INF3580 – Semantic Technologies – Spring 2010

Lecture 11: Foundations, repetition

Audun Stolpe

20th April 2010

Department of Informatics

University of Oslo

Today's Plan

1. Basic notions
2. Semantics
3. Walkthroughs
4. Recalling soundness and completeness

Outline

Sets

Definition

- *A set is a finite or infinite collection of objects called elements of the set, considered exclusively in terms of membership. That is:*
  - *the ordering of elements doesn't matter*
  - *the number of occurrences of an element doesn't matter*

Sets

Definition

- *A set is a finite or infinite collection of objects called elements of the set, considered exclusively in terms of membership. That is:*
  - *the ordering of elements doesn't matter*
  - *the number of occurrences of an element doesn't matter*

Extensionality

- Two sets A and B are equal, $A = B$, if and only if they contain the same elements (in any order, any number of times)

Sets

Definition

- *A set is a finite or infinite collection of objects called elements of the set, considered exclusively in terms of membership. That is:*
  - *the ordering of elements doesn't matter*
  - *the number of occurrences of an element doesn't matter*

Extensionality

- Two sets A and B are equal, $A = B$, if and only if they contain the same elements (in any order, any number of times)

Notation

- The object a is/is not an element in A: $a \in A$, $a \notin A$
- E. g. the set of natural numbers from 1 to 4 inclusive: $\{1, 2, 3, 4\}$

Set-builder notation, cardinality

Set-builders

- Construct sets by restricting other sets
- Correspond to definitions *"the set of all elements $a \in A$ such that ..."*
- Is usually written $\{a \in A |\ \text{restriction on } a\}$ (expect variation)
- Example: $\{i \in \mathbb{Z} | \, i < 0\} = \{\dots, -2, -3, -1\}$

Set-builder notation, cardinality

Set-builders

- Construct sets by restricting other sets
- Correspond to definitions *"the set of all elements $a \in A$ such that ..."*
- Is usually written $\{a \in A|$ restriction on $a\}$ (expect variation)
- Example: $\{i \in \mathbb{Z}| \, i < 0\} = \{\dots, -2, -3, -1\}$

Cardinality

The size of a set A is called its cardinality. It is usually denoted $|A|$ or $\sharp A$. For instance

- $\sharp\{a, b, c\} = |\{a, b, c\}| = 3$
- $\sharp\{a, b, d, a, c, b\} = \sharp\{d, c, b, b, a\} = \sharp\{a, b, c, d\} = 4$

The inclusion exclusion principle: $|A \cup B| = |A| + |B| - |A \cap B|$

Families of sets, singleton sets, the empty set

Families of sets

Sets can be elements of other sets (given that its not the very same set):

- $\{\{\ldots, -3, -2, -1\}, \{0\}, \{1, 2, 3, \ldots\}\}$
- $\{\{1, 3, 5 \ldots\}, \{2, 4, 6, \ldots\}\}$

Singletons

A set that contains exactly one element is called a singleton

- $\{a\}$ is a singleton
- $\{\{a\}\}$ is a singleton
- $\{b, b\}$ is a singleton

Two distinguished sets

The universal set

The *universal set* is the sum total of objects that are assumed to exist relative to a given problem. We shall denote it Δ. The assumption is that:

- $A \subseteq \Delta$ for all sets A

The empty set

The empty set is the unique set without elements. It is denoted $\emptyset$ or simple $\{\}$. The empty set *is* a set, and

- $\emptyset \subseteq A$ for all A

Some examples

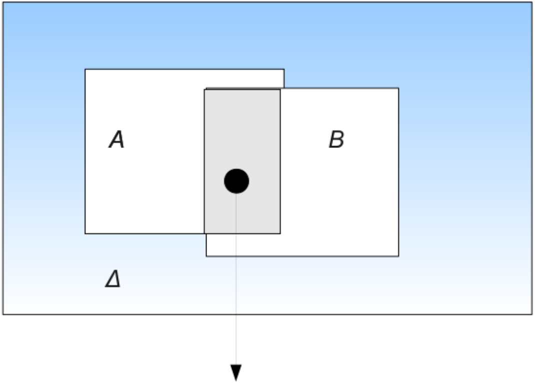
Equalities and non-equalities

- Some basic equalities:

$$\begin{aligned}\{a, b, c\} &= \{a, a, b, c\}\\ &= \{b, c, a\}\\ &= \{c, a, b, b\}\end{aligned}$$

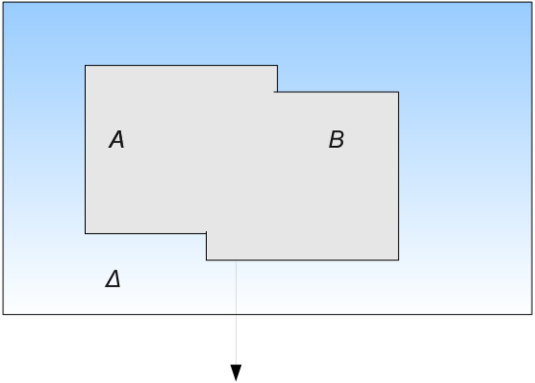
- Equalities involving set-builders:
  - $\{2k+1 |\, k \in \mathbb{N}\} = \{3, 5, 7, 9, 11 \ldots\}$
  - $\{\{0\}, \{1\}, \{2\}, \ldots\} = \{\{n\} |\, n \in \mathbb{N}\}$.
  - $\{\{0\}, \{0, 1\}, \{0, 1, 2\}, \ldots\} = \{\{m |\, 0 \leq m \leq n\} |\, n \in \mathbb{N}\}$.
- Non-equalities:
  - $\{a, b, c\} \neq \{a, b\} \neq \{a, b, d\}$
  - $\emptyset \neq \{\emptyset\}$
  - $\{b, b\} \neq \{\{b\}\}$

Operations on sets: Intersection



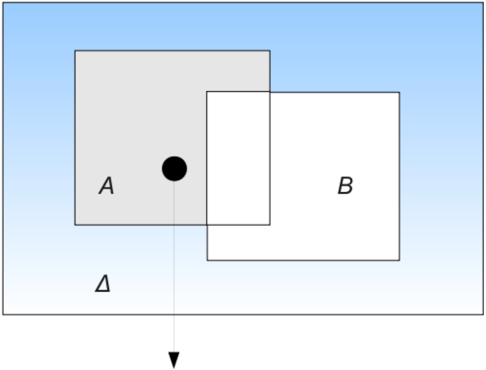
$$A \cap B = \{a \in \Delta | \, a \in A \,\&\, a \in B\}$$

Operations on sets: Union



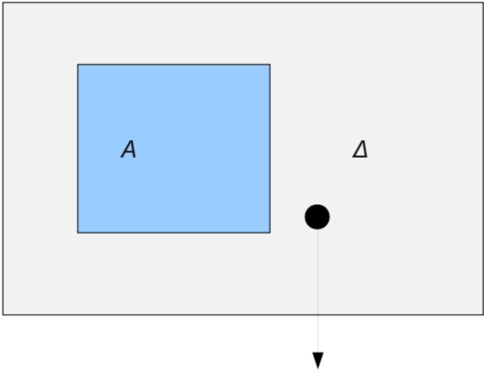
$$A \cup B = \{a \in \Delta \mid a \in A \text{ or } a \in B\}$$

Operations on sets: Relative complement/difference



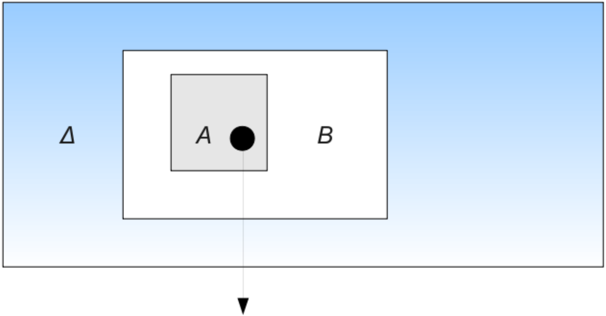
$$A - B = \{a \in \Delta | \, a \in A \; \& \; a \notin B\}$$

Operations on sets: Absolute complement



$$-A = \{a \in \Delta |\, a \notin A\}$$

Relations between sets: Subsumption



$A \subseteq B$ *iff* $a \in A$ *implies* $a \in B$

The algebra of sets

Associativity:

$$A \cup (B \cup C) = (A \cup B) \cup C \qquad A \cap (B \cap C) = (A \cap B) \cap C$$

Commutativity:

$$A \cup B = B \cup A \qquad A \cap B = B \cap A$$

Units and zeros:

$$A \cup \emptyset = A \quad A \cup \Delta = \Delta \quad A \cap \Delta = A \quad A \cap \emptyset = \emptyset$$

Idempotence:

$$A \cup A = A \qquad A \cap A = A$$

Distribution:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \; A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Complementation:

$$A \cup -A = \Delta \qquad -\Delta = \emptyset \qquad -(-A) = A$$

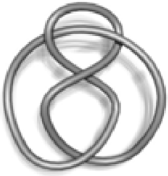
$$A \cap -A = \emptyset \qquad -\emptyset = \Delta$$

De Morgan's Laws:

$$-(A \cup B) = -A \cap -B \qquad -(A \cap B) = -A \cup -B$$

Go figure

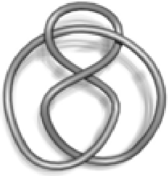
From this meager framework comes very surprising things, e.g.



Go figure

From this meager framework comes very surprising things, e.g.

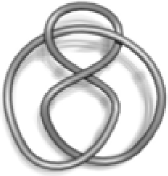
- That infinity comes in different sizes



Go figure

From this meager framework comes very surprising things, e.g.

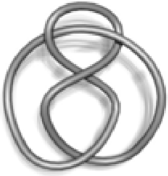
- That infinity comes in different sizes
- that an infinite set can have a *proper* subset of equal size



Go figure

From this meager framework comes very surprising things, e.g.

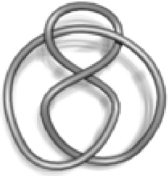
- That infinity comes in different sizes
- that an infinite set can have a *proper* subset of equal size
- that there are just as many points along a line as in a plane



Go figure

From this meager framework comes very surprising things, e.g.

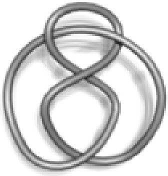
- That infinity comes in different sizes
- that an infinite set can have a *proper* subset of equal size
- that there are just as many points along a line as in a plane
- that some sets cannot be counted, even in principle



Go figure

From this meager framework comes very surprising things, e.g.

- That infinity comes in different sizes
- that an infinite set can have a *proper* subset of equal size
- that there are just as many points along a line as in a plane
- that some sets cannot be counted, even in principle
- anyway .... back to topic



Pairs and products

Ordered pair

An *ordered pair* is an object of the form (a, b) where a is an element of some set A and b is an element of some set B.

- The pair is ordered in the sense that $(a, b) \neq (b, a)$ unless $a = b$.
- It follows that $(a, b) \neq \{a, b\}$

Cartesian product

The set of all ordered pairs (a, b) where $a \in A$ and $b \in B$ is called the *Cartesian product* of A and B. It is written $A \times B$.

- $A \times B = \{(a, b) | \, a \in A \,\&\, b \in B\}$

Relations

Binary relation

A binary relation R between two sets A and B is a subset of the Cartesian product $A \times B$. In the special case that $A = B$ we say that R is a relation on A.

Notation

That x is R-related to y may be written

1. $(x, y) \in R$
2. $R(x, y)$
3. xRy

We may regard 2 and 3 as syntactical sugar for 1.

Properties of relations

Some very common properties

A relation R on a set A is

Reflexive when $(x, x) \in R$ for all $x \in A$.

Symmetric if $(x, y) \in R$ whenever $(y, x) \in R$ for all $x, y \in A$

Transitive if $(x, z) \in R$ whenever $(x, y), (y, z) \in R$ for all $x, y, z \in A$

Asymmetric if $(y, x) \in R$ and $(x, y) \in R$ is true of no $x, y \in A$.

... there are many more

A comprehensive list of OWL-supported properties was given in lecture 9.

Some operations on relations

Inverse

Let R be a binary relation on Δ. The *inverse of R* is:

$$R^{-1} = \{(b, a) : (a, b) \in R\}$$

Composition

Let R and S be binary relations on Δ. The *composition of R and S* is:

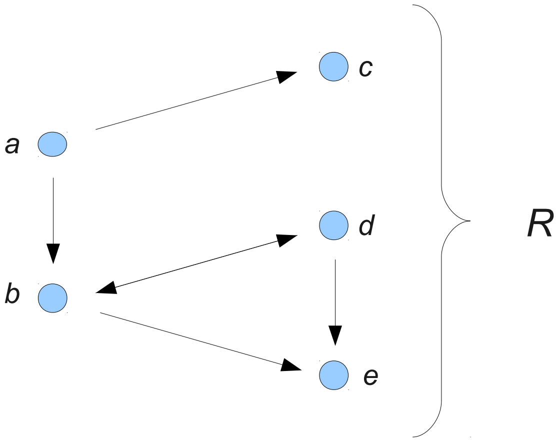
$$R \circ S = \{(a, c) : (a, b) \in R \textit{ and } (b, c) \in S \textit{ for some } b \in \Delta\}$$

Image formation

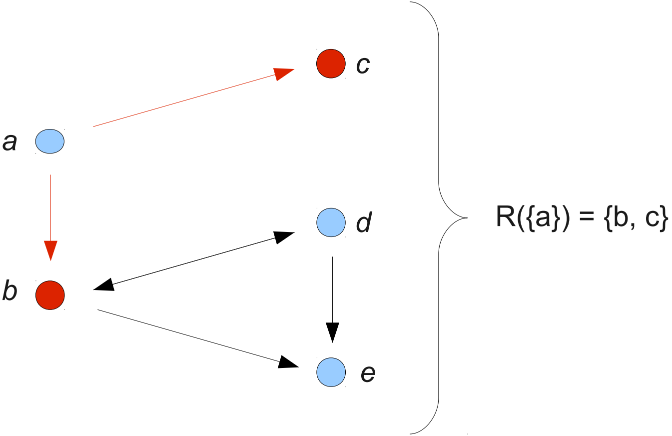
Let R a binary relation on Δ and $A \subseteq \Delta$. The *image* of R under A, is:

$$R(A) = \{b \in \Delta : (a, b) \in R \textit{ and } a \in A\}$$

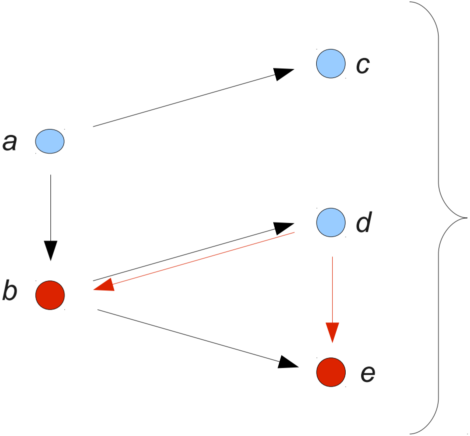
Some images



Some images

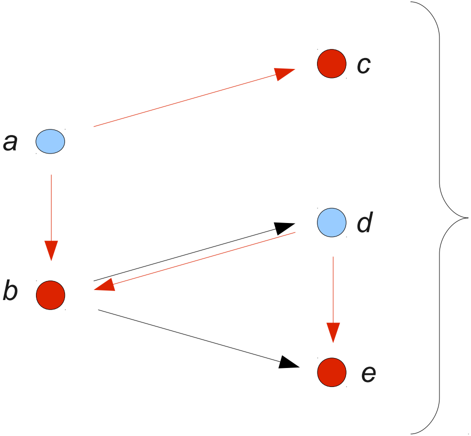


Some images



R({d}) = {b, e}

Some images



R({d, a}) = R({a}) U R({d})
= {b, c, e}

Functions

Definition

A function f from a set A to a set B is a special kind of binary relation in which every element of A is associated with a unique element of B. In other words:

- *For every $a \in A$ there is precisely one pair of the form $(a, b) \in f$*
- *stated differently, if $(a, b) \in f$ and $(a, c) \in f$ then $b = c$*

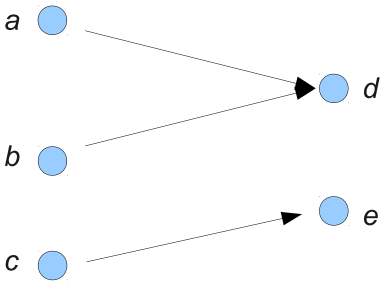
Notation

It is common to write $(a, b) \in f$ as

- $f(a) = b$, or
- $a^f = b$

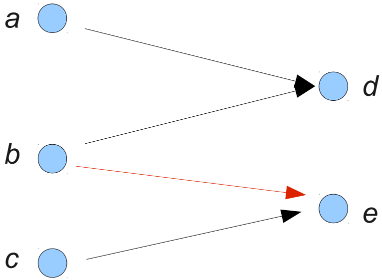
We think of f as being *applied* to the argument a.

A function and a non-function



A function f from $\{a, b, c\}$ to $\{d, e\}$

A function and a non-function



A *relation but not a function* from {*a, b, c*} to {*d, e*}

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

- they are also called *interpretations*

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

- they are also called *interpretations*
- they *interpret* a formal language in terms of objects, sets and relations

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

- they are also called *interpretations*
- they *interpret* a formal language in terms of objects, sets and relations
- that is, interpretations assign *meanings* to linguistic entities, e.g.:

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

- they are also called *interpretations*
- they *interpret* a formal language in terms of objects, sets and relations
- that is, interpretations assign *meanings* to linguistic entities, e.g.:
  - objects to names

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

- they are also called *interpretations*
- they *interpret* a formal language in terms of objects, sets and relations
- that is, interpretations assign *meanings* to linguistic entities, e.g.:
  - objects to names
  - sets of objects to concepts

The function of functions

Functions may be said to describe processes (broadly conceived) whereby

- the elements of one set are *transformed* into those of another

The models of model-theoretic semantics are functions

- they are also called *interpretations*
- they *interpret* a formal language in terms of objects, sets and relations
- that is, interpretations assign *meanings* to linguistic entities, e.g.:
  - objects to names
  - sets of objects to concepts
  - relations to predicates

Outline

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Interpretations thus assign a meaning to all *simple non-logical symbols*

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Interpretations thus assign a meaning to all *simple non-logical symbols*

- however, there are also *complex* relations and classes as well

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Interpretations thus assign a meaning to all *simple non-logical symbols*

- however, there are also *complex* relations and classes as well
- so this does not in general suffice to interpret *arbitrary formulae*

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Interpretations thus assign a meaning to all *simple non-logical symbols*

- however, there are also *complex* relations and classes as well
- so this does not in general suffice to interpret *arbitrary formulae*
- we need in addition to say

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Interpretations thus assign a meaning to all *simple non-logical symbols*

- however, there are also *complex* relations and classes as well
- so this does not in general suffice to interpret *arbitrary formulae*
- we need in addition to say
  - how the meaning of a complex expression ....

Revisiting $\mathcal{ALCQ}$ semantics

Interpretation

An interpretation $\mathcal{I}$ fixes a set $\Delta^{\mathcal{I}}$, the *domain*, and

- $A^{\mathcal{I}} \subseteq \Delta$ for each atomic concept A,
- $R^{\mathcal{I}} \subseteq \Delta \times \Delta$ for each role R, and
- $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each name a.

Interpretations thus assign a meaning to all *simple non-logical symbols*

- however, there are also *complex* relations and classes as well
- so this does not in general suffice to interpret *arbitrary formulae*
- we need in addition to say
  - how the meaning of a complex expression ....
  - depends on the meaning of its simple parts

Interpretation of complex $\mathcal{ALCQ}$ concepts

Interpretation of concept descriptions

$$
\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \mathit{if}\ (a,b) \in R^{\mathcal{I}}\ \mathit{then}\ b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \mathit{there\ is\ a}\ b \in \Delta^{\mathcal{I}} \mathit{s.t.} (a,b) \in R^{\mathcal{I}} \mathit{and}\ b \in C^{\mathcal{I}}\} \\
(\geq_n R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid \sharp\{b \mid (a,b) \in R^{\mathcal{I}}\ \mathit{and}\ b \in C^{\mathcal{I}}\} \geq n\}
\end{array}
$$

Notational variants

$$
\begin{array}{rcl}
(\forall R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(a) \subseteq C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid R^{\mathcal{I}}(a) \cap C^{\mathcal{I}} \neq \emptyset\} \\
(\geq_n R.C)^{\mathcal{I}} & = & \{a \in \Delta^{\mathcal{I}} \mid |R^{\mathcal{I}}(a) \cap C^{\mathcal{I}}\}| \geq n\}
\end{array}
$$

The form of DL/OWL ontologies

TBox and ABox formulae

An $\mathcal{ALCQ}$ knowledge base consists of two kinds of formulae

Subsumption axioms:

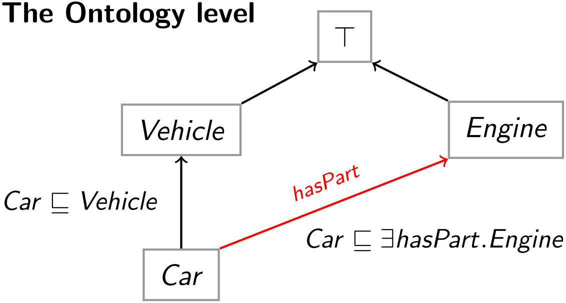
- Are of the form $C \sqsubseteq D$ (where C and D are concepts)
- model general relationships
- belong to the ontological level or the TBox

Assertions:

- Are of the form $C(a)$ or $R(a, b)$
- where C is a concept, and R a role
- describe facts
- belong to the dataset or the ABox

Connection with OWL ontologies

The Ontology level



The data level

```
Car(myBeetle)
Engine(theEngine)
hasPart(myBeetle, theEngine)
```

Satisfation/truth

Satisfaction/truth

Subsumption axioms: $C \sqsubseteq D$ is true in an interpretation $\mathcal{I}$:

- Written $\mathcal{I} \models C \sqsubseteq D$,
- holds if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- alternatively, if and only if $\mathcal{I}(C) \subseteq \mathcal{I}(D)$

Assertions: $C(a)$ or $R(a, b)$ is true in $\mathcal{I}$:

- Written $\mathcal{I} \models C(a)$ or $\mathcal{I} \models R(a, b)$,
- $\mathcal{I} \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \models R(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

We say that $\mathcal{I}$ satisfies a set of sentences S, written $\mathcal{I} \models S$ iff

- $\mathcal{I} \models s$ for all $s \in S$.

Taking stock

- Interpretations/models $\mathcal{I}$ are functions
- $C^{\mathcal{I}}$ might have been written $\mathcal{I}(C)$
- Interpretations fix reference/meaning in a set or domain $\Delta^{\mathcal{I}}$, e. g.:
  - $\Delta^{\mathcal{I}} = \{a, b, c, d, e\}$
  - $C^{\mathcal{I}} = \{c, d, e\}$
  - $R^{\mathcal{I}} = \{(a, d), (a, e), (b, c)\}$

$$\begin{aligned}(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(a) \cap C^{\mathcal{I}} \neq \emptyset\} \\ &= \{a, b\}\end{aligned}$$

- Truth is in turn defined in terms of reference ...
- to yield a complex notion of a statement's being true in a model $\mathcal{I}$

What's the point?

- Semantic technology is about computable descriptions of data

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,

What's the point?

- Semantic technology is about computable descriptions of data
    - where the data descriptions are declarative,
    - give the intended interpretation of the data,
    - and of the relationship between data items

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to
  - check for consistency

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to
  - check for consistency
  - add implicit information

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to
  - check for consistency
  - add implicit information
  - answer complex queries

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to
  - check for consistency
  - add implicit information
  - answer complex queries
- Automated inference is based on *logical entailment*

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to
  - check for consistency
  - add implicit information
  - answer complex queries
- Automated inference is based on *logical entailment*
  - which is defined in terms of *truth in a class of models*

What's the point?

- Semantic technology is about computable descriptions of data
  - where the data descriptions are declarative,
  - give the intended interpretation of the data,
  - and of the relationship between data items
- The descriptions enable computers to reason *logically*, e.g. to
  - check for consistency
  - add implicit information
  - answer complex queries
- Automated inference is based on *logical entailment*
  - which is defined in terms of *truth in a class of models*
  - hence we need a precise definition of what truth in a model is

Entailment, countermodels and consistency in general

Entailment, countermodels and consistency in general

Validity

A set of sentences S *entails* a formula ψ, written $S \vDash \psi$, iff $\mathcal{I} \vDash \psi$ whenever $\mathcal{I} \vDash S$ for all interpretations $\mathcal{I}$ of the given class.

Entailment, countermodels and consistency in general

Validity

A set of sentences S *entails* a formula ψ, written $S \vDash \psi$, iff $\mathcal{I} \vDash \psi$ whenever $\mathcal{I} \vDash S$ for all interpretations $\mathcal{I}$ of the given class.

Consistency

A set of sentences S is consistent iff it has a model. That is, if and only if there is a model $\mathcal{I}$ such that $\mathcal{I} \vDash S$.

Entailment, countermodels and consistency in general

Validity

A set of sentences S *entails* a formula ψ, written $S \vDash \psi$, iff $\mathcal{I} \vDash \psi$ whenever $\mathcal{I} \vDash S$ for all interpretations $\mathcal{I}$ of the given class.

Consistency

A set of sentences S is consistent iff it has a model. That is, if and only if there is a model $\mathcal{I}$ such that $\mathcal{I} \vDash S$.

Countermodels

A set of sentences S does not entail a formula ψ if there is a model $\mathcal{I}$ such that $\mathcal{I} \vDash S$ but $\mathcal{I} \nvDash \psi$. We say that $\mathcal{I}$ is a countermodel for the entailment $S \Rightarrow \psi$

… and in $\mathcal{ALCQ}$ TBoxes

Validity

A subsumption axiom $C \sqsubseteq D$ is entailed by an ontology $\mathcal{O}$ iff $\mathcal{O} \models C \sqsubseteq D$, that is, iff $\mathcal{I} \models C \sqsubseteq D$ whenever $\mathcal{I} \models \mathcal{O}$ for all $\mathcal{ALCQ}$ models $\mathcal{I}$

Countermodels

An ontology $\mathcal{O}$ does not entail a subsumption axiom $C \sqsubseteq D$ if there is an $\mathcal{ALCQ}$ model $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{O} \cup \{C\}$ but $\mathcal{I} \not\models D$.

Outline

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Ax2 $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Ax2 $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

- All drive axles of *TwoCV*s are front axles

Revisiting some examples from lecture 8

Ax1 $\mathit{TwoCV} \sqsubseteq \mathit{Car}$

- Any *TwoCV* is a car

Ax2 $\mathit{TwoCV} \sqsubseteq \forall \mathit{driveAxle}.\mathit{FrontAxle}$

- All drive axles of *TwoCV*s are front axles

Ax3 $\mathit{FrontDrivenCar} \equiv \mathit{Car} \sqcap \forall \mathit{driveAxle}.\mathit{FrontAxle}$

Revisiting some examples from lecture 8

Ax1 $\mathit{TwoCV} \sqsubseteq \mathit{Car}$

- Any *TwoCV* is a car

Ax2 $\mathit{TwoCV} \sqsubseteq \forall \mathit{driveAxle}.\mathit{FrontAxle}$

- All drive axles of *TwoCV*s are front axles

Ax3 $\mathit{FrontDrivenCar} \equiv \mathit{Car} \sqcap \forall \mathit{driveAxle}.\mathit{FrontAxle}$

- A front driven car is one where all drive axles are front axles

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Ax2 $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

- All drive axles of *TwoCV*s are front axles

Ax3 $FrontDrivenCar \equiv Car \sqcap \forall driveAxle.FrontAxle$

- A front driven car is one where all drive axles are front axles

Now let's ask some questions:

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Ax2 $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

- All drive axles of *TwoCV*s are front axles

Ax3 $FrontDrivenCar \equiv Car \sqcap \forall driveAxle.FrontAxle$

- A front driven car is one where all drive axles are front axles

Now let's ask some questions:

- Does Ax1 entail that any *TwoCV* is a *Car*?

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Ax2 $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

- All drive axles of *TwoCV*s are front axles

Ax3 $FrontDrivenCar \equiv Car \sqcap \forall driveAxle.FrontAxle$

- A front driven car is one where all drive axles are front axles

Now let's ask some questions:

- Does Ax1 entail that any *TwoCV* is a *Car*?
- Does Ax1 and Ax2 entail that any *TwoCV* has a *FrontAxle*?

Revisiting some examples from lecture 8

Ax1 $TwoCV \sqsubseteq Car$

- Any *TwoCV* is a car

Ax2 $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

- All drive axles of *TwoCV*s are front axles

Ax3 $FrontDrivenCar \equiv Car \sqcap \forall driveAxle.FrontAxle$

- A front driven car is one where all drive axles are front axles

Now let's ask some questions:

- Does Ax1 entail that any *TwoCV* is a *Car*?
- Does Ax1 and Ax2 entail that any *TwoCV* has a *FrontAxle*?
- Is it consistent to assume that a front driven car may lack a drive axle?

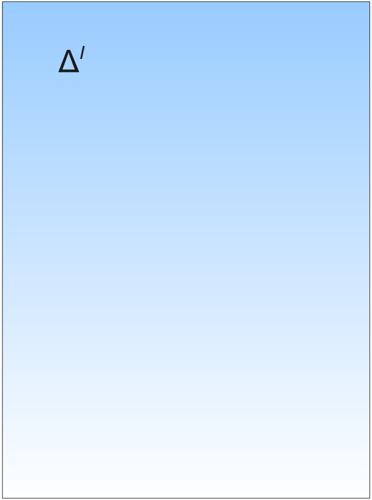
Does Ax1 entail that any *TwoCV* is a *CAR*?

Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,

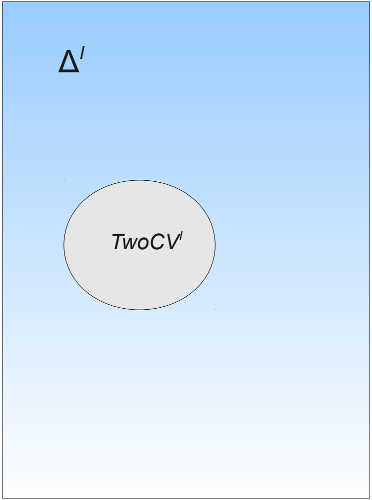
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,



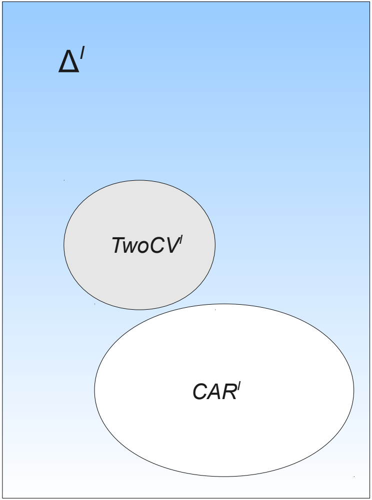
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$



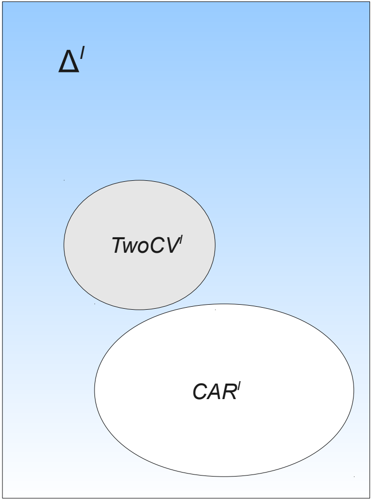
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$



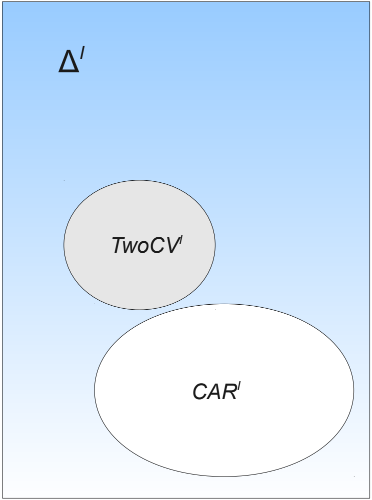
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$
- Check what the axioms require



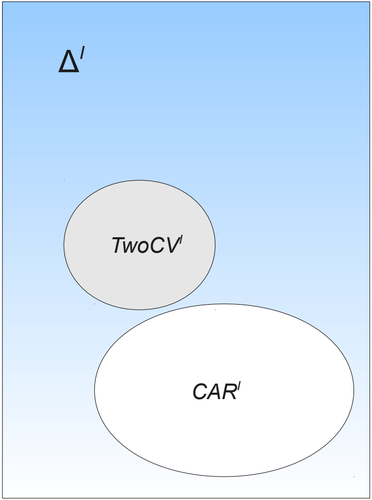
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$
- Check what the axioms require
- In this case $TwoCV^{\mathcal{I}} \subseteq CAR^{\mathcal{I}}$ (Ax1)



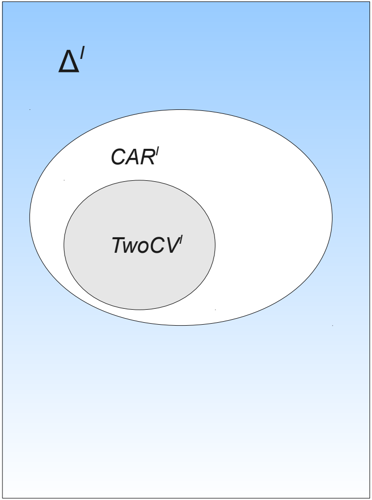
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$
- Check what the axioms require
- In this case $TwoCV^{\mathcal{I}} \subseteq CAR^{\mathcal{I}}$ (Ax1)
- Adjust the model accordingly



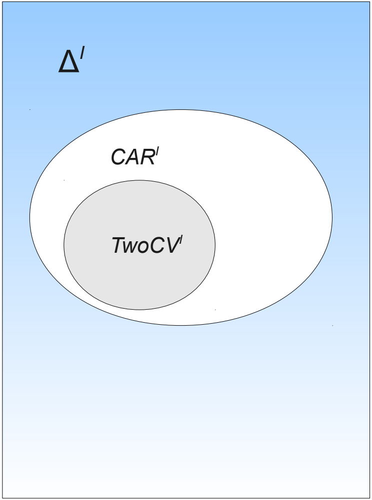
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$
- Check what the axioms require
- In this case $TwoCV^{\mathcal{I}} \subseteq CAR^{\mathcal{I}}$ (Ax1)
- Adjust the model accordingly
- *TwoCV*s are *CAR*s in *this model*



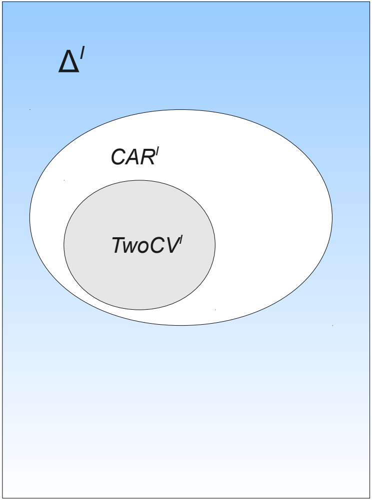
Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$
- Check what the axioms require
- In this case $TwoCV^{\mathcal{I}} \subseteq CAR^{\mathcal{I}}$ (Ax1)
- Adjust the model accordingly
- *TwoCV*s are *CAR*s in *this model*
- The model was chosen arbitrarily



Does Ax1 entail that any *TwoCV* is a *CAR*?

- Fix any domain of objects $\Delta^{\mathcal{I}}$,
- Fix a set $TwoCV^{\mathcal{I}}$
- Fix a set $CAR^{\mathcal{I}}$
- Check what the axioms require
- In this case $TwoCV^{\mathcal{I}} \subseteq CAR^{\mathcal{I}}$ (Ax1)
- Adjust the model accordingly
- *TwoCV*s are *CAR*s in *this model*
- The model was chosen arbitrarily
- So *TwoCV*s ar *CAR*s in *all models*



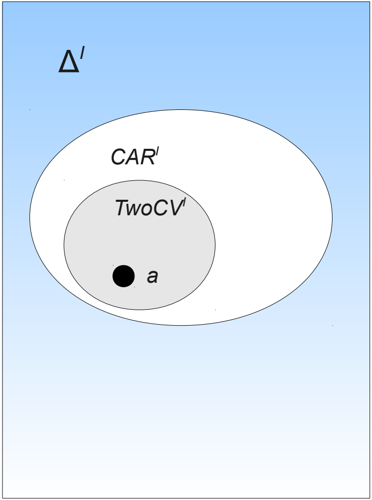
Does being a *TwoCV* *entail* having a *FrontAxle*?

Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*

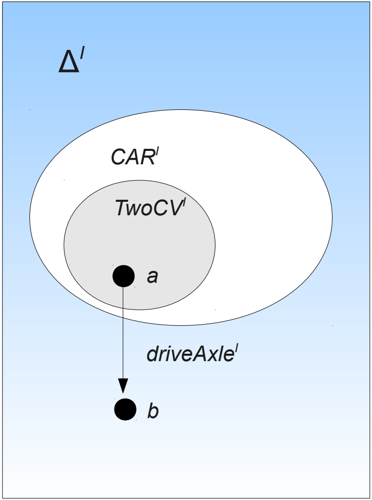
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*



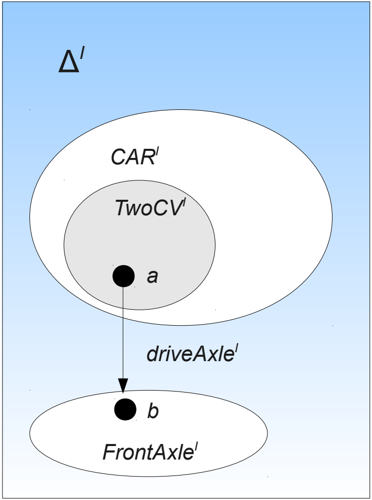
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*



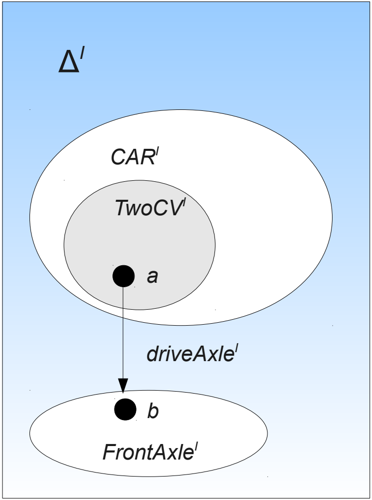
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*



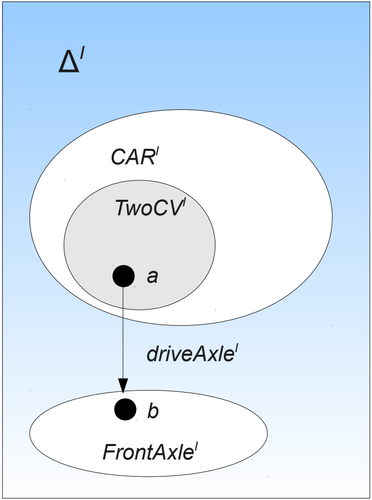
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV a*
- assume it has a *driveAxle b*
- which is a *FrontAxle*
- Ax1 is still satisfied



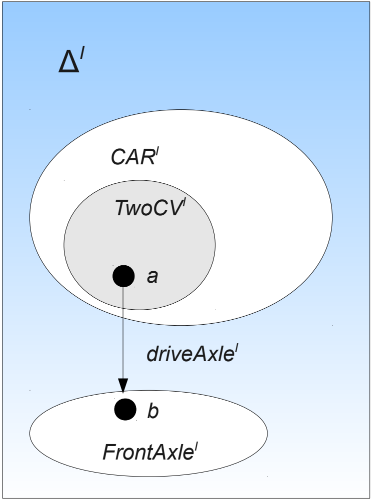
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*
- Ax1 is still satisfied
- Ax2 requires



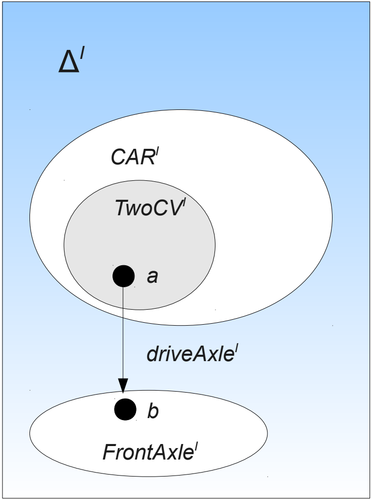
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*
- Ax1 is still satisfied
- Ax2 requires
    - that *TwoCV*s only have front axles



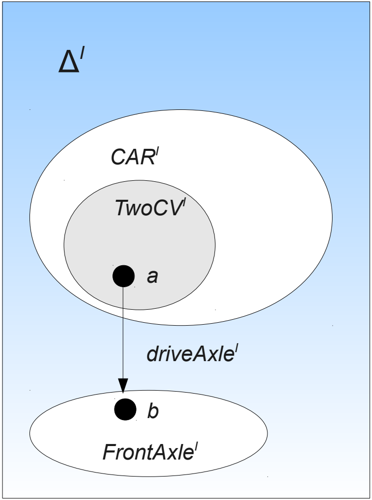
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*
- Ax1 is still satisfied
- Ax2 requires
  - that *TwoCV*s only have front axles
  - in this case it requires $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$



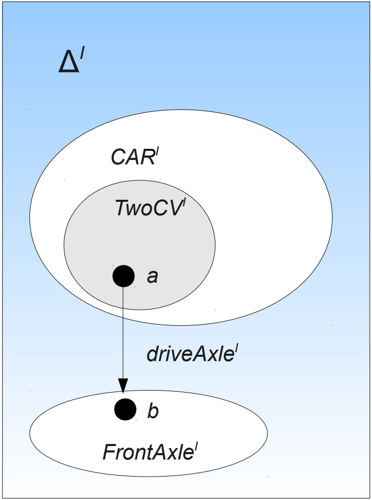
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*
- Ax1 is still satisfied
- Ax2 requires
  - that *TwoCV*s only have front axles
  - in this case it requires $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- now $driveAxle^{\mathcal{I}}(a) = b \in Frontaxle^{\mathcal{I}}$



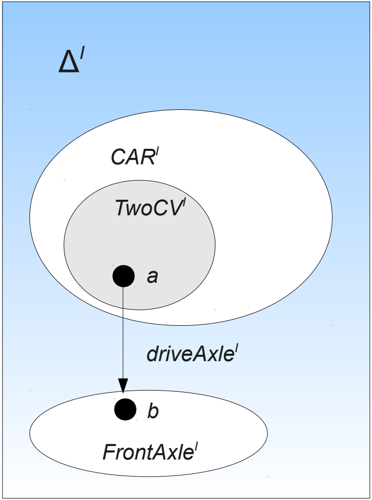
Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*
- Ax1 is still satisfied
- Ax2 requires
  - that *TwoCV*s only have front axles
  - in this case it requires $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- now $driveAxle^{\mathcal{I}}(a) = b \in Frontaxle^{\mathcal{I}}$
- hence both Ax1 and Ax2 are satisfied



Does being a *TwoCV* *entail* having a *FrontAxle*?

- Let's start with a particular *TwoCV* *a*
- assume it has a *driveAxle* *b*
- which is a *FrontAxle*
- Ax1 is still satisfied
- Ax2 requires
  - that *TwoCV*s only have front axles
  - in this case it requires $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- now $driveAxle^{\mathcal{I}}(a) = b \in Frontaxle^{\mathcal{I}}$
- hence both Ax1 and Ax2 are satisfied
- and *a* has a *FrontAxle*



Hang on !?

- This does not in fact answer our question

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models
- But this is what we did in the first example, what is the difference?

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models
- But this is what we did in the first example, what is the difference?
- In the first example we chose the model arbitrarily, i. e.:

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models
- But this is what we did in the first example, what is the difference?
- In the first example we chose the model arbitrarily, i. e.:
    - We did not make any particular assumptions about it

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models
- But this is what we did in the first example, what is the difference?
- In the first example we chose the model arbitrarily, i. e.:
    - We did not make any particular assumptions about it
    - except of course, that it satisfies the axioms,

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models
- But this is what we did in the first example, what is the difference?
- In the first example we chose the model arbitrarily, i. e.:
  - We did not make any particular assumptions about it
  - except of course, that it satisfies the axioms,
  - therefore whatever properties that *that* model has *all* models have

Hang on !?

- This does not in fact answer our question
- Entailment is truth in *all* models
- Truth in *one* model does not necessarily generalize to *all* models
- But this is what we did in the first example, what is the difference?
- In the first example we chose the model arbitrarily, i. e.:
  - We did not make any particular assumptions about it
  - except of course, that it satisfies the axioms,
  - therefore whatever properties that *that* model has *all* models have
  - .... again, given that they satisfy the axioms

Hang on a little longer!

- In the second example we chose an intended model, i.e.

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,
  - notably, that the only *TwoCV* in the model has a *FrontAxel*

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,
  - notably, that the only *TwoCV* in the model has a *FrontAxel*
  - clearly, this is just an assumption

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,
  - notably, that the only *TwoCV* in the model has a *FrontAxel*
  - clearly, this is just an assumption
  - it says nothing about other *TwoCV*s in other models

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,
  - notably, that the only *TwoCV* in the model has a *FrontAxel*
  - clearly, this is just an assumption
  - it says nothing about other *TwoCV*s in other models
- Truth in a model and validity in a class of models is not the same

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,
  - notably, that the only *TwoCV* in the model has a *FrontAxel*
  - clearly, this is just an assumption
  - it says nothing about other *TwoCV*s in other models
- Truth in a model and validity in a class of models is not the same
- Truth in one model does not rule out falsity in another

Hang on a little longer!

- In the second example we chose an intended model, i.e.
  - we chose a model according to our intutitions about cars and axles
  - in so doing, we made certain assumptions beyond the axioms,
  - notably, that the only *TwoCV* in the model has a *FrontAxel*
  - clearly, this is just an assumption
  - it says nothing about other *TwoCV*s in other models
- Truth in a model and validity in a class of models is not the same
- Truth in one model does not rule out falsity in another
- Let's see if we can find a countermodel ...

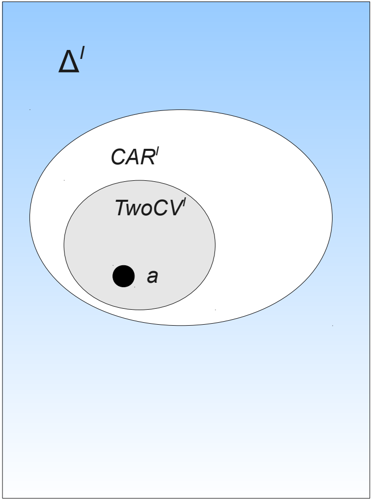
A countermodel

A countermodel

- Let's start with the same *TwoCV*

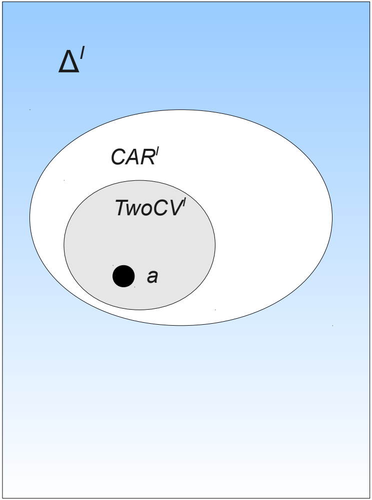
A countermodel

- Let's start with the same *TwoCV*



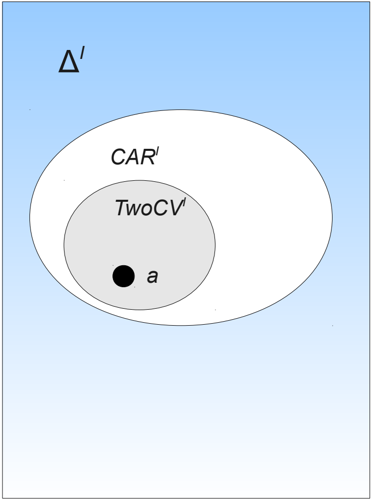
A countermodel

- Let's start with the same *TwoCV*
- only this time it has no *driveAxle*



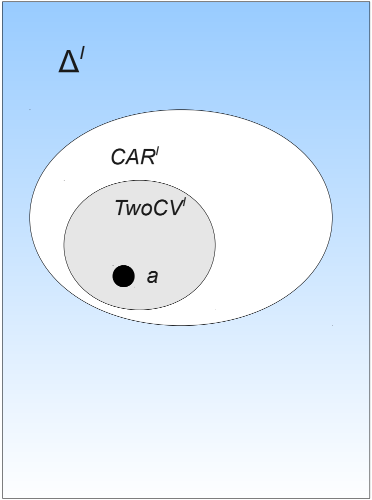
A countermodel

- Let's start with the same *TwoCV*
- only this time it has no *driveAxle*
- are Ax1 and Ax2 still satisfied?



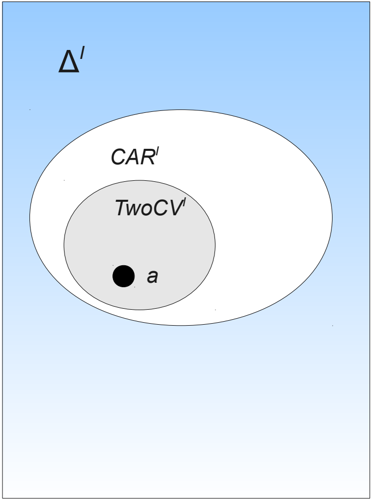
A countermodel

- Let's start with the same *TwoCV*
- only this time it has no *driveAxle*
- are Ax1 and Ax2 still satisfied?
- Clearly Ax1 is



A countermodel

- Let's start with the same *TwoCV*
- only this time it has no *driveAxle*
- are Ax1 and Ax2 still satisfied?
- Clearly Ax1 is
- For Ax2 we need to calculate a little:



Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$

Is Ax2 satisfied?

- We recall that Ax2 is $\mathit{TwoCV} \sqsubseteq \forall \mathit{driveAxle}.\mathit{FrontAxle}$
- This axioms is satisfied if $\mathit{TwoCV}^{\mathcal{I}} \subseteq (\forall \mathit{driveAxle}.\mathit{FrontAxle})^{\mathcal{I}}$

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : if\ driveAxle^{\mathcal{I}}(a, b)\ then\ b \in FrontAxle^{\mathcal{I}}\}$$

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : \textit{if } driveAxle^{\mathcal{I}}(a, b) \textit{ then } b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : \mathit{if}\ driveAxle^{\mathcal{I}}(a, b)\ \mathit{then}\ b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

- Hence we must show that $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : \mathit{if}\ driveAxle^{\mathcal{I}}(a, b)\ \mathit{then}\ b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

- Hence we must show that $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- But $driveAxle^{\mathcal{I}}(a) = \emptyset$; a has *no* drive axle at all

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : \textit{if } driveAxle^{\mathcal{I}}(a, b) \textit{ then } b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

- Hence we must show that $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- But $driveAxle^{\mathcal{I}}(a) = \emptyset$; a has *no* drive axle at all
- Since the emptyset is a subset of every other set we thus have

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : if\ driveAxle^{\mathcal{I}}(a, b)\ then\ b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

- Hence we must show that $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- But $driveAxle^{\mathcal{I}}(a) = \emptyset$; a has *no* drive axle at all
- Since the emptyset is a subset of every other set we thus have
- $\emptyset \subseteq FrontAxle^{\mathcal{I}}$, whence

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : \textit{if } driveAxle^{\mathcal{I}}(a, b) \textit{ then } b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

- Hence we must show that $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- But $driveAxle^{\mathcal{I}}(a) = \emptyset$; a has *no* drive axle at all
- Since the emptyset is a subset of every other set we thus have
- $\emptyset \subseteq FrontAxle^{\mathcal{I}}$, whence
- $driveAxle^{\mathcal{I}}(a) = \emptyset \subseteq FrontAxle^{\mathcal{I}}$

Is Ax2 satisfied?

- We recall that Ax2 is $TwoCV \sqsubseteq \forall driveAxle.FrontAxle$
- This axioms is satisfied if $TwoCV^{\mathcal{I}} \subseteq (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- Now, $a \in TwoCV^{\mathcal{I}}$ so we must put $a \in (\forall driveAxle.FrontAxle)^{\mathcal{I}}$
- According to $\mathcal{ALC}$ semantics we have

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : \textit{if } driveAxle^{\mathcal{I}}(a, b) \textit{ then } b \in FrontAxle^{\mathcal{I}}\}$$

- or

$$(\forall driveAxle.FrontAxle)^{\mathcal{I}} = \{a \in \Delta : driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}\}$$

- Hence we must show that $driveAxle^{\mathcal{I}}(a) \subseteq FrontAxle^{\mathcal{I}}$
- But $driveAxle^{\mathcal{I}}(a) = \emptyset$; a has *no* drive axle at all
- Since the emptyset is a subset of every other set we thus have
- $\emptyset \subseteq FrontAxle^{\mathcal{I}}$, whence
- $driveAxle^{\mathcal{I}}(a) = \emptyset \subseteq FrontAxle^{\mathcal{I}}$
- In other words, Ax2 is satisfied (vacuously), whence $\mathcal{I}$ is a countermodel.

Can a front driven car lack a drive axle?

Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:

Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model

Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object a s.t.:

Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object a s.t.:
  - a is a Car

Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object a s.t.:
  - a is a *Car*
  - a has either no axle or

Can a front driven car lack a drive axle?

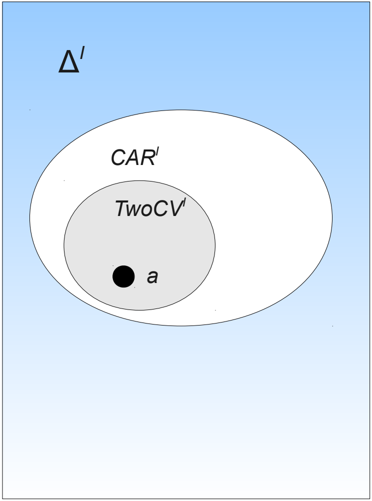
- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object *a* s.t.:
    - *a* is a *Car*
    - *a* has either no axle or
    - only (one or more) front axles

Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object *a* s.t.:
  - *a* is a *Car*
  - *a* has either no axle or
  - only (one or more) front axles
- we already have such a model, namely

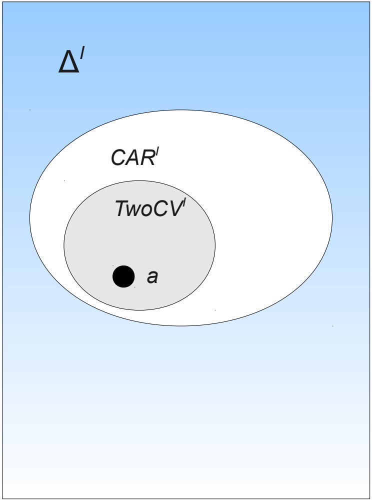
Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object *a* s.t.:
    - *a* is a *Car*
    - *a* has either no axle or
    - only (one or more) front axles
- we already have such a model, namely



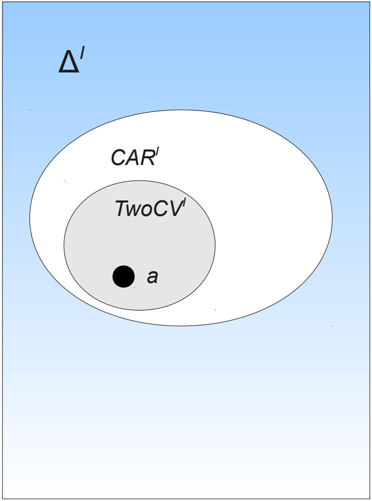
Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object *a* s.t.:
  - *a* is a *Car*
  - *a* has either no axle or
  - only (one or more) front axles
- we already have such a model, namely
- Ax1-3 entail that *TwoCV*s are front driven cars



Can a front driven car lack a drive axle?

- Let's try to answer it affirmatively:
- We need to construct a model
- in which there is an object *a* s.t.:
  - *a* is a *Car*
  - *a* has either no axle or
  - only (one or more) front axles
- we already have such a model, namely
- Ax1-3 entail that *TwoCV*s are front driven cars
- and we have already proved the property for *TwoCV*s



Outline

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
    - There are always *infinitely many models* to check

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
    - There are always *infinitely many models* to check
    - but an algorithm is a finite object

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

It is therefore common to supplement a semantics with a proof system

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

It is therefore common to supplement a semantics with a proof system

- which is a system of *inference rules*

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

It is therefore common to supplement a semantics with a proof system

- which is a system of *inference rules*
- in which each step of a derivation is determined by *syntactical form alone*

Semantics and calculi

Model-theoretic semantics yields an unambigous notion of entailment,

- But does not lend itself naturally to implementation
- The problem is that
  - There are always *infinitely many models* to check
  - but an algorithm is a finite object
  - so entailment cannot be checked directly

It is therefore common to supplement a semantics with a proof system

- which is a system of *inference rules*
- in which each step of a derivation is determined by *syntactical form alone*
- and every derivation terminates

Recalling a few RDFS rules

We have seen that RDFS can be characterised by rules that includes:

Membership abstraction:

$$\frac{\texttt{u rdfs:subClassOf x .} \qquad \texttt{v rdf:type u .}}{\texttt{v rdf:type x .}}\ \text{rdfs9}$$

Transitivity of subsumption:

$$\frac{\texttt{u rdfs:subClassOf v .} \qquad \texttt{v rdfs:subClassOf x .}}{\texttt{u rdfs:subClassOf x .}}\ \text{rdfs11}$$

Domain reasoning:

$$\frac{\texttt{p rdfs:domain u .} \qquad \texttt{x p y .}}{\texttt{x rdf:type u .}}\ \text{rdfs2}$$

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

- One proves that,

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

- One proves that,
    I. every conclusion derivable in the calculus from a set of premises A, is true in all models that satisfy A

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

- One proves that,
  I. every conclusion derivable in the calculus from a set of premises A, is true in all models that satisfy A
  II. and conversely that every statement entailed by A-models is derivable in the calculus when the elements of A are used as premises.

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

- One proves that,
  I. every conclusion derivable in the calculus from a set of premises A, is true in all models that satisfy A
  II. and conversely that every statement entailed by A-models is derivable in the calculus when the elements of A are used as premises.

We say that the calculus is

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

- One proves that,
  I. every conclusion derivable in the calculus from a set of premises A, is true in all models that satisfy A
  II. and conversely that every statement entailed by A-models is derivable in the calculus when the elements of A are used as premises.

We say that the calculus is

- sound wrt the semantics, if (I) holds, and

Soundness and completeness

Semantics and calculus are typically made to work like chopsticks:

- One proves that,
  I. every conclusion derivable in the calculus from a set of premises A, is true in all models that satisfy A
  II. and conversely that every statement entailed by A-models is derivable in the calculus when the elements of A are used as premises.

We say that the calculus is

- sound wrt the semantics, if (I) holds, and
- complete wrt the semantics, if (II) holds.

The complementarity of semantics and calculus

A proof system provides

- a finite means of reasoning *as if* we could run through an infinite number of models
- that is, a finite means of checking entailment

A semantics provides

- An intuitive justification for logical properties
- and a way to prove that properties *do not* hold (countermodels)

When semantics and proof system coincides

- We have a powerful way of checking positive and negative properties
- a finite representation of an infinite number of models,
- and a semantic justification for the inference rules in the proof system

Next time

David Norheim talks about Computas' RDF development projects:

1. Mediasone:
    - An RDF-based navigation application for Deichmanske Bibliotek:
    - Uses OWL with the Pellet reasoner, Virtuoso triple store and SPARQL
    - Collects data from dbpedia
2. Sublima:
    - Uses the OWL vocabulary SKOS (Simple Knowledge Oragnisation System)
    - Together with a Web interface for manual annotation of data