Description Logics

Biswanath Dutta

Introduction: Logic

- Logic is the branch of philosophy.
- Logic is not the study of truth, but of the relationship between the truth of one statement and that of another.
- Logic is the study of how to make formal correct deductions and inferences.
- Logic is concerned with the use and study of valid reasoning.
    - Logic is to -> enable automation.
- The study of logic features prominently in mathematics and computer science.

Why Logic?

Used for	Advantages	Disadvantages
Formal specification	Well-understood with formal syntax and formal semantics: we can better specify and prove correctness	It cannot be used to interact with users
Automation	Pragmatically efficient for automation exploiting the explicitly codified semantics: reasoning services	An exponential grow in cost (computational, man power)

Logics, formal syntax and formal semantic

- A logic is a representation language with
  - a formal syntax
  - a formal semantics
- Any language can have these characteristics
  - eg., using mathematical notation, textual, graphical, …
- As formal languages, logics are suitable for:
  - representing (specification)
  - reasoning (automation) about data and knowledge.

Logics for Specification

- Logic as a formal language
  - is good for the specification (representation) of knowledge

- Logic as a formal semantics
  - is good for specification of declarative data and knowledge (as different from programs)
  - The meaning of sentences is declaratively defined, i.e. with logic we describe what holds without caring about how it can be computed.

Logics for Reasoning

- Logics provides a notion of deduction
  - axioms, deductive machinery, theorem
- Deduction can be used to implement reasoners
  - Reasoners allow inferring conclusions from a given knowledge base (i.e, a set of "premises", premises can be axioms or theorems).
- From implicit knowledge to explicit knowledge

- Model: **A model is an abstraction of a part of the world**
- Theory: A set of statements which describe the (part of the) world as abstracted in the (mental) model.

Specification / Representation

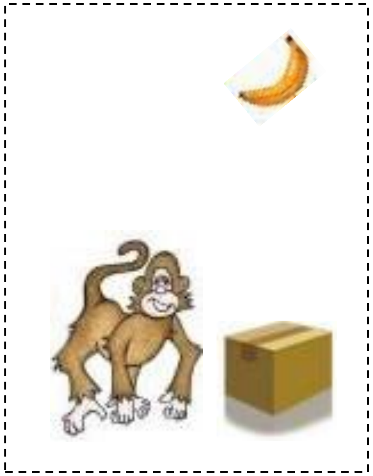
$L = \{MonkeyLow, BananaHigh, MonkeyClimbBox, MonkeyGetBanana, \wedge, \vee, \neg\}$

$T = \{\neg (MonkeyLow \wedge BananaHigh \wedge MonkeyGetBanana)$

$\neg (MonkeyLow \wedge MonkeyClimbBox)$

$\neg (\neg MonkeyLow \wedge \neg BananaHigh \wedge MonkeyGetBanana)\}$

MODEL#1



Informal Semantics:

"If the monkey is low and the banana is high in position, then the monkey cannot get the banana."

Formal Semantics:

I(MonkeyLow) = T

I(BananaHigh) = T

I(MonkeyClimbBox) = F

I(MonkeyGetBanana) = F

Reasoning / Automation

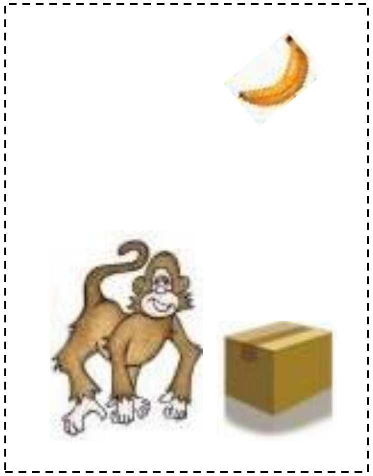
L = {MonkeyLow, BananaHigh, MonkeyClimbBox, MonkeyGetBanana, $\wedge$, $\vee$, $\neg$}

T = {$\neg$ (MonkeyLow $\wedge$ BananaHigh $\wedge$ MonkeyGetBanana)

$\neg$ (MonkeyLow $\wedge$ MonkeyClimbBox)

$\neg$ ($\neg$ MonkeyLow $\wedge$ $\neg$ BananaHigh $\wedge$ MonkeyGetBanana)}

MODEL#1



Given that:

MonkeyLow = T

BananaHigh = T

We derive that:

MonkeyGetBanana = F

Types of Logic

- **Syllogistic Logic**

As defined by Aristotle, from the combination of a general statement (the major premise) and a specific statement (the minor premise), a conclusion is deduced. For example, knowing that all men are mortal (major premise) and that Socrates is a man (minor premise), we may validly conclude that Socrates is mortal.

- **Propositional logic (propositional calculus)**

A propositional calculus or logic (also a sentential calculus) is a formal system in which *formulae representing propositions can be formed by combining atomic propositions* using logical connectives, and in which a system of formal proof rules establishes certain formulae as "theorems". Propositional logic is the foundation of first-order logic and higher-order logic. For example: **Premise 1: If it's raining then it's cloudy. Premise 2: It's raining. Conclusion: It's cloudy.**

- **Predicate logic (predicate calculus)**

In mathematical logic, predicate logic is the generic term for symbolic formal systems like first-order logic, second-order logic, many-sorted logic, or infinitary logic. This formal system is distinguished from other systems in that its formulae contain variables which can be quantified. Two common quantifiers are the existential ∃ ("there exists") and universal ∀ ("for all") quantifiers. The variables could be elements in the universe under discussion, or perhaps relations or functions over that universe.

In informal usage, the term "predicate logic" occasionally refers to first-order logic.

Types of Logic

- **Modal Logic**

- **Information Logic**

- **Mathematical Logic**

Mathematical logic really refers to two distinct areas of research: the first is the application of the techniques of formal logic to mathematics and mathematical reasoning, and the second, in the other direction, the application of mathematical techniques to the representation and analysis of formal logic.

- **Philosophical Logic**

- **Conputational Logic**

Logic cut to the heart of computer science as it emerged as a discipline: Alan Turing's work on the Entscheidungs problem followed from Kurt Gödel's work on the incompleteness theorems. The notion of the general purpose computer that came from this work was of fundamental importance to the designers of the computer machinery in the 1940s.

…

First Order Logic (FOL)

- It is also known as first-order predicate calculus, the lower predicate calculus, quantification theory, and predicate logic. First order predicate logic (FOL) is an `extension' of propositional logic, which enables us to represent more knowledge in more detail.

- First-order logic is a collection of formal systems used in mathematics, philosophy, linguistics, and computer science.

- First-order logic uses quantified variables over (non-logical) objects. It allows the use of sentences that contain variables, so that rather than propositions such as Socrates is a man one can have expressions in the form X is a man where X is a variable.

    - This distinguishes it from propositional logic, which does not use quantifiers.

Why FOL is not considered as a SW Language

- FOL is highly expressive
- It is too bulky for modelling
- It is not appropriate to find consensus in modelling
- Its proof theoretically is very complex (semi-decidable)
- It is not a Markup Language for the Web

Source: https://www.youtube.com/watch?v=9Yu7poe30pQ

Description Logics (DL)

- A DL is a structured fragment of FOL.
    - Compromise of expressivity and scalability

- A DL models concepts, roles and individuals and their relationships.

- Any (basic) Description Logic language is a subset of $\mathcal{L}3$, i.e., the function-free FOL using only at most three variable names, and its representation is at the predicate level: no variables are present in the formalism.

- DLs provide a logical reconstruction and (claimed to be a) unifying formalism for other knowledge representation languages, such as frames-based systems, object-oriented modelling, Semantic data models, etc.

- They provide the language to formulate theories and systems declaratively expressing structured information and for accessing and reasoning with it, and they are used for, among others, terminologies and ontologies, formal conceptual data modelling, and information integration.

What Are Description Logics?

•A family of logic based Knowledge Representation formalisms

- More expressive than Predicate logic
- Has efficient decision power

--(DL is a decidable fragments of first order logic)

- DL Logics are equipped with a formal semantics

-- Formal semantics of DL allows humans and computer systems to exchange DL ontologies without ambiguity as to their intended meaning, and also makes it possible to use logical deduction **to infer additional information from the facts stated explicitly in an ontology** (this is an important feature that distinguishes DLs from other modelling languages such as UML)

•DL describes

–domain in terms of **concepts** (classes), **roles** (properties, relationships) and **individuals**

–**Operators** allow for composition of complex concepts

–**Names** can be given to complex concepts (E.g., Happy parents)

•*Axiom*, the fundamental modeling concept of a DL, *is a logical statement relating roles and/or concepts*.

•Example for a DL: W3C Standard OWL 2 DL is based on description logics

Basic Building Blocks of DL Ontologies

- DLs provide means to model the relationships between entities in a domain of interest.
- In DLs, there are three kinds of entities: Concepts, Roles and Individual names.
- Concepts names denote sets of individuals and are equivalent to unary predicates.
  - In general, equivalent to formulae with one free variable (unary predicates / formulae with one free variable), E.g., Person, Female
- Roles names denote binary relations between the individuals and are equivalent to binary predicates.
  - In general, equivalent to formulae with two free variables (binary predicates /formulae with two free variables), E.g., hasChild
- Individuals names denote single individuals in the domain and are equivalent to constants, E.g., Mary, John
- Constructors
  - Union $\sqcup$: Man $\sqcup$ Woman
  - Intersection $\sqcap$: Doctor $\sqcap$ Mother
  - Exists restriction $\exists$: $\exists$hasChild.Doctor
  - Value restriction $\forall$: $\forall$hasChild.Doctor
  - Complement /negation $\neg$: Man $\sqsubseteq \neg$Mother
  - Number restriction $\geq$n, $\leq$n
- Axioms : Subsumption $\sqsubseteq$: Mother $\sqsubseteq$ Parent

Basic Building Blocks of DL Ontologies

- Unlike a database, a DL ontology does not fully describe a particular situation or "state of the world."
- DL, rather, consists of a set of statements, called axioms, each of which must be true in the situation described. These axioms typically capture only partial knowledge about the situation that the ontology is describing, and there may be many different states of the world that are consistent with the ontology.
- Although, from the point of view of logic, there is no principal difference between different types of axioms, it is customary to separate them into three groups:
  - **Assertional (ABox)** axioms (e.g., Mother(Mary))
  - **Terminological (TBox)** axioms (e.g., Mother $\sqsubseteq$ Parent)
  - **Relational (RBox)** axioms (RBox axioms refer to properties of roles. As for concepts, DLs support role inclusion (e.g., parentOf $\sqsubseteq$ ancestorOf) and role equivalence axioms.)
  - **Note**: In role inclusion axioms, role composition can be used to describe roles such as uncleOf. Intuitively, if Bob is a brother of Mary and Mary is a parent of John, then Bob is an uncle of John. This kind of relationship between the roles brotherOf, parentOf and uncleOf is captured by the **complex role inclusion** axiom: brotherOf o parentOf $\sqsubseteq$ uncleOf .
  - In DLs we can write **disjoint roles** as follows: Disjoint(parentOf; childOf).
  - **RBox axioms include role characteristics** such as reflexivity, symmetry and transitivity of roles.

DL System Architecture/ DL Knowledge Base

Knowledge Base

Tbox (schema)

$\mathcal{T}$ **Knowledge about concepts of a domain**

Man $\equiv$ Human $\sqcap$ Male

Writer $\equiv$ Person $\sqcap$ $\exists$author.Book

Happy-Father $\equiv$ Man $\sqcap$ $\exists$hasChild.Female $\sqcap$ $\exists$hasChild.Male $\sqcap$ $\forall$hasChild.(Rich $\sqcup$ Happy)

Abox (data)

$\mathcal{A}$ **knowledge about individuals/ entities**

writer(Ranganathan)

author(Ranganathan, Prolegomena…)

Happy-Father(John)

hasChild(John, Mary)

RBox Role-centric Knowledge

$\mathcal{R}$ Knowledge about roles interdependencies

illustrator $\sqsubseteq$ contributor

Inference System

Interface

Description Logics: some concerns

- The capability of inferring additional knowledge increases the modelling power of DLs but it also requires
  - ***some understanding on the side of the modeller; and***
  - ***good tool support for computing the conclusions.***
- The computation of inferences is called reasoning.
- An important goal of DL language design has been to ensure that reasoning algorithms of good performance are available.
  - ***This is one of the reasons why there is not just a single description logic***.
  - The best balance between expressivity of the language and complexity of reasoning depends on the intended application.

Notation

- C and D be concepts
- a and b be individuals
- R be a role

Symbol	Description	Example	Read
$\top$	all concept names	$\top$	top
$\bot$	empty concept	$\bot$	bottom
$\sqcap$	*intersection* or *conjunction* of concepts	$C \sqcap D$	C and D
$\sqcup$	*union* or *disjunction* of concepts	$C \sqcup D$	C or D
$\neg$	*negation* or *complement* of concepts	$\neg C$	not C
$\forall$	*universal restriction*	$\forall R.C$	all R-successors are in C
$\exists$	*existential restriction*	$\exists R.C$	an R-successor exists in C
$\sqsubseteq$	Concept *inclusion*	$C \sqsubseteq D$	all C are D
$\equiv$	Concept *equivalence*	$C \equiv D$	C is equivalent to D
$\doteq$	Concept *definition*	$C \doteq D$	C is defined to be equal to D
:	Concept *assertion*	$a : C$	a is a C
:	Role *assertion*	$(a, b) : R$	a is R-related to b

DL Family

- Smallest deductively complete DL is *ALC* (Attribute Language with Complement)
  - Concepts constructed using the following class constructors: disjunction, conjunction and complement
    - $\sqcup$, $\sqcap$, $\neg$
  - **plus** restricted quantifiers $\exists$,$\forall$
    - Quantifiers restricts the domain and range of roles which helps in maintaining the decidability

Description Logic (DL) Family

There are many varieties of DL and there is an informal naming convention, roughly describing the operators allowed.

- $\mathcal{F}$ Functional properties.
- $\mathcal{E}$ Full existential qualification (Existential restrictions that have fillers other than `owl:thing`).
- $\mathcal{U}$ Concept union.
- $\mathcal{C}$ Complex concept negation.
- $\mathcal{S}$ An abbreviation for $\mathcal{ALC}$ with transitive roles.
- $\mathcal{H}$ 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`).
- $\mathcal{Q}$ Qualified cardinality restrictions (available in OWL 2, cardinality restrictions that have fillers other than `owl:thing`).
- $(\mathcal{D})$ Use of datatype properties, data values or data types.

- ❑ OWL 2 provides the expressiveness of SROIQ (D)
- ❑ OWL DL closely corresponds to SHOIN(D)
- ❑ OWL Lite closely corresponds to SHIF(D)

Nominals (singleton concepts), e.g., {India}

Wff and atomic formula

- Formula / well formed formula / wff
  - is a word (i.e. a finite sequence of symbols from a given alphabet) which is part of a formal language
  - A formula is a syntactic formal object that can be informally given a semantic meaning
- Atomic formula
  - An atomic formula is a formula that contains no logical connectives nor quantifiers, or equivalently a formula that has no strict sub formulas.
  - The precise form of atomic formulas depends on the formal system under consideration for propositional logic,
  - E.g., the atomic formulas are the propositional variables.
  - For predicate logic, the atoms are predicate symbols together with their arguments, each argument being a term

AL (Attributive language) Logical Symbols

- Formation rules:

  <Atomic> ::= A | B | ... | P | Q | ... | ⊥ | ⊤

  <wff> ::= <Atomic> | ¬<Atomic> | <wff> ⊓ <wff> | ∀R.C | ∃R.⊤

  NOTE: no ⊔, ∃R.⊤ = limited existential quantifier, ¬ on atomic only

- Person ⊓ Female

  "persons that are female"

- Person ⊓ ∀hasChild.⊤
  "(all those) persons that have a child"

- Person ⊓ ∀hasChild.⊥
  "(all those) persons without a child"

- Person ⊓ ∀hasChild.Female
  "persons all of whose children are female"

ALU (AL with disjunction)

- Formation rules:

  <Atomic> ::= A | B | ... | P | Q | ... | ⊥ | ⊤

  <wff> ::= <Atomic> | ¬<Atomic> | <wff> ⊓ <wff> | ∀R.C | ∃R.⊤ |

  <wff> ⊔ <wff>

- Father ⊔ Mother

  “the notion of parent”

ALE (AL with extended existential)

- Formation rules:

  $\langle Atomic\rangle ::= A \mid B \mid \ldots \mid P \mid Q \mid \ldots \mid \bot \mid \top$

  $\langle wff\rangle ::= \langle Atomic\rangle \mid \neg\langle Atomic\rangle \mid \langle wff\rangle \sqcap \langle wff\rangle \mid \forall R.C \mid \exists R.\top \mid$

  $\exists R \mid \exists R.C$

- ∃R (there exists an arbitrary role)
- ∃R.C (full existential quantification)

- Parent ⊓ ∃hasChild.Female

  “parents having at least a daughter”

ALN (AL with number restriction)

- Formation rules:

  <Atomic> ::= A | B | ... | P | Q | ... | ⊥ | T

  <wff> ::= <Atomic> | ¬<Atomic> | <wff> ⊓ <wff> | ∀R.C | ∃R.⊤ |

  ≥nR | ≤nR

- ≥nR (at-least number restriction)
- ≤nR (at-most number restriction)

- Parent ⊓ ≥2 hasChild

  "parents having at least two children"

ALC (AL with full concept negation)

- <u>Formation rules</u>:

  <Atomic> ::= A | B | ... | P | Q | ... | ⊥ | ⊤

  <wff> ::= <Atomic> | ¬ <wff> | <wff> ⊓ <wff> | ∀R.C | ∃R.⊤

- ¬ (Mother ⊓ Father)

  “it cannot be both a mother and father”

ALC (AL with full concept negation)

The DL language ALC (Attributive Language with Concept negation) contains the following elements:

- **Concepts** denoting entity types/classes/unary predicates/universals, including top $\top$ and bottom $\bot$;
- **Roles** denoting relationships/associations/n-ary predicates/properties;
- **Constructors**: and $\sqcup$, or $\sqcap$, and not $\neg$; quantiers forall $\forall$ and exists $\exists$
- **Complex concepts** using constructors: Let C and D be concept names, R a role name, then
– $\neg$ C, C $\sqcap$ D, and C $\sqcup$ D are concepts
- $\forall$R.C and $\exists$R.C are concepts
- - Individuals

Examples in ALC

Some examples that can be represented in ALC are:

- Concepts (primitive, atomic): Book, Course
- Roles: Enrolled, Reads
- Complex concepts:
- Student $\sqsubseteq$ $\exists$Enrolled.(Course $\sqcup$ DegreeProgramme)
- Mother $\sqsubseteq$ Woman $\sqcap$ $\exists$parentOf.Person
- Parent $\equiv$ (Male $\sqcup$ Female) $\sqcap$ $\exists$parentOf.Mammal $\sqcap$ $\exists$caresFor.Mammal
- Individuals in the ABox: Student(Bob), Mother(Mary), $\neg$Student(John), ENROLLED(Bob; COMP101)

DL Semantics

- The semantics of description logics are defined by interpreting concepts as sets of individuals and roles as sets of ordered pairs of individuals.
- Those individuals are typically assumed from a given domain.
- The semantics of non-atomic concepts and roles is then defined in terms of atomic concepts and roles.
    - This is done by using a recursive definition similar to the syntax.

DL Semantics: Interpretation function (*I*)

- Intuitively, a model is a situation
- A situation is a *semantic* entity, providing us with a certain amount of things we can talk about.
- A model for a given vocabulary gives us two pieces of information:
  - tells us what kind of collection of entities (usually called the domain) we can talk about
  - for each symbol in the vocabulary, it gives us an appropriate semantic entity, built from the items in
  - this task being carried out by a function (interpretation function) which, for each symbol in the vocabulary, specifies an appropriate semantic value

DL Semantics

- Semantics given by standard FO model theory
- The vocabulary is the set of names (consist of concepts and roles)
  - we use in our model of (part of) the world
  - E.g., {Tree, Cow, Dog, Animal, Person, Car, University, ...}
- A *Terminological interpretation I* is a tuple (Δ^I, $.^I$) over a signature (N_C, N_R and N_O) consists of
  - Domain Δ is a non-empty set of objects
  - **Interpretation:** $.^I$ is the interpretation function, domain Δ^I
  - $.^I$ maps every concept name C (names of unary predicates (classes/concepts)) to a subset $C^I \subseteq \Delta^I$
  - $.^I$ maps every role name R (names of a binary predicate (properties/roles)) to a subsets of $R^I \subseteq \Delta^I \times \Delta^I$
  - $.^I$ maps every individual a to elements of Δ^I : $a^I \in \Delta^I$
  - **Note**: $\top^I = \Delta^I$ and $\bot^I = \varnothing$

DL Semantics (ALC)

Using the typical notation where C and D are concepts, R a role, and a and b are individuals, then they have the following meaning, with on the left-hand side of the "=" the syntax of ***ALC*** under an **interpretation and on the right-hand side its semantics**:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$\bot^{\mathcal{I}} = \emptyset$$

$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$ (Union means disjunction)

$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$ (Intersection means conjunction)

$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ (Complement means negation)

$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \text{for every } y, (x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$$

$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | \text{there exists } y, (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$$

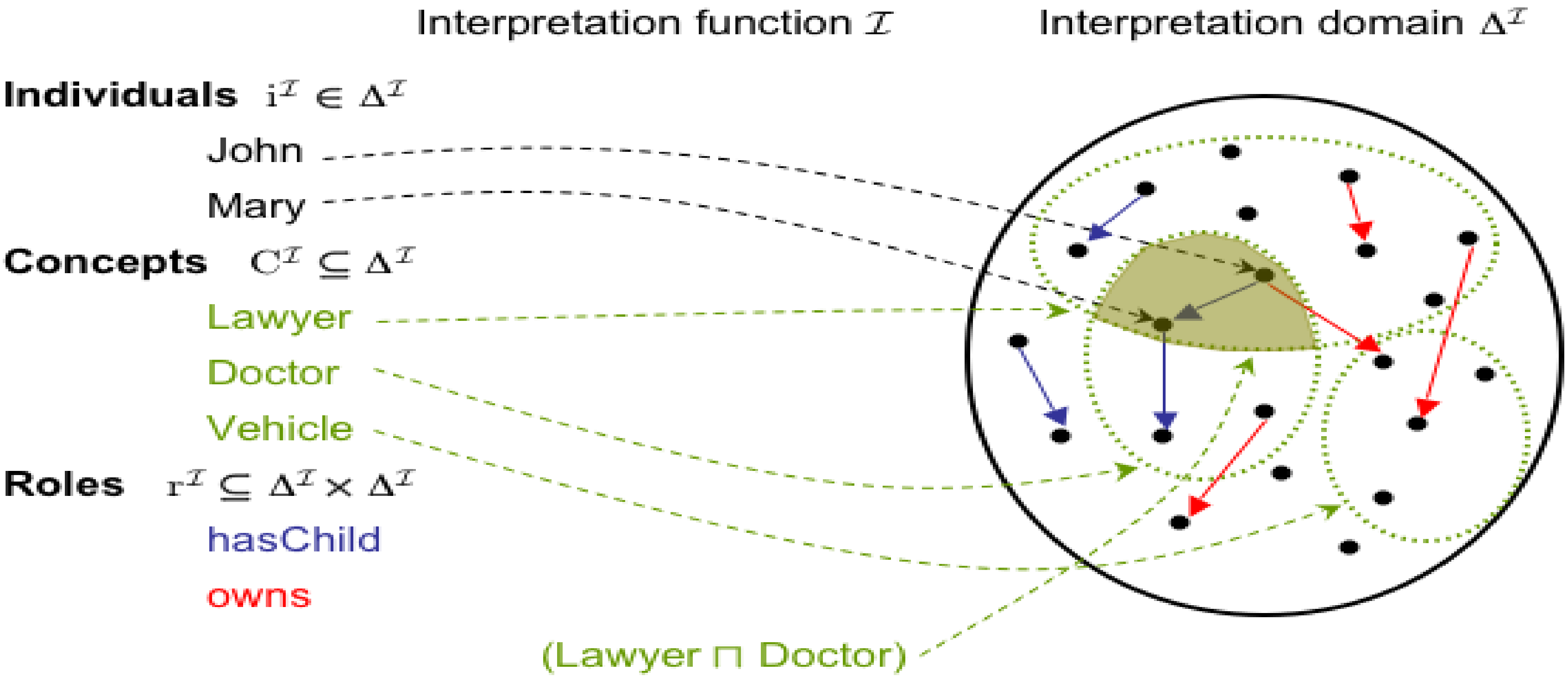
Based on the above, we can specify the notion of ***satisfaction****:* $\mathcal{I} \models$ (read in *I holds*)

- 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

***MODEL**: a model is an abstraction of a state of the world that satisfies all axioms in the ontology. An ontology is consistent if it has at least one model.

DL Semantics

- Well defined (model theoretic) **semantics**



AL’s extensions and sub-languages

- *The basic Description Language is AL*
- By extending *AL* with any subsets of the above constructors yields a particular DL language.
- Each language is denoted by a string of the form *AL*[*U*][*E*][*N*][*C*], where a letter in the name stands for the presence of the corresponding constructor.
- *ALC* is considered the **most important** for many reasons.

  NOTE: $ALU \subseteq ALC$ and $ALE \subseteq ALC$
- By eliminating some of the syntactical symbols and rules, we get some sub-languages of *AL*
  - The most important sub-language obtained by elimination in the AL family is ClassL
  - We also have FL^- and FL_0 (where FL = frame language)

Summary: *AL* and extensions

Constructor	Syntax	Semantic	
Atomic concept	A	$A^I \subseteq \Delta^I$	AL
Atomic role	R	$R^I \subseteq \Delta^I \times \Delta^I$	
Conjunction	$C \cap D$	$C^I \cap D^I$	
Existent quantification	$\exists R.T$	$\{d \mid$ it exists a $e \in \Delta^I$, $(d,e) \in R^I\}$	
Universal quantification	$\forall R.C$	$\{d \mid$ for all $e \in R^I$ it follows $e \in C^I\}$	
Transitive role	$R \in R_+$	$R^I = (R^I)^+$	
Complement	$\neg C$	$\Delta^I \backslash C^I$	C
Disjunction	$C \cup D$	$C^I \cup D^I$	U
Existent restriction	$\exists R.C$	$\{d \mid$ it exists a $e \in \Delta^I$, $(d,e) \in R^I$ and $e \in C^I\}$	E
Role hierarchy	$R \subseteq S$	$R^I \subseteq S^I$	
Inverse role	R^-	$\{(d,e) \mid (e,d) \in R^I\}$	
Value restriction	$> nR$ $< nR$	$\{d \mid \#\{e \mid (d,e) \in R^I\} > n\}$ $\{d \mid \#\{e \mid (d,e) \in R^I\} < n\}$	N
Qualified value restriction	$> nR.C$ $< nR.C$	$\{d \mid \#\{e \mid (d,e) \in R^I$ and $e \in C^I\} > n\}$ $\{d \mid \#\{e \mid (d,e) \in R^I$ and $e \in C^I\} < n\}$	Q
Functional restriction	$< 1R$	$\{d \mid \#\{e \mid (d,e) \in R^I\} < 1\}$	F
Nominals (One-off)	$\{d_1 \dots d_n\}$	$\{d^I_1 \dots d^I_n\}$	O

Complext Results of Description Logics. W. Gong, D. Zhang and J. Zhao. In High Performance Networking, Computing, and Communication Systems (Communications in Computer and Information Science (CCIS), Springer, 2011)

AL's Contractions: FL^{-} and FL_0

- FL^{-} a sub-language of *FL*, which is obtained by disallowing role restriction.
  - This is equivalent to *AL* without atomic negation
- FL_0 is *a* sub-language of FL^{-}, which is obtained by disallowing limited existential quantification

- FL^{-} *is AL* with the elimination of ⊤, ⊥ and ¬
- Formation rules:

  <Atomic> ::= A | B | ... | P | Q | ...

  <wff> ::= <Atomic> | <wff> ⊓ <wff> | ∀R.C | ∃R.⊤
- FL_0 *is FL-* with the elimination of ∃R.⊤
- Formation rules:

  <Atomic> ::= A | B | ... | P | Q | ...

  <wff> ::= <Atomic> | <wff> ⊓ <wff> | ∀R.C

Limitation of DL

Being a fragment of first order predicate logic, the DL cannot express the following:

•Fuzzy expressions - “It **often** rains in autumn.”

•Non-monotonicity - “Birds fly, penguin is a bird, but penguin does not fly.”

•Propositional attitudes - “Eve **thinks** that 2 is not a prime number.” (It is true that she thinks it, but what she thinks is not true.)

Source: OWL 2 and SWRL Tutorial. https://dior.ics.muni.cz/~makub/owl/