CITS3005 Knowledge Representation

Lecture 10: Reasoning in Ontologies

The University of Western Australia

2023

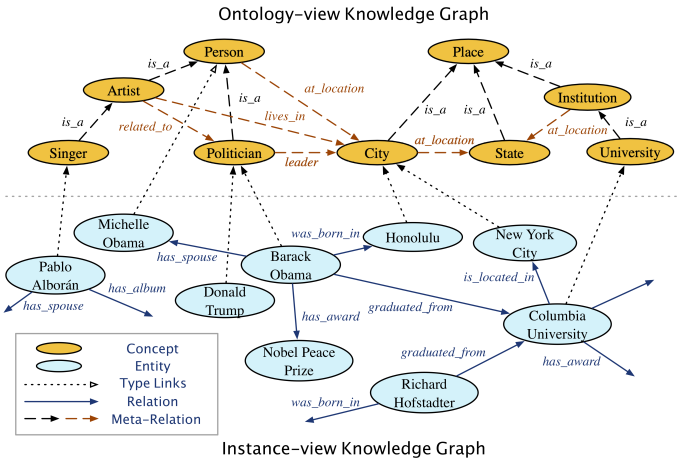
Reasoning in OWL

Reasoning applies rules and abstract knowledge by make *deduce* new knowledge from given facts. This lecture will look at the theory behind deduction, as well as processes for specifying rules in ontologies computing entailment.
Overview:

- Review deduction and entailment
- Limits of deduction and undecidability
- Description logics syntax and semantics
- General rules: SWRL

Entailment

Recall an ontology is an abstract graph that describes *many* interpretations/models.



- An interpretation I maps concepts to entities.
- A graph $\Gamma = (V_\Gamma, E_\Gamma, L_\Gamma)$, with an interpretation I is a model of the graph $G = (V, E, L)$ if $V^I \subset V_\Gamma$, $L^I \subset L_\Gamma$ and for all $(u, \ell, v) \in E$ we have $(u^I, \ell^I, v^I) \in E_\Gamma$.
- A semantic condition, ϕ, is a property over graphs, and a model of G satisfying ϕ is a ϕ-model.
- Given a set of semantic properties Φ, G_1 entails G_2 if every ϕ-model of G_1 is also a ϕ-model of G_2.

Intuitively this means that the latter graph says nothing new over the former graph and thus holds as a logical consequence of the former graph.

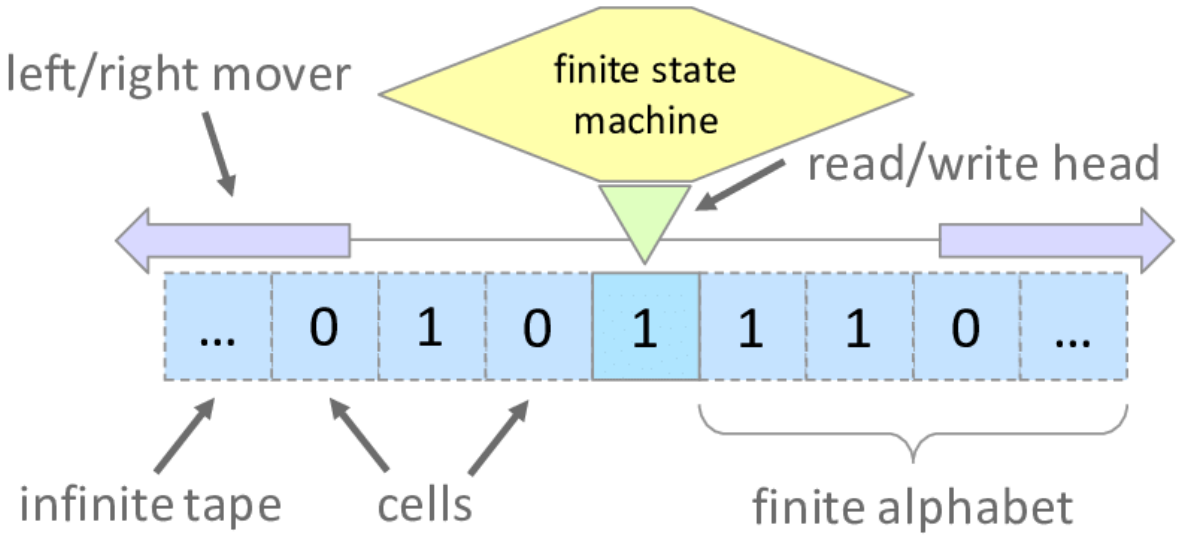
Reasoning

Reasoning is the process of determining *entailment*: property ψ follows from ϕ if the graph pattern for ψ is entailed by the graph pattern for ψ. Unfortunately, given two graphs, deciding if the first entails the second is undecidable: no (finite) algorithm for such entailment can exist that halts on all inputs with the correct answer.
However, there may be algorithms that:

1. halt on any pair of input ontologies but may miss entailments, returning false instead of true in some cases (SWRL)
2. always halt with the correct answer but only accept input ontologies with restricted features (Description logics)
3. only return correct answers for any pair of input ontologies but may never halt on certain inputs (prolog)

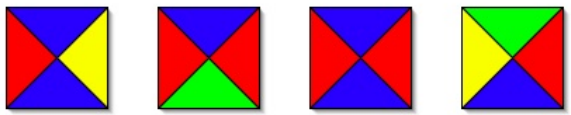
Undecidability

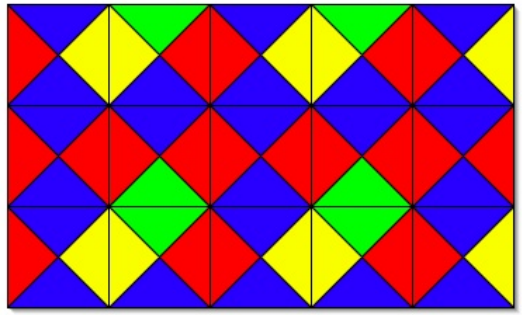
The decidability problem (*Entscheidungsproblem*) was first posed by David Hilbert, and subsequently addressed by Alonzo Church, Emil Post and Alan Turing.
The *decision problem* asked whether there was an effective procedure that could determine the truth of all logical questions in arithmetic.
It was shown that this is not the case: for example, there is no *algorithm* that determine if a given Turing machine will halt.

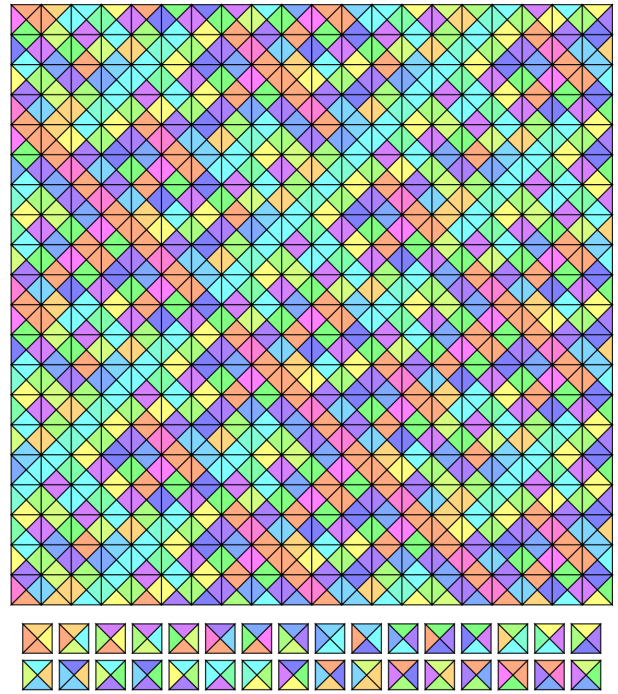


Proving Undecidability

The halting problem is **co-RE**, meaning that it would take an infinite amount of time to compute, but there exist harder problems!
The tiling problem asks whether a finite set of coloured tiles can tile the infinite plane so that their side always match. Defining concepts for each *tile* and relations for *left* and *up* it's not hard to see how this problem could be represented as an owl ontology.

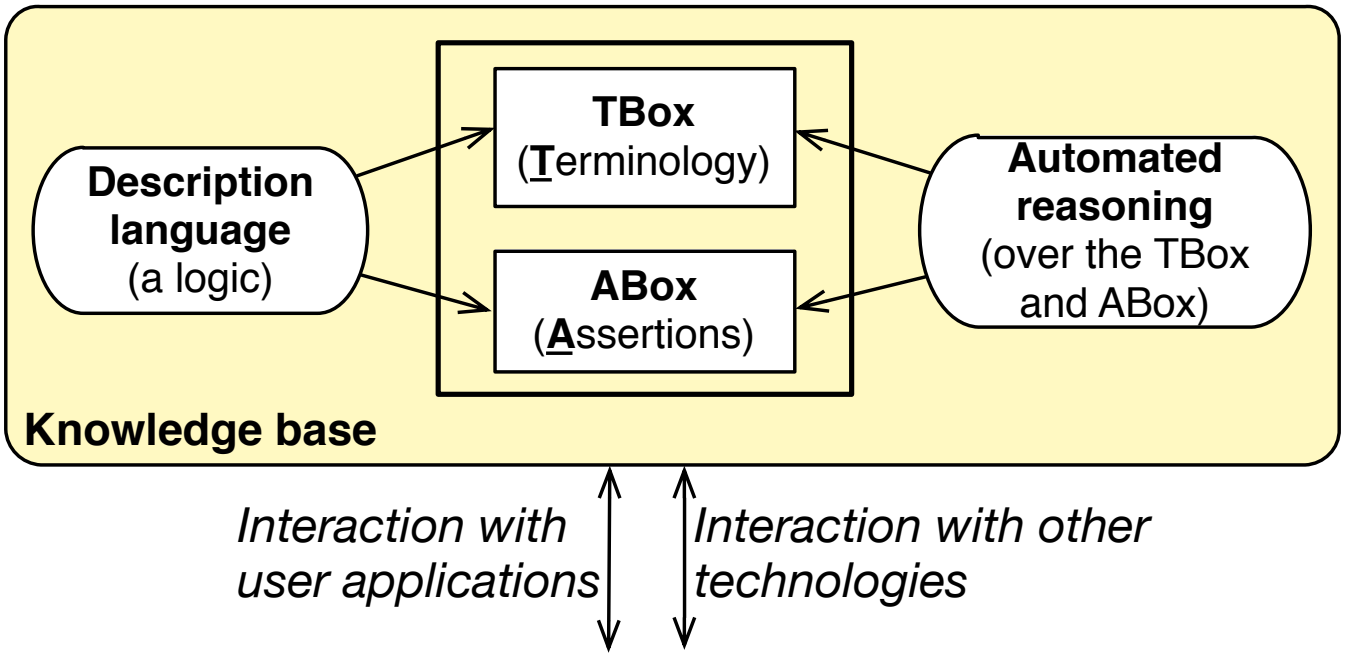






Description Logic

OWL is based on a decidable *description logic* ($\mathcal{SROIQ}(D)$).
Description Logics are a fragment of first order logic suitable for representing graph structures, and basic reasoning tasks.
Description Logics provide a theoretical framework for representing knowledge graphs.



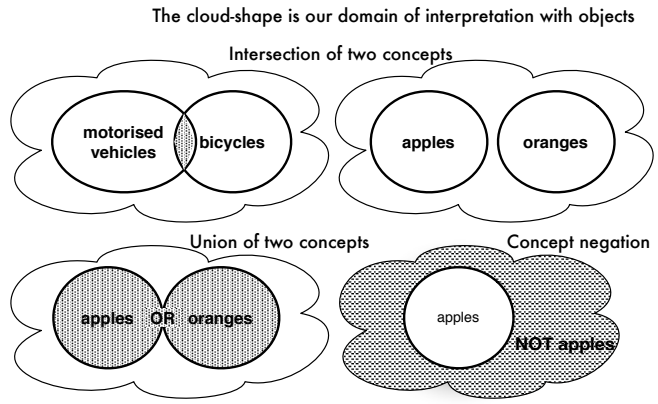
Description Logic ($\mathcal{ALC}$) syntax

- Concepts denoting entity types, include $\top$;
  Example: (primitive, atomic): `Book`, `Course`
- Roles denoting relationships;
- Operators: 'and' $\sqcap$, 'or' $\sqcup$, and 'not' $\neg$; 'for all' $\forall$ and 'exists' $\exists$
- Individuals (objects)
  Example: `Student(Mandla)`, `Mother(Sally)`, $\neg$`Student(Sally)`, `ENROLLED(Mandla, CS101/19/2)`
- Complex concepts using operators (where C and D are concept names, R a role name):
  - $\neg C$, $C \sqcap D$, and $C \sqcup D$ are concepts, and
  - $\forall R.C$ and $\exists R.C$ are concepts
- *T-Boxes*
  - `Student` $\sqsubseteq \exists$`ENROLLED.(Course` $\sqcup$ `DegreeProgramme)`
  - `Parent` $\equiv$ `(Male` $\sqcup$ `Female)` $\sqcap \exists$`CARESFOR.Mammal`

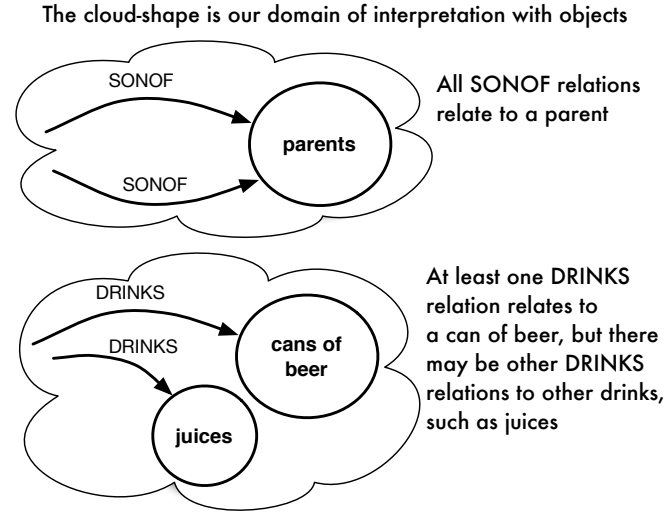
Semantics of $\mathcal{ALC}$

The semantics of $\mathcal{ALC}$ interprets the roles and concepts over a set of objects Δ with an interpretation: $\cdot^{\mathcal{I}}$ where

- $\cdot^{\mathcal{I}}$ maps every concept name A to a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- $\cdot^{\mathcal{I}}$ maps every role name R to a subset $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
- $\cdot^{\mathcal{I}}$ maps every individual name a to elements of $\Delta^{\mathcal{I}}$: $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$



- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
- $(\forall R.C)^{\mathcal{I}} = \{x \mid \forall y.R^{\mathcal{I}}(x,y) \rightarrow C^{\mathcal{I}}(y)\}$
- $(\exists R.C)^{\mathcal{I}} = \{x \mid \exists y.R^{\mathcal{I}}(x,y) \wedge C^{\mathcal{I}}(y)\}$



Semantics of $\mathcal{ALC}$

- C and D are concepts, R a role, a and b are individuals
- An interpretation $\mathcal{I}$ satisfies the statement $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
- An interpretation $\mathcal{I}$ satisfies the statement $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$
- $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $R(a, b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

- An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is a model of a knowledge base $\mathcal{KB}$ if every axiom of $\mathcal{KB}$ is satisfied by $\mathcal{I}$
- A knowledge base $\mathcal{KB}$ is said to be satisfiable if it admits a model

Description logic is a fragment of first order logic. eg:

- $C \sqsubseteq D \qquad \forall x(C(x) \rightarrow D(x))$
- $C \sqsubseteq D \sqcap E \qquad \forall x(C(x) \rightarrow D(x) \wedge E(x))$
- $C \sqsubseteq \exists R.D \qquad \forall x(C(x) \rightarrow \exists y(R(x, y) \wedge D(y))$
- $C \equiv \exists R.D \sqcup \exists S.D \qquad \forall x(C(x) \leftrightarrow \exists y((R(x, y) \vee S(x, y)) \wedge D(y))$

Full Description Logic Semantics

Name	Syntax	Semantics ($\cdot^I$)
	Class Definitions	
Atomic Class	A	A^I (a subset of Δ^I)
Top Class	$\top$	Δ^I
Bottom Class	$\bot$	$\emptyset$
Class Negation	$\neg C$	$\Delta^I \setminus C^I$
Class Intersection	$C \sqcap D$	$C^I \cap D^I$
Class Union	$C \sqcup D$	$C^I \cup D^I$
Nominal	$\{a_1, ..., a_n\}$	$\{a_1^I, ..., a_n^I\}$
Existential Restriction	$\exists R.C$	$\{x \mid \exists y : (x,y) \in R^I \text{ and } y \in C^I\}$
Universal Restriction	$\forall R.C$	$\{x \mid \forall y : (x,y) \in R^I \text{ implies } y \in C^I\}$
Self Restriction	$\exists R.\mathsf{Self}$	$\{x \mid (x,x) \in R^I\}$
Number Restriction	$\star n\, R$ (where $\star \in \{\geq, \leq, =\}$)	$\{x \mid \#\{y : (x,y) \in R^I\} \star n\}$
Qualified Number Restriction	$\star n\, R.C$ (where $\star \in \{\geq, \leq, =\}$)	$\{x \mid \#\{y : (x,y) \in R^I \text{ and } y \in C^I\} \star n\}$
	Class Axioms (T-Box)	
Class Inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
	Relation Definitions	
Relation	R	R^I (a subset of $\Delta^I \times \Delta^I$)
Inverse Relation	R^-	$\{(y,x) \mid (x,y) \in R^I\}$
Universal Relation	U	$\Delta^I \times \Delta^I$

Name	Syntax	Semantics ($\cdot^I$)
	Class Definitions	
Atomic Class	A	A^I (a subset of Δ^I)
Top Class	$\top$	Δ^I
Bottom Class	$\bot$	$\emptyset$
Class Negation	$\neg C$	$\Delta^I \setminus C^I$
Class Intersection	$C \sqcap D$	$C^I \cap D^I$
Class Union	$C \sqcup D$	$C^I \cup D^I$
Nominal	$\{a_1, ..., a_n\}$	$\{a_1^I, ..., a_n^I\}$
Existential Restriction	$\exists R.C$	$\{x \mid \exists y : (x,y) \in R^I \text{ and } y \in C^I\}$
Universal Restriction	$\forall R.C$	$\{x \mid \forall y : (x,y) \in R^I \text{ implies } y \in C^I\}$
Self Restriction	$\exists R.\mathsf{Self}$	$\{x \mid (x,x) \in R^I\}$
Number Restriction	$\star n\, R$ (where $\star \in \{\geq, \leq, =\}$)	$\{x \mid \#\{y : (x,y) \in R^I\} \star n\}$
Qualified Number Restriction	$\star n\, R.C$ (where $\star \in \{\geq, \leq, =\}$)	$\{x \mid \#\{y : (x,y) \in R^I \text{ and } y \in C^I\} \star n\}$
	Class Axioms (T-Box)	
Class Inclusion	$C \sqsubseteq D$	$C^I \subseteq D^I$
	Relation Definitions	
Relation	R	R^I (a subset of $\Delta^I \times \Delta^I$)
Inverse Relation	R^-	$\{(y,x) \mid (x,y) \in R^I\}$
Universal Relation	U	$\Delta^I \times \Delta^I$

Automated reasoning

We require a method to automate logical entailment, without creating undecidable problems.

- $\mathcal{KB} \models \phi$ if every model of $\mathcal{KB}$ is a model of ϕ
- Example:
  TBox: $\exists \texttt{TEACHES.Course} \sqsubseteq \neg \texttt{Undergrad} \sqcup \texttt{Professor}$
  ABox: $\texttt{TEACHES}(\texttt{John}, \texttt{cs101})$, $\texttt{Course}(\texttt{cs101})$, $\texttt{Undergrad}(\texttt{John})$
- $\mathcal{KB} \models \texttt{Professor}(\texttt{John})$
- What if:
  TBox: $\exists \texttt{TEACHES.Course} \sqsubseteq \texttt{Undergrad} \sqcup \texttt{Professor}$
  ABox: $\texttt{TEACHES}(\texttt{John}, \texttt{cs101})$, $\texttt{Course}(\texttt{cs101})$, $\texttt{Undergrad}(\texttt{John})$
- $\mathcal{KB} \models \texttt{Professor}(\texttt{John})$? or perhaps $\mathcal{KB} \models \neg\texttt{Professor}(\texttt{John})$?

Questions for DL-based OWL ontologies

- Concept (and role) satisfiability ($\mathcal{KB} \not\models C \sqsubseteq \bot$)
  - is there a model of $\mathcal{KB}$ in which C (resp. R) has a nonempty extension?
- Consistency of the knowledge base ($\mathcal{KB} \not\models \top \sqsubseteq \bot$)
  - Is the $\mathcal{KB} = (\mathcal{T}, \mathcal{A})$ consistent (non-selfcontradictory), i.e., is there at least a model for $\mathcal{KB}$?
- Concept (and role) subsumption ($\mathcal{KB} \models C \sqsubseteq D$)
  - i.e., is the extension of C (resp. R) contained in the extension of D (resp. S) in every model of $\mathcal{T}$?
- Instance checking ($\mathcal{KB} \models C(a)$ or $\mathcal{KB} \models R(a, b)$)
  - is a (resp. (a, b)) a member of concept C (resp. R) in $\mathcal{KB}$, i.e., is the fact $C(a)$ (resp. $R(a, b)$) satisfied by every interpretation of $\mathcal{KB}$?
- Instance retrieval ($\{a \mid \mathcal{KB} \models C(a)\}$)
  - find all members of C in $\mathcal{KB}$, i.e., compute all individuals a s.t. $C(a)$ is satisfied by every interpretation of $\mathcal{KB}$
- Query answering
  - compute all tuples of individuals t s.t. query $q(t)$ is entailed by $\mathcal{KB}$.

Tableaux Reasoners (Hermit and Pellet)

- We just require a sound and complete consistency checker for OWL.
- The tableaux method is a decision procedure which checks the existence of a model.
- It exhaustively looks at all the possibilities, so that it can eventually prove that no model could be found for unsatisfiable formulas.
- $\phi \models \psi$ iff $\phi \wedge \neg\psi$ is NOT satisfiable—if it is satisfiable, we have found a counterexample
- Decompose the formula in top-down fashion

- Tableaux calculus works only if the formula has been translated into Negation Normal Form,
- If a model satisfies a conjunction, then it also satisfies each of the conjuncts
- If a model satisfies a disjunction, then it also satisfies one of the disjuncts so it generates two alternative branches.
- Apply rules until (a) an explicit contradiction or (b) there is a completed branch where no more rule is applicable.

Number	Tableau	Explanation
1	$\forall x.R(x,x)$	Reflexivity axiom in the original theory T
2	$\forall x,y.\ \neg R(x,y) \vee \neg R(y,x)$	Asymmetry axiom in the original theory T
3	$\forall x,y.R(x,y)$	The negated axiom added to theory T
4		Substitute *x* for term *a* in 1,2,3
5	$R(a,a)$	
6	$\forall y.\ \neg R(a,y) \vee \neg R(y,a)$	
7	$\forall y.R(a,y)$	
8		Substitute *y* for term *a* in 2 and 3
9	$R(a,a)$	
10	$\neg R(a,a) \vee \neg R(a,a)$	
11	$R(a,a)$	
12		Split the disjunction of 10
13	$\neg R(a,a)$ $\quad$ $\neg R(a,a)$	Which each generate a clash with 9 and 11, hence, $\neg\forall x,y.R(x,y)$ is entailed by T.

Tableau reasoning for DLs

- Like for FOL: *Unfold the TBox; Convert the result into negation normal form; Apply the tableau rules to generate more Aboxes; and Stop when none of the rules are applicable*
- $\mathcal{T} \vdash C \sqsubseteq D$ *if all Aboxes contain clashes*
- $\mathcal{T} \nvdash C \sqsubseteq D$ *if some Abox does not contain a clash*

C and D are concepts, R a role

- $\neg\neg C$ gives C
- $\neg(C \sqcap D)$ gives $\neg C \sqcup \neg D$
- $\neg(C \sqcup D)$ gives $\neg C \sqcap \neg D$
- $\neg(\forall R.C)$ gives $\exists R.\neg C$
- $\neg(\exists R.C)$ gives $\forall R.\neg C$

$\sqcap$-rule If $(C_1 \sqcap C_2)(a) \in S$ but S does not contain both $C_1(a)$ and $C_2(a)$, then
$S = S \cup \{C_1(a), C_2(a)\}$

$\sqcup$-rule If $(C_1 \sqcup C_2)(a) \in S$ but S contains neither $C_1(a)$ nor $C_2(a)$, then
$S = S \cup \{C_1(a)\}$
$S = S \cup \{C_2(a)\}$

$\forall$-rule If $(\forall R.C)(a) \in S$ and S contains $R(a, b)$ but not $C(b)$, then
$S = S \cup \{C(b)\}$

$\exists$-rule If $(\exists R.C)(a) \in S$ and there is no b such that $C(b)$ and $R(a, b)$, then
$S = S \cup \{C(b), R(a, b)\}$

Example

- Let's say our ontology contains only:
  - 1a $\mathit{Vegan} \equiv \mathit{Person} \sqcap \forall \mathit{eats}.\mathit{Plant}$
  - 1b $\mathit{Vegetarian} \equiv \mathit{Person} \sqcap \forall \mathit{eats}.(\mathit{Plant} \sqcup \mathit{Dairy})$
- We want to know whether all vegans are vegetarians, i.e.: $\mathcal{T} \vdash \mathit{Vegan} \sqsubseteq \mathit{Vegetarian}$
- If that's true, then there is, or can be, an individual that is an instance of both, or:
- If that's true, then some object that instantiates the subclass but *not* the superclass *cannot* exist
  - 2 $\mathcal{S} = \{(\mathit{Vegan} \sqcap \neg \mathit{Vegetarian})(a)\}$
- Before entering the tableau, we'll 'unfold' it (informally, here: complex concepts on the left-hand side are replaced with their properties declared on the right-hand side)
- Check for NNF and rewrite if needed
- Then (finally) apply the tableau rules

Flavours of Description Logic

$\mathcal{ALC}$ is a relatively basic description logic, corresponding to the modal logic **K**. This is just one of a large taxonomy of languages: The expressivity is encoded in the label for a logic starting with one of the following basic logics:

- $\mathcal{AL}$ Attributive language. This is the base language which allows:
  - Atomic negation (negation of concept names that do not appear on the left-hand side of axioms)
  - Concept intersection
  - Universal restrictions
  - Limited existential quantification
- $\mathcal{FL}$ Frame based description language allows:
  - Concept intersection
  - Universal restrictions
  - Limited existential quantification
  - Role restriction
- $\mathcal{EL}$ Existential language, allows:
  - Concept intersection
  - Existential restrictions (of full existential quantification)

Flavours of Description Logic cont.

Further restrictions follow, with the labelling:

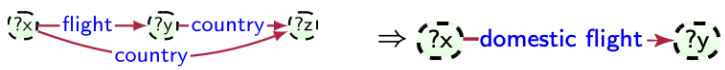
- $\mathcal{F}$ Functional properties, a special case of uniqueness quantification.
- $\mathcal{E}$ Full existential qualification (existential restrictions that have fillers other than $\top$).
- $\mathcal{U}$ Concept union.
- $\mathcal{C}$ Complex concept negation.
- $\mathcal{H}$ Role hierarchy (`subproperties: rdfs:subPropertyOf`).
- $\mathcal{R}$ Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
- $\mathcal{O}$ Nominals. (Enumerated classes of object value restrictions: `owl:oneOf`, `owl:hasValue`).
- $\mathcal{I}$ Inverse properties.
- $\mathcal{N}$ Cardinality restrictions (owl:cardinality, owl:maxCardinality), a special case of counting quantification
- $\mathcal{Q}$ Qualified cardinality restrictions (available in OWL 2), cardinality restrictions that have fillers other than $\top$.
- $^{(\mathcal{D})}$ Use of datatype properties, data values or data types.

$\mathcal{S}$ is used to abbreviate $\mathcal{ALC}$ and Description logic in Protègè is then: $\mathcal{SHOIN}^{(\mathcal{D})}$.

Roll your own axioms

OWL provides a compromise that allows entailment to be computable (but potentially very slow), but can limit expressivity.

Sometimes you would like to capture a specific axiom that you can't express in OWL (and accept the risk of undecidability).



Entailments can be applied directly to inference rules (or simply rules) encoding if–then-style consequences. A rule is composed of a body (if) and a head (then), like prolog, where with the body and head are given as graph patterns.

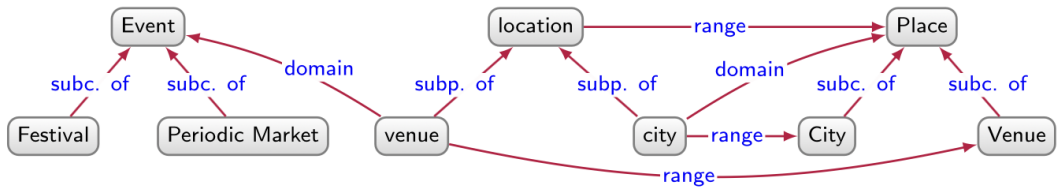
Table 4.4: Example rules for sub-class, sub-property, domain, and range features

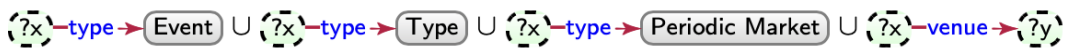
Feature	Body	⇒	Head
SUB-CLASS (I)	?x —type→ ?c —subc. of→ ?d	⇒	?x —type→ ?d
SUB-CLASS (II)	?d —subc. of→ ?d —subc. of→ ?e	⇒	?d —subc. of→ ?e
SUB-PROPERTY (I)	?x —?p→ ?y, ?p —subp. of→ ?q	⇒	?x —?q→ ?y
SUB-PROPERTY (II)	?p —subp. of→ ?q —subp. of→ ?r	⇒	?p —subp. of→ ?r
DOMAIN	?x —?p→ ?y, ?p —domain→ ?c	⇒	?x —type→ ?c
RANGE	?x —?p→ ?y, ?p —range→ ?c	⇒	?y —type→ ?c

Applying Rules.

Rules can be leveraged for reasoning in a number of ways:

- Materialisation refers to the idea of applying rules recursively to a graph, adding the conclusions generated back to the graph until a fixedpoint is reached and nothing more can be added.
- Query rewriting, given a query, will automatically extend the query in order to find solutions entailed by a set of rules.





Rule Definitions

Given a graph pattern Q, let $var(Q)$ be the variables in Q.

Definition

A *rule* is a pair $R = (B, H)$ such that B and H are graph patterns and $var(H) \subseteq var(B)$. B is the body of the rule and H is the head.

The head is interpreted as conjunction of edges, and the application of the rule inserts the head edges wherever the body is matched.

Definition

Given a rule $R = (B, H)$ and a graph G we define the *application of* R over G as the graph $R(G) = \bigcup_{\mu \in B(G)} \mu(H)$

Of course applying a rule, may introduce new edges that then create new matches for the body (e.g. a transitive closure).

Definition

Given a set of rule $\mathcal{R} = \{R_1, ..., R_n\}$, the *least model of* $\mathcal{R}$ *over* G is defined as $\mathcal{R}^*(G) = \bigcup_{k \in \mathbb{N}} \mathcal{R}^k(G)$ where $\mathcal{R}(G) = \bigcup_{R \in \mathcal{R}} R(G)$, $\mathcal{R}^0(G) = G$ and $\mathcal{R}^{i+1}(G) = \mathcal{R}(\mathcal{R}^{\rangle}(G))$.

Semantic Web Rule Language

SWRL (Semantic Web Rule Language) is a language that allows you to integrate your own inference rules into ontologies. Rules can be written in the Protgé editor or in Python, using Owlready, and then executed via the integrated HermiT or Pellet reasoners.
Rules are similar to prolog and consist of clauses, made of predicates and variables.

```
Bacterium(?b),
gram_positive(?b, true),
has_shape(?b, ?f), Ronde(?f)
has_grouping(?b, ?r), InCluster(?r)
-> Staphylococcus(?b)
```

SWRL Syntax

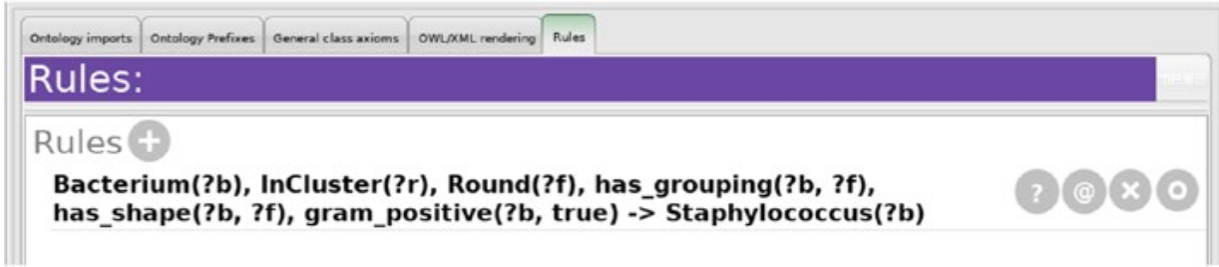
A SWRL rule is a clause with *body* and *head* separated by an arrow `->` (composed of the two characters: minus and greater than). The head and body are comma separated lists, where the comma is a logical "and".
The elements that make up conditions and consequences are called atoms, and the same atoms can be used in conditions and in consequences.
SWRL rules use variables, whose names start with `?`, for example, `?b`.
These variables can represent individuals or values but never classes or properties
The available atoms are:

- Membership of a class: `Class (?x)` (eg `Person(?x)`)
- Object property value: `object_property(?x, ?y)` (e.g. `studies(?s,?u)`)
- Data property value: `data_property(?x, ?y)` (e.g.`age(?s,19)`)
- Identical individuals: `SameAs(?x, ?y)`
- Distinct individuals: `DifferentFrom(?x, ?y)`
- Membership in a datatype: `datatype(?x)`

SWRL also provides the following built in functions and relations for defining atoms: such as `add(?result,?x,?y)`, `lessThan(?x,?y)` or `stringConcat(?result, ?x, ?y)`.

Defining SWRL Rules in Protege

SWRL rules can be entered in Protég e, in the *Active ontology* tab and the *Rules* subtab, as in the following image.



The rules defined in Protégé can then be executed with reasoners (e.g., using the `sync_reasoner()` function with Owlready). When the consequences of rules create new relationships for object properties, run the reasoner with option `infer_property_value= True`, and when they create new relationships for data properties, use the option `infer_data_property_value = True` option and make sure you use use the Pellet reasoner.

SWRL in OWLReady

In Owlready, the `Imp` (implies) class allows you to create SWRL rules,
In the following example, we create an ontology of people and calculate the Body Mass Index (BMI = $weight/size^2$) of the person.
We first create the ontology, with a Person class and three data properties: weight, size, and bmi.

```
>>> onto_person = get_ontology("http://test.org/person2.owl#")
>>> with onto_person:
...     class Person(Thing): pass
...     class weight(Person >> float, FunctionalProperty): pass
...     class size  (Person >> float, FunctionalProperty): pass
...     class bmi   (Person >> float, FunctionalProperty): pass
```

Then, we create a defined class for obese people (BMI¿30):

```
>>> with onto_person:
...     class Obese(Person):
...         equivalent_to = [Person & (bmi >= 30.0)]
```

Adding SWRL

We add a SWRL rule for BMI, where `?x` is an individual of the class Person with weight `?w`; size `?s`; size-squared `?s2`; and bmi `?b`.
We use `Imp` to create a new rule:

```
>>> with onto_person:
... ...
imp = Imp()
imp.set_as_rule("Person(?x), weight(?x, ?w),\
size(?x, ?s), multiply(?s2, ?s, ?s), divide(?b, ?w, ?s2)\
-> bmi(?x, ?b)")
```

We can then add two individuals and run pellet:

```
>>> p1 = Person(size = 1.7, weight = 65.0)
>>> p2 = Person(size = 1.7, weight = 90.0)
>>> sync_reasoner_pellet(infer_property_values = True,
...                      infer_data_property_values = True)
:
```

Inferred data

We can now access the updated concepts:

```
>>> p1.bmi
22.491348
>>> p1.is_a
[person2.Person]
>>> p2.bmi
31.141868
>>> p2.is_a
[person2.Obese]
```

We can access the rule through the `imp` variable:

```
>>> str(imp)
'Person(?x), weight(?x, ?w), size(?x, ?s),
multiply(?s2, ?s, ?s), divide(?b, ?w, ?s2) -> bmi(?x, ?b)'

>>> imp.body
[Person(?x), weight(?x, ?w), size(?x, ?s), multiply(?s2, ?s, ?s)
>>> imp.head
[bmi(?x, ?b)]
```

Summary

OWL2 enables reasoning at various levels from basic inexpressive operations (RDFS), through to description logics (ACL) and first order logic (SWRL).
It is possible to define undecidable entailments using the more expressive languages.
As the complexity increases with expressivity it is recommended that you always use the minimum level of expressivity you can get away with.
Description Logics give a formal foundation for OWL reasoning and entailment can be computed via entailment.

Next week, induction and learning!