# Multi-agent and Semantic Web Systems: Revision

Fiona McNeill

School of Informatics

25th March 2013

# What will be in the exam?

Potentially:

- Anything in the slides, including guest lectures
- Discussions we have had in class and the additional reading will not be necessary for adequate answers to exam questions. But answers to discursive questions which show a good awareness will score more highly - so do this if you can!

# What will be in the exam?

Potentially:

- Anything in the slides, including guest lectures
- Discussions we have had in class and the additional reading will not be necessary for adequate answers to exam questions. But answers to discursive questions which show a good awareness will score more highly - so do this if you can!

The exam will be a mixture between:

- *Bookwork:* essay*-type questions in which you discuss your understanding of a particular aspect of the course; this is where additional reading comes in especially useful.
- *Worked:* where you have to find solutions.

* This does not imply the answers should be long: one paragraph, or a few, may be sufficient.

# What will be in the exam?

Potentially:

- Anything in the slides, including guest lectures
- Discussions we have had in class and the additional reading will not be necessary for adequate answers to exam questions. But answers to discursive questions which show a good awareness will score more highly - so do this if you can!

The exam will be a mixture between:

- *Bookwork:* essay*-type questions in which you discuss your understanding of a particular aspect of the course; this is where additional reading comes in especially useful.
- *Worked:* where you have to find solutions.

The emphasis is on practical methods, ability to explain and critically evaluate.

* This does not imply the answers should be long: one paragraph, or a few, may be sufficient.

# What will be in the exam?

You should be able to do at least the following:

- Formalise domains in RDF(S), formulate and perform SPARQL queries.

# What will be in the exam?

You should be able to do at least the following:

- Formalise domains in RDF(S), formulate and perform SPARQL queries.
- Be able to read, explain, and write content in all the formats discussed in the course.

# What will be in the exam?

You should be able to do at least the following:

- Formalise domains in RDF(S), formulate and perform SPARQL queries.
- Be able to read, explain, and write content in all the formats discussed in the course.
- Compare different technologies, sketch potential system and algorithm designs at a high level.

# What will be in the exam?

You should be able to do at least the following:

- Formalise domains in RDF(S), formulate and perform SPARQL queries.
- Be able to read, explain, and write content in all the formats discussed in the course.
- Compare different technologies, sketch potential system and algorithm designs at a high level.
- Discuss the ideas and philosophies covered in this course.

# How to answer exam questions

- You will have to answer two questions:
  - Question 1 is *compulsory*
  - Then answer Question 2 *or* Question 3.
  - Spend some time deciding which of these to answer (but not too much!) and make sure you look through the whole question.

# How to answer exam questions

- You will have to answer two questions:
  - Question 1 is *compulsory*
  - Then answer Question 2 *or* Question 3.
  - Spend some time deciding which of these to answer (but not too much!) and make sure you look through the whole question.
- The marks awarded for each section give you a clue as to how much time you should devote to each: a 10-mark question should generally take around twice as much time as a 5-mark question.

# How to answer exam questions

- You will have to answer two questions:
  - Question 1 is *compulsory*
  - Then answer Question 2 *or* Question 3.
  - Spend some time deciding which of these to answer (but not too much!) and make sure you look through the whole question.

- The marks awarded for each section give you a clue as to how much time you should devote to each: a 10-mark question should generally take around twice as much time as a 5-mark question.

- As a rule of thumb, every mark allocated to a question is looking for something specific. So a good answer to a question: *Discuss some of the issues* ... should contain (at least) three issues.

# How to answer exam questions

- You are marked for what you know, not marked down for what you don't know. If in doubt, put it in.

# How to answer exam questions

- You are marked for what you know, not marked down for what you don't know. If in doubt, put it in.

- In solution-based questions, include your workings, especially if you are not sure of your answer. You may get some credit even for a wrong answer if you can demonstrate you had the right idea.

# How to answer exam questions

- You are marked for what you know, not marked down for what you don't know. If in doubt, put it in.

- In solution-based questions, include your workings, especially if you are not sure of your answer. You may get some credit even for a wrong answer if you can demonstrate you had the right idea.

- Time yourself carefully. Do not allow Question 1 to take up more than half the exam (you can always return to it if you answer the next question quickly).

# What was this course about?

- Semantic Web foundations

# What was this course about?

- Semantic Web foundations
- RDF and RDFS

# What was this course about?

- Semantic Web foundations
- RDF and RDFS
- DL and OWL

# What was this course about?

- Semantic Web foundations
- RDF and RDFS
- DL and OWL
- Matching and Meaning

# What was this course about?

- Semantic Web foundations
- RDF and RDFS
- DL and OWL
- Matching and Meaning
- Agents and Services

# What was this course about?

- Semantic Web foundations
- RDF and RDFS
- DL and OWL
- Matching and Meaning
- Agents and Services
- Small ideas vs big ideas

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning
- RDF vs. RFDS vs. OWL

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning
- RDF vs. RFDS vs. OWL
- Data, resources, meta-data and inference

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning
- RDF vs. RFDS vs. OWL
- Data, resources, meta-data and inference
- Semantic web vs. databases

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning
- RDF vs. RFDS vs. OWL
- Data, resources, meta-data and inference
- Semantic web vs. databases
- Communication: human, service, agent

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning
- RDF vs. RFDS vs. OWL
- Data, resources, meta-data and inference
- Semantic web vs. databases
- Communication: human, service, agent
- Anyone can say anything about anything

# Semantic Web foundations

- Ontologies: concepts, relations, hierarchies
- Representation: expressivity vs efficiency of reasoning
- RDF vs. RFDS vs. OWL
- Data, resources, meta-data and inference
- Semantic web vs. databases
- Communication: human, service, agent
- Anyone can say anything about anything
- How to obtain semantic metadata

# RDF and RDFS

- Fundamental languages for capturing meaning

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace
- Different notations: XML, N3, Turtle, Qnames, datatypes

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace
- Different notations: XML, N3, Turtle, Qnames, datatypes
- RDF and RDBS

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace
- Different notations: XML, N3, Turtle, Qnames, datatypes
- RDF and RDBS
- Classes, properties, instances domains, ranges

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace
- Different notations: XML, N3, Turtle, Qnames, datatypes
- RDF and RDBS
- Classes, properties, instances domains, ranges
- Inference in RDF and RDFS

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace
- Different notations: XML, N3, Turtle, Qnames, datatypes
- RDF and RDBS
- Classes, properties, instances domains, ranges
- Inference in RDF and RDFS
- XML query vs. SPARQL inference

# RDF and RDFS

- Fundamental languages for capturing meaning
- The Semantic Web layer cake, URIs, namespace
- Different notations: XML, N3, Turtle, Qnames, datatypes
- RDF and RDBS
- Classes, properties, instances domains, ranges
- Inference in RDF and RDFS
- XML query vs. SPARQL inference

# DL and OWL

- Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

# DL and OWL

- Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

- DLs are primarily concerned with *complex concept definitions*.

# DL and OWL

- Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

- DLs are primarily concerned with *complex concept definitions*.

- DL ontologies are divided into A-boxes and T-boxes.

# DL and OWL

- Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

- DLs are primarily concerned with *complex concept definitions*.

- DL ontologies are divided into A-boxes and T-boxes.

- Some DLs allow *existential* and *universal* restriction.

# DL and OWL

• Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

• DLs are primarily concerned with *complex concept definitions*.

• DL ontologies are divided into A-boxes and T-boxes.

• Some DLs allow *existential* and *universal* restriction.

• Operate under the *closed world assumption* (assume something is true unless you can prove it isn't).

# DL and OWL

• Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

• DLs are primarily concerned with *complex concept definitions*.

• DL ontologies are divided into A-boxes and T-boxes.

• Some DLs allow *existential* and *universal* restriction.

• Operate under the *closed world assumption* (assume something is true unless you can prove it isn't.

• OWL comes in three varieties: *OWL-lite*, *OWL-DL* and *OWL-full* (not covered).

# DL and OWL

• Description Logics are a family of logics; the letters in the name of a DL describes its functionality.

• DLs are primarily concerned with *complex concept definitions*.

• DL ontologies are divided into A-boxes and T-boxes.

• Some DLs allow *existential* and *universal* restriction.

• Operate under the *closed world assumption* (assume something is true unless you can prove it isn't.

• OWL comes in three varieties: *OWL-lite*, *OWL-DL* and *OWL-full* (not covered).

• OWL-lite and OWL-DL are based on DLs.

# Matching and Meaning

- Important to the Semantic Web because individuals develop their own data.

# Matching and Meaning

- Important to the Semantic Web because individuals develop their own data.

- Matching can be applied to ontologies or web service integration, etc.

# Matching and Meaning

- Important to the Semantic Web because individuals develop their own data.

- Matching can be applied to ontologies or web service integration, etc.

- Perfect matching requires full understanding of the semantics of the developer's mind ⇒ we aim for good enough matching.

# Matching and Meaning

- Important to the Semantic Web because individuals develop their own data.

- Matching can be applied to ontologies or web service integration, etc.

- Perfect matching requires full understanding of the semantics of the developer's mind ⇒ we aim for good enough matching.

- Matching can mean: *mapping, merging, alignment, translation*.

# Matching and Meaning

- Important to the Semantic Web because individuals develop their own data.

- Matching can be applied to ontologies or web service integration, etc.

- Perfect matching requires full understanding of the semantics of the developer's mind ⇒ we aim for good enough matching.

- Matching can mean: *mapping, merging, alignment, translation*.

- Off-line vs run-time; complete vs focussed; simple hierarchies vs complex ontologies.

# Agents and services

- Service-oriented vs. agent-oriented view

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language
- Choreographing/orchestrating workflows

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language
- Choreographing/orchestrating workflows
- Client-server interaction using HTTP and XML (JSON, MIME)

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language
- Choreographing/orchestrating workflows
- Client-server interaction using HTTP and XML (JSON, MIME)
- Service-orientation and P2P networks of services

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language
- Choreographing/orchestrating workflows
- Client-server interaction using HTTP and XML (JSON, MIME)
- Service-orientation and P2P networks of services
- Web services: interfaces, bindings, endpoints, types

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language
- Choreographing/orchestrating workflows
- Client-server interaction using HTTP and XML (JSON, MIME)
- Service-orientation and P2P networks of services
- Web services: interfaces, bindings, endpoints, types
- WSDL, SOAP, RPC/XML, REST

# Agents and services

- Service-oriented vs. agent-oriented view
- Agent reasoning, communication, and coordination
- The Jason agent programming language
- Choreographing/orchestrating workflows
- Client-server interaction using HTTP and XML (JSON, MIME)
- Service-orientation and P2P networks of services
- Web services: interfaces, bindings, endpoints, types
- WSDL, SOAP, RPC/XML, REST
- Semantic Web Services, Web Service Composition

# Small ideas vs. big ideas

- A recurring theme throughout the course

# Small ideas vs. big ideas

- A recurring theme throughout the course
- Folksonomies vs. ontologies

# Small ideas vs. big ideas

- A recurring theme throughout the course
- Folksonomies vs. ontologies
- Web APIs vs. "big" Web Services

# Small ideas vs. big ideas

- A recurring theme throughout the course
- Folksonomies vs. ontologies
- Web APIs vs. "big" Web Services
- LinkedData, OpenData, and Triple Stores

# Small ideas vs. big ideas

- A recurring theme throughout the course
- Folksonomies vs. ontologies
- Web APIs vs. "big" Web Services
- LinkedData, OpenData, and Triple Stores
- SPARQL inference vs. OWL

# Drop-in sessions

- Thursday, 25th April, 1200-1300
- Thursday, 18th May, 1200-1300