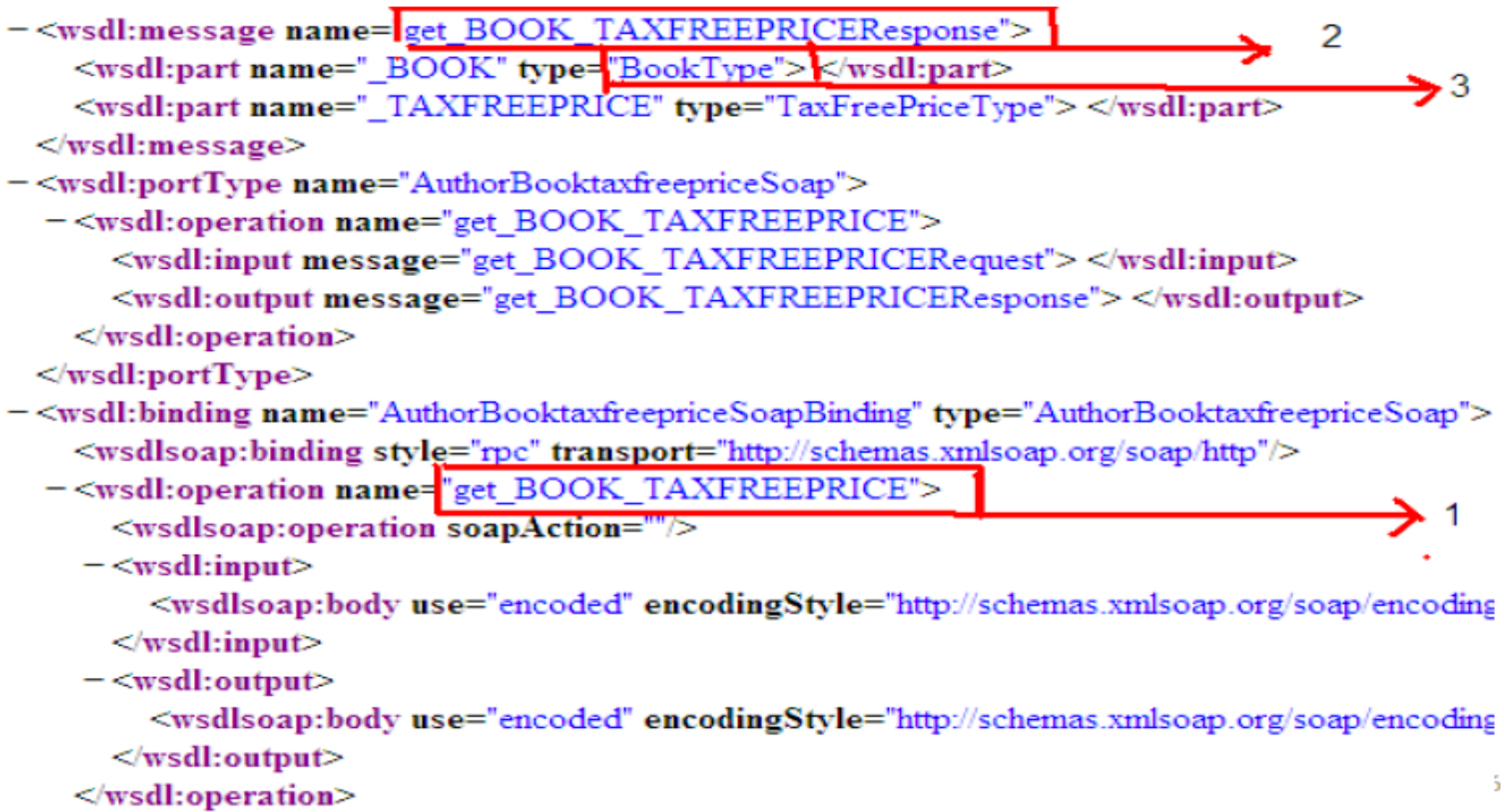
- Match C_{output} with C_{input} :

  if C_{output} isSameAs C_{input} then
  return *Exact*
  else if C_{input} isSubClassOf C_{ouput} then
  return *Subsumption*
  else if C_{output} isSubClassOf C_{input} then
  return *Plug-in*
  else if C_{output} hasRelationWith C_{input} then
  return *Structural*
  else
  return *NotMatched*
  end if

How to get Semantic Classes annotating input and outputs ?

- Using SAWSDL annotations: http://www.w3.org/TR/sawsdl/#Using

author_booktaxfreeprice_service.wsdl




```
−<wsdl:message name="get_BOOK_TAXFREEPRICEResponse">
    <wsdl:part name="_BOOK" type="BookType"></wsdl:part>
    <wsdl:part name="_TAXFREEPRICE" type="TaxFreePriceType"> </wsdl:part>
  </wsdl:message>
−<wsdl:portType name="AuthorBooktaxfreepriceSoap">
  −<wsdl:operation name="get_BOOK_TAXFREEPRICE">
      <wsdl:input message="get_BOOK_TAXFREEPRICERequest"> </wsdl:input>
      <wsdl:output message="get_BOOK_TAXFREEPRICEResponse"> </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
−<wsdl:binding name="AuthorBooktaxfreepriceSoapBinding" type="AuthorBooktaxfreepriceSoap">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  −<wsdl:operation name="get_BOOK_TAXFREEPRICE">
      <wsdlsoap:operation soapAction=""/>
    −<wsdl:input>
        <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding
      </wsdl:input>
    −<wsdl:output>
        <wsdlsoap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding
      </wsdl:output>
    </wsdl:operation>
```

How to get Semantic Classes annotating input and outputs ?

- BookType is annotated with #Book semantic class in ontology “books.owl”

```
<xsd:complexType name="BookType" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Book">
 - <xsd:sequence>
      <xsd:element name="isTitled" type="Title"/>
      <xsd:element name="hasType" type="Book-Type"/>
      <xsd:element name="writtenBy" type="AuthorType"/>
      <xsd:element name="publishedBy" type="Publisher"/>
      <xsd:element name="datePublished" type="Date"/>
      <xsd:element name="timePublished" type="Once"/>
   </xsd:sequence>
</xsd:complexType>
```

- Sometimes, constituting elements of a complex type are also annotated , see next slide.

How to get Semantic Classes annotating input and outputs ?

- Using provided SAWSDL annotations: look at *author _booktaxfreeprice_service.wsdl*

```
- <xsd:simpleType name="Title" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Title">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
- <xsd:simpleType name="Publisher" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Publisher">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
- <xsd:simpleType name="Book-Type" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Book-Type"
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
- <xsd:simpleType name="AuthorType" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Author">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>
```

How to get Semantic Classes annotating input and outputs ?

```
- <xsd:simpleType name="Title" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Title">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
- <xsd:simpleType name="Publisher" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Publisher">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
- <xsd:simpleType name="Book-Type" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Book-Type"
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
- <xsd:simpleType name="AuthorType" sawsdl:modelReference="http://127.0.0.1/ontology/books.owl#Author">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:schema>
```

- In above example semantic classes are: Title, Publisher, Book-type and Author
- Classes are located in “SUMO.owl” ontology

Extracting Semantic Classes annotating input and output elements

- Trace from operation messages to respective input outputs elements which are annotated

- If both complex-type and constituting elements are annotated only consider the higher level annotated complex type!

Matching Degrees:

- No Standard rule for degrees, so for the time being simply assume:

- Exact = 1.0
- Subsumption = 0.8
- Plug-in = 0.6
- Structural = 0.5
- Not matched = 0.0

How to find relationships: *isSameAs, subClassOf, hasStructuralRelation*

- Of course, using ontology !

- We provide some Java code to play with, but feel free to extend it or improve it to fit your requirements,
- Use Protégé 4.0 tool to see the ontology. http://protege.stanford.edu/download/protege/4.0/
- We also provide some SAWSDL annotated web services
- We also unified all ontologies used in those SAWSDL services into one Ontology(SUMO.OWL), so you can just ignore the specified ontology in the SAWSDL file, and use the unified one.

Your Task 3

- Extend the matching in the first part of the project to do semantic matching
- Find pair of Web services where the output of first matches semantically the input of another one
- Use the similar output format as you used for for Tasks 1 and 2
- Use matching threshold = 0.5

Deliverables and Deadline

- Textual report describing what did you do
- Send source code+ Instructions how to install, and run your system
- Email Subject: PWS17-Project
- Submit your deliverables in Canvas (one submission per group is fine)
- Deadline: 6 March 2017
- Presentation: TBA