IN3060/IN4060 - Repetition

Jieying Chen

2021.05.21

General Information about the Exam

- Home examination
- Disclosure time: June 11 at 9:00 AM
- Submission deadline: June 11 at 1:00 PM
- Place: Inspera
- Good Internet!
- Front page of exam
  - MN
  - IN3060/IN4060(2020)

General information:

• Important messages during the exam are given directly from the course teacher on the course's semester page or in Canvas (if used in your course). It is therefore important that you check the course's semester page / Canvas room regularly.

• Your answer should reflect your own independent work and should be a result of your own learning and work effort.

• All sources of information are allowed for written home exams. If you reproduce a text from books, online articles, etc., a reference to these sources must be provided to avoid suspicions of plagiarism. This also applies if a text is translated from other languages.

• You are responsible for ensuring that your exam answers are not available to others during the exam period, neither physically nor digitally.

• Remember that your exam answers must be anonymous; do not state either your name or that of fellow students.

• If you want to withdraw from the exam, press the hamburger menu at the top right of Inspera and select "Withdraw".

Collaboration during the exam:
It is not allowed to collaborate or communicate with others during the exam. Cooperation and communication will be considered as attempted cheating. A plagiarism control is performed on all submitted exams where text similarities between answers are checked. If you use notes that have been prepared in collaboration with others before the exam, this might be detected in a plagiarism control. Such text similarities will be considered an attempt at cheating.

Use of sources:
The answer must be written with an academic standard. Read UiO's website Use of sources and citations

Cheats:
Read about what is considered cheating on UiO's website.

General Information about the Exam (cont.)

- Similar as last year, but different (time, question type)
- Scope: all lectures
- The exam consists of five sections:
  - RDF and OTTR
  - SPARQL
  - RDFS Inferences
  - DL and OWL
  - RDF and OWL semantics, SHACL

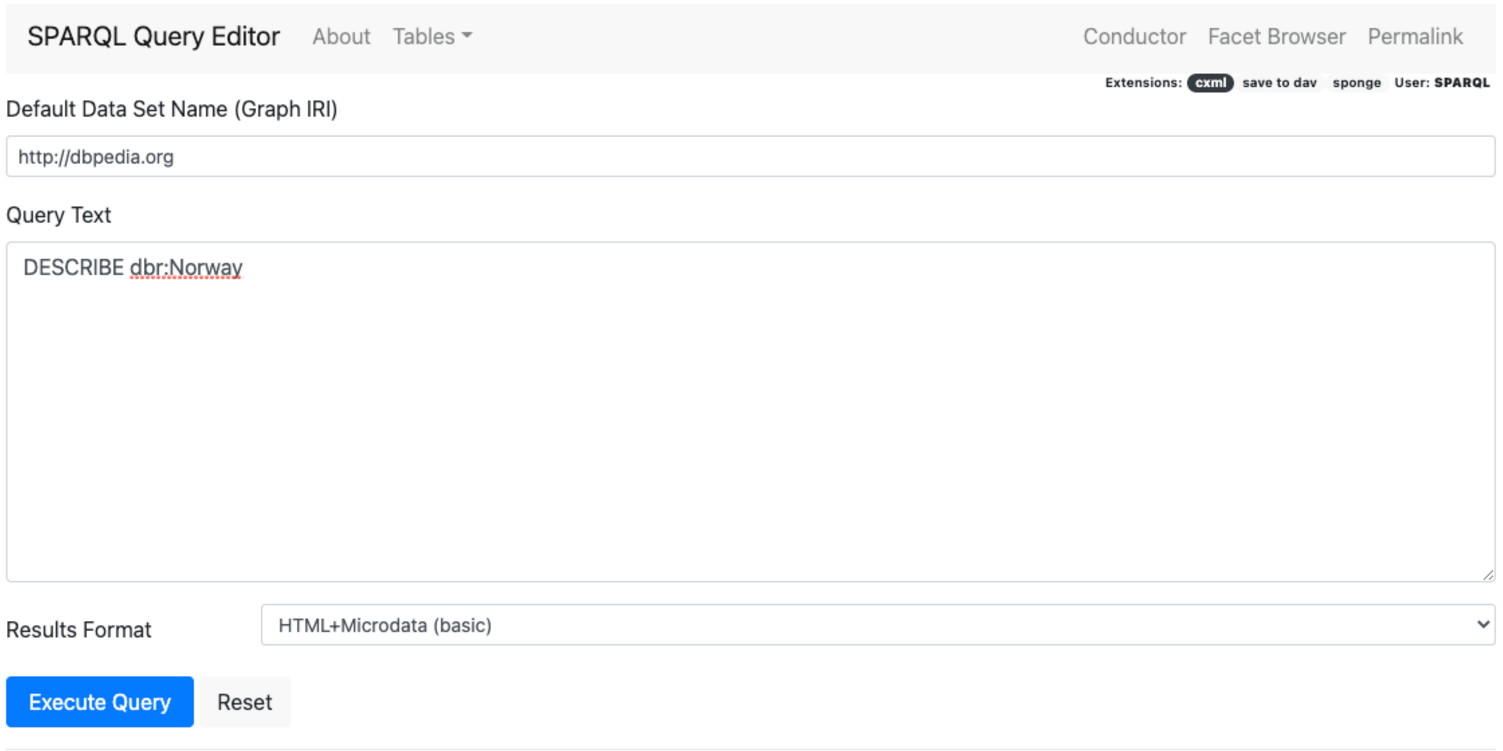
Some pratical suggestions about the Exam

- Familiar with all materials
- Try Inspera system before the exam, e.g., how to upload pdf file
- Draw RDF graph, hand draw -> take picture -> upload
- Multiple choice questions:
  - select all correct answers
  - get a negative score for wrong answers
  - but never less than 0.

Q&A

When do we use DESCRIBE?

- The simplest DESCRIBE query is just an IRI in the DESCRIBE clause:
  - DESCRIBE dbr:Norway



Result

When do we use DESCRIBE?

- The resources to be described can also be taken from the bindings to a query variable in a result set.
- For example:
  - PREFIX foaf: <http://xmlns.com/foaf/0.1/>

    DESCRIBE ?x

    WHERE { ?x foaf:name "Alice" }

  - PREFIX foaf: <http://xmlns.com/foaf/0.1/>

    DESCRIBE ?x ?y

    WHERE {?x foaf:knows ?y}

- DESCRIBE vs. CONSTRUCT: http://skkudatalab.weebly.com/sparql-describe.html

Lecture 7: Derivation vs. Entailment vs. Inference

- **Derivation**, aka formal proof, is a syntactic notion
  - is the result of deriving a conclusion from assumptions by applying **inference rules**.

- **Entailment** is a model-semantic notion.

  - A *entails* B, written $A \models B$ if
    $\mathcal{I} \models B$
    for all interpretations $\mathcal{I}$ with $\mathcal{I} \models A$

```
(1) :Account rdfs:subClassOf :BankService .
(2) :SavingsAccount rdfs:subClassOf :Account .
(3) :DebitAccount rdfs:subClassOf :Account .
(4) :hasAccount rdfs:domain :Customer .
(5) :hasAccount rdfs:range :Account .
(6) :hasAccount rdfs:subPropertyOf :hasService .
(7) :hasSavingsAccount rdfs:subPropertyOf :hasAccount .
(8) :hasDebitAccount rdfs:subPropertyOf :hasAccount .

(9) :sa rdf:type :SavingsAccount .
(10) :sandra :hasSavingsAccount :sa .
(11) :peter :hasAccount :ba .
(12) _:x :hasService :service .
```

The derivation for ":sa rdf:type :BankService ."

```
(a1) :sa rdf:type :Account . (2, 9, rdfs9)
(a2) :sa rdf:type :BankService . (a1, 1, rdfs9)
```

Lecture 8: Soundness vs. Completeness

- A inference rule or calculus is **sound**: every conclusion that can be inferred via using this inference rule or calculus is correct.
  - E.g. rdfs11,

$$\frac{A \sqsubseteq B \qquad B \sqsubseteq C}{A \sqsubseteq C}\ \text{rdfs11}$$

  - Soundness means that
    - For any choice of three classes A, B, C
    - $\{A \sqsubseteq B, B \sqsubseteq C\} \models A \sqsubseteq C$
  - Proof:
    - Let $\mathcal{I}$ be an arbitrary interpretation with $\mathcal{I} \models \{A \sqsubseteq B, B \sqsubseteq C\}$
    - Then by model semantics, $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$ and $B^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
    - By set theory, $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$
    - By model semantics, $\mathcal{I} \models A \sqsubseteq C$
    - Q.E.D.

Lecture 8: Soundness vs. Completeness

- A inference rule or calculus is **complete**, we can use this inference rule or calculus to infer all correct conclusions.

  - Two directions:
    1. If $\mathcal{A} \models \mathcal{B}$ then $\mathcal{B}$ can be derived from $\mathcal{A}$
    2. If $\mathcal{B}$ can be derived from $\mathcal{A}$ then $\mathcal{A} \models \mathcal{B}$

```
(1) :Account rdfs:subClassOf :BankService .
(2) :SavingsAccount rdfs:subClassOf :Account .
(3) :DebitAccount rdfs:subClassOf :Account .
(4) :hasAccount rdfs:domain :Customer .
(5) :hasAccount rdfs:range :Account .
(6) :hasAccount rdfs:subPropertyOf :hasService .
(7) :hasSavingsAccount rdfs:subPropertyOf :hasAccount .
(8) :hasDebitAccount rdfs:subPropertyOf :hasAccount .

(9) :sa rdf:type :SavingsAccount .
(10) :sandra :hasSavingsAccount :sa .
(11) :peter :hasAccount :ba .
(12) _:x :hasService :service .
```

RDFS entailment rules are not complete!

E.g. `:hasAccount rdfs:range :BankService` is entailed, since the range of `:hasAccount` is a subclass of `:BankService`, but it is not derivable, since there are no RDFS entailment rule deriving a range-statement.

If the calculus is sound then every statement that is entailed can be derived.

- True
- X False

If the calculus is complete then every statement that is entailed can be derived.

- X True
- False

If the calculus is sound then every statement that can be derived is entailed.

- X True
- False

If the calculus is complete then every statement that can be derived is entailed.

- True
- X False

Exam 2019: RDF and Turtle

Consider the following triples in Turtle notation:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix world: <http://world.org/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

world:Oslo a world:City ;
   world:hasMayor [foaf:name "Marianne Borgen"; foaf:gender "female"];
   world:isCityInCountry world:Norway .
world:Norway a world:Country ;
   world:hasCapital world:Oslo ;
   world:hasPopulation "5258000"^^xsd:int ;
   world:hasName "Norway"@en , "Norge"@no .
```

- **How many of these triples have an object of datatype xsd:string?** ☐

Exam 2019: RDF and Turtle

https://www.w3.org/TR/turtle/

7.2 RDF Term Constructors

This table maps productions and lexical tokens to `RDF terms` or components of `RDF terms` listed in section 7. Parsing:

production	type	procedure
IRIREF	IRI	The characters between "<" and ">" are taken, with the numeric escape sequences unescaped, to form the unicode string of the IRI. Relative IRI resolution is performed per Section 6.3.
PNAME_NS	prefix	When used in a prefixID or sparqlPrefix production, the `prefix` is the potentially empty unicode string matching the first argument of the rule is a key into the namespaces map.
	IRI	When used in a PrefixedName production, the `iri` is the value in the namespaces map corresponding to the first argument of the rule.
PNAME_LN	IRI	A potentially empty prefix is identified by the first sequence, `PNAME_NS`. The namespaces map `MUST` have a corresponding `namespace`. The unicode string of the IRI is formed by unescaping the reserved characters in the second argument, `PN_LOCAL`, and concatenating this onto the `namespace`.
STRING_LITERAL_SINGLE_QUOTE	lexical form	The characters between the outermost ""s are taken, with numeric and string escape sequences unescaped, to form the unicode string of a lexical form.
STRING_LITERAL_QUOTE	lexical form	The characters between the outermost "'s are taken, with numeric and string escape sequences unescaped, to form the unicode string of a lexical form.
STRING_LITERAL_LONG_SINGLE_QUOTE	lexical form	The characters between the outermost """s are taken, with numeric and string escape sequences unescaped, to form the unicode string of a lexical form.
STRING_LITERAL_LONG_QUOTE	lexical form	The characters between the outermost """"s are taken, with numeric and string escape sequences unescaped, to form the unicode string of a lexical form.
LANGTAG	language tag	The characters following the `@` form the unicode string of the language tag.
RDFLiteral	literal	The literal has a lexical form of the first rule argument, `String`. If the `'^^' iri` rule matched, the datatype is `iri` and the literal has no language tag. If the `LANGTAG` rule matched, the datatype is `rdf:langString` and the language tag is `LANGTAG`. If neither matched, the datatype is `xsd:string` and the literal has no language tag.

Exam 2019: RDF and Turtle

https://www.w3.org/TR/turtle/

LANGTAG	language tag	The characters following the `@` form the unicode string of the language tag.
RDFLiteral	literal	The literal has a lexical form of the first rule argument, `String`. If the `'^^' iri` rule matched, the datatype is `iri` and the literal has no language tag. If the `LANGTAG` rule matched, the datatype is `rdf:langString` and the language tag is `LANGTAG`. If neither matched, the datatype is `xsd:string` and the literal has no language tag.

- The literal has a lexical form of the first rule argument, String. If the '^^' iri rule matched, the datatype is iri and the literal has no language tag.
- If the LANGTAG rule matched, the datatype is rdf:langString and the language tag is LANGTAG.
- If neither matched, the datatype is xsd:string and the literal has no language tag.

Lecture 13: When do we want to use SHACL? How do we use it in practice?

Given the following SHACL shape

```
@prefix world:   <http://sws.ifi.uio.no/ont/world.owl#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .

world:CityShape a sh:NodeShape;
   sh:targetClass world:City;
   sh:property [
      sh:path world:isCityInCountry ;
      sh:minCount 1;
      sh:maxCount 1;
      sh:nodeKind sh:IRI ;
      sh:class world:Country ;
   ] .
```

And the following data:

```
:Paris a world:City ;
    world:hasName "Paris"@fr ;
    world:isCityInCountry :France .
:France world:hasName "France"@fr .
```

There will be an error when validating the RDF graph with respect to the SHACL shape.

a) Explain why there is a validation error, and how the error can be fixed.

Lecture 7: How to construct a counter-example

- To show that $\mathcal{A} \models T$ does *not* hold:
  - Describe an interpretation $\mathcal{I}$ (using your fantasy)
  - Prove that $\mathcal{I} \models \mathcal{A}$ (using the semantics)
  - Prove that $\mathcal{I} \not\models T$ (using the semantics)

- $\mathcal{A}$ as before:

$$\mathcal{A} = \{\mathit{loves}(\mathit{romeo}, \mathit{juliet}),\ \mathit{Lady}(\mathit{juliet}),\ \mathit{Lady} \sqsubseteq \mathit{Person},\ \mathit{loves} \sqsubseteq \mathit{knows},\ \mathrm{dom}(\mathit{loves}, \mathit{Lover}),\ \mathrm{rg}(\mathit{loves}, \mathit{Beloved})\}$$

- Does $\mathcal{A} \models \mathit{Lover} \sqsubseteq \mathit{Beloved}$?

  - Given an interpretation $\mathcal{I}$, define $\models$ as follows:
  - $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
  - $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
  - $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y\rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
  - $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y\rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$

Lecture 7: How to construct a counter-example

- A *DL-interpretation* $\mathcal{I}$ consists of
  - A set $\Delta^{\mathcal{I}}$, called the *domain* (sorry!) of $\mathcal{I}$
  - For each individual URI i, an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
  - For each class URI C, a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
  - For each property URI r, a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

  - Given an interpretation $\mathcal{I}$, define $\models$ as follows:
  - $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
  - $\mathcal{I} \models r \sqsubseteq s$ iff $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
  - $\mathcal{I} \models \mathrm{dom}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $x \in C^{\mathcal{I}}$
  - $\mathcal{I} \models \mathrm{rg}(r, C)$ iff for all $\langle x, y \rangle \in r^{\mathcal{I}}$, we have $y \in C^{\mathcal{I}}$

Lecture 7: How to construct a counter-example

Given the following OWL axioms:

(1) Cat and Dog SubClassOf Nothing ($Cat \sqcap Dog \sqsubseteq \bot$)

(2) Pet SubClassOf Cat or Dog ($Pet \sqsubseteq Cat \sqcup Dog$)

These axioms do not entail the following

(3) Cat SubClassOf Pet ($Cat \sqsubseteq Pet$)

(a) The following is a faulty attempt at a countermodel:

$$
\begin{aligned}
\Delta^I &= \{meow, arf, roar\} \\
HouseCat^I &= \{meow\} \\
Tiger^I &= \{roar\} \\
Cat^I &= \{HouseCat, Tiger\} \\
Dog^I &= \{arf\} \\
Pet^I &= \{HouseCat, Dog\}
\end{aligned}
$$

Explain what is wrong with this!

Universal restriction vs. Existential restriction

Interpretation of concept descriptions

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{for all } b, \text{ if } \langle a, b\rangle \in R^{\mathcal{I}} \text{ then } b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \text{there is a } b \text{ where } \langle a, b\rangle \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}
\end{aligned}
$$

Kahoot! Quiz

Monotonicity

- Assume $\mathcal{A} \models \mathcal{B}$
- Now add information to $\mathcal{A}$, i.e. $\mathcal{A}' \supseteq \mathcal{A}$
- Then $\mathcal{B}$ is still entailed: $\mathcal{A}' \models \mathcal{B}$
- We say that RDF/RDFS entailment is *monotonic*
- What would non-monotonic reasoning be like?
  - $\{Bird \sqsubseteq CanFly, Bird(tweety)\} \models CanFly(tweety)$
  - $\{\ldots, Penguin \sqsubseteq Bird, Penguin(tweety), Penguin \sqsubseteq \neg CanFly\} \not\models CanFly(tweety)$
  - Interesting for human-style reasoning
  - Hard to combine with semantic web technologies

If someone is responsible for a project, then he/she is also assigned to the project.

106

46
Answers

▲ $Person \sqcap \exists isResponsibleForProject.Project \sqsubseteq Person \sqcap \exists isAssignedToProject.Project$

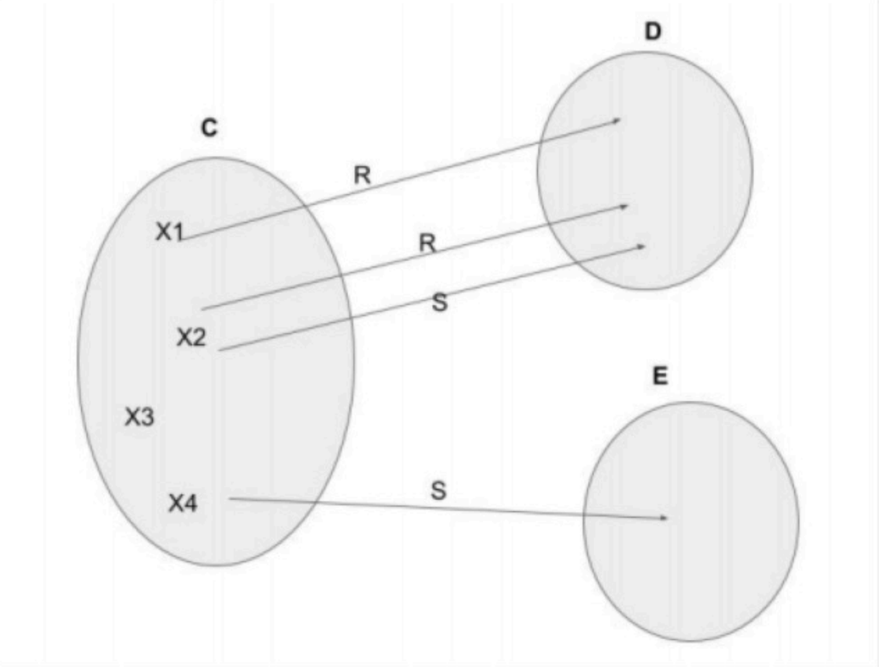
◆ $isResponsibleForProject \sqsubseteq isAssignedToProject$

● $Person \sqcap \forall isResponsibleForProject.Project \sqsubseteq Person \sqcap \forall isAssignedToProject.Project$

■ None of the above answers is correct.

Which of the following statements does NOT hold in the given interpretation?

102



47
Answers

▲ $C \sqsubseteq \forall R.D$

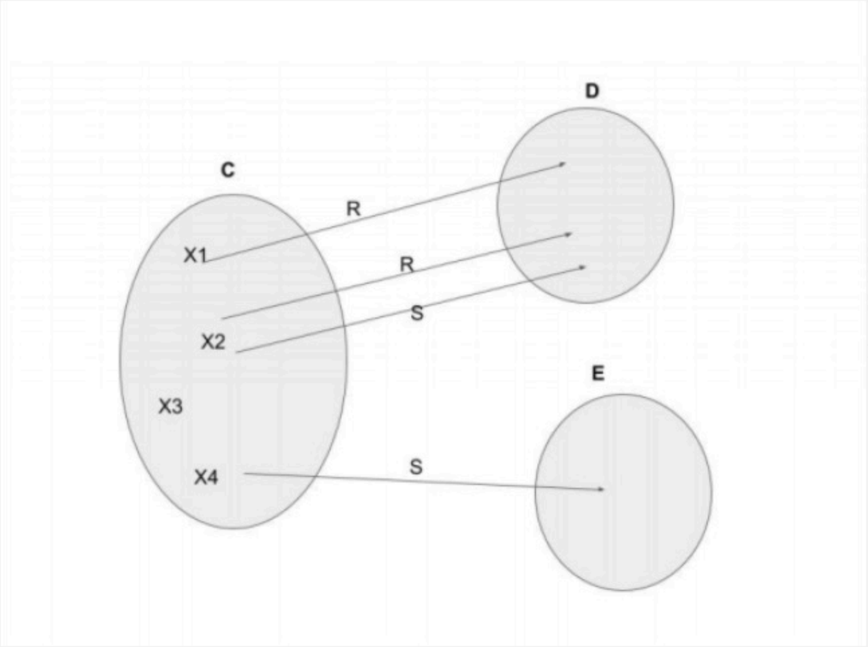
◆ $C \sqsubseteq \forall S.(D \sqcup E)$

● $C \sqsubseteq \forall R.(D \sqcup E)$

■ $C \sqsubseteq \forall S.(D \sqcap E)$

Which of the following statements is NOT correct w.r.t. the given interpretation?

36



211
Answers

▲ $X3 \in (\exists R.\ D)^I$

◆ $X3 \in (\forall R.\ D)^I$

● $X4 \in (\exists S.\ E)^I$

■ $X4 \in (\forall R.\ D)^I$