An opinionated look at

Description Logics

Alex Borgida
Dept. of Computer Science
Rutgers University

Outline

Motivation

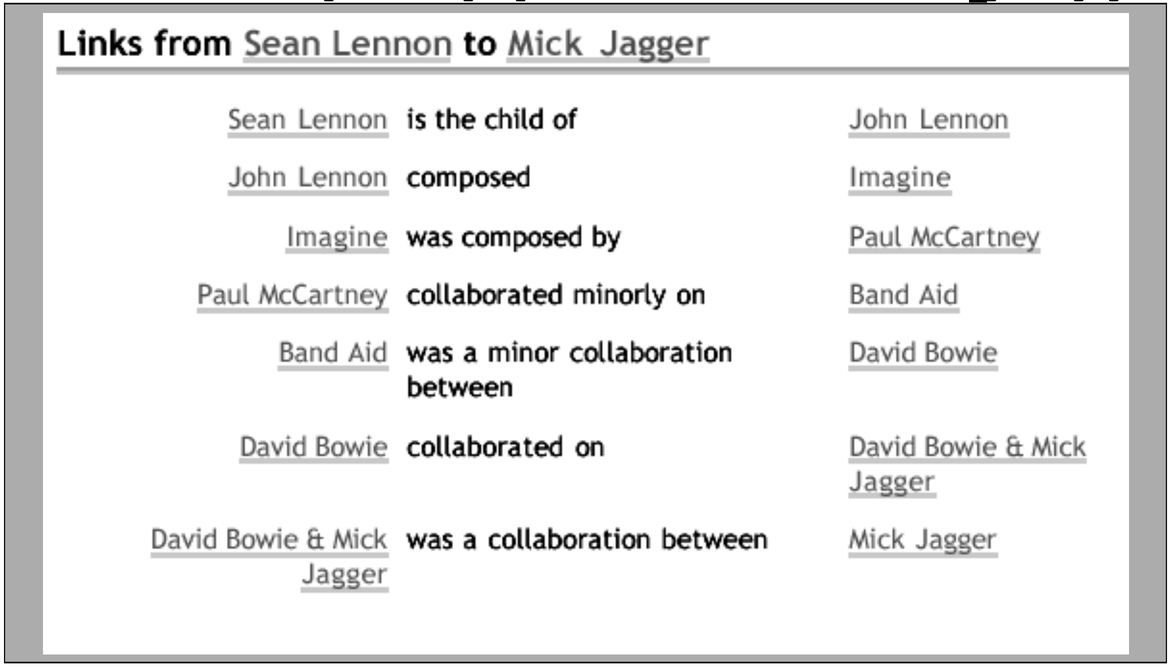
***Conceptual models** are needed in*

- ***artificial intelligence** (meaning of natural language sentences, representing knowledge in general)*
- ***database design** (Entity Relationship diagrams)*
- ***software engineering** (requirements, UML)*
- ***in the age of the Internet:***
  - ***information integration***
  - ***finding and composing web services***

Motivation (cont'd)

"How are Sean Lennon and Mick Jagger connectd?"

`http://www.pumpthemusic.com/oracle/index_post.php`

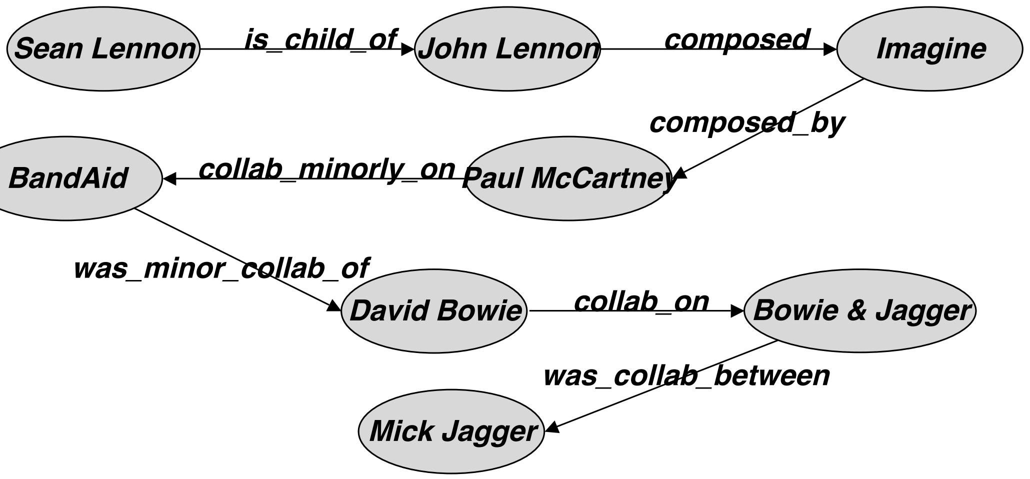


Links from Sean Lennon to Mick Jagger

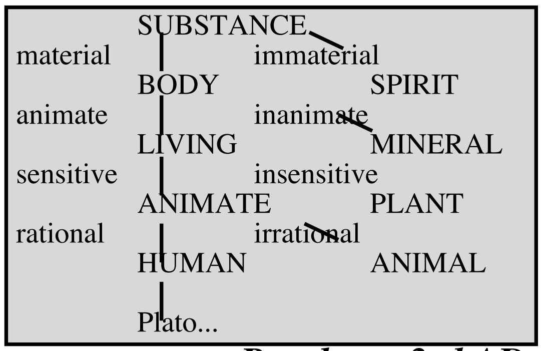
Sean Lennon	is the child of	John Lennon
John Lennon	composed	Imagine
Imagine	was composed by	Paul McCartney
Paul McCartney	collaborated minorly on	Band Aid
Band Aid	was a minor collaboration between	David Bowie
David Bowie	collaborated on	David Bowie & Mick Jagger
David Bowie & Mick Jagger	was a collaboration between	Mick Jagger

Graphical representation

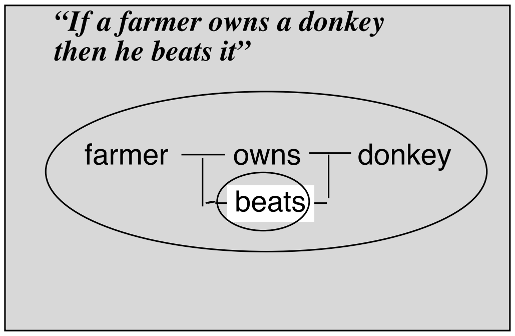
Sean Lennon	is the child of	John Lennon
John Lennon	composed	Imagine
Imagine	was composed by	Paul McCartney
Paul McCartney	collaborated minorly on	Band Aid
Band Aid	was a minor collaboration between	David Bowie
David Bowie	collaborated on	David Bowie & Mick Jagger
David Bowie & Mick Jagger	was a collaboration between	Mick Jagger



Not entirely new idea



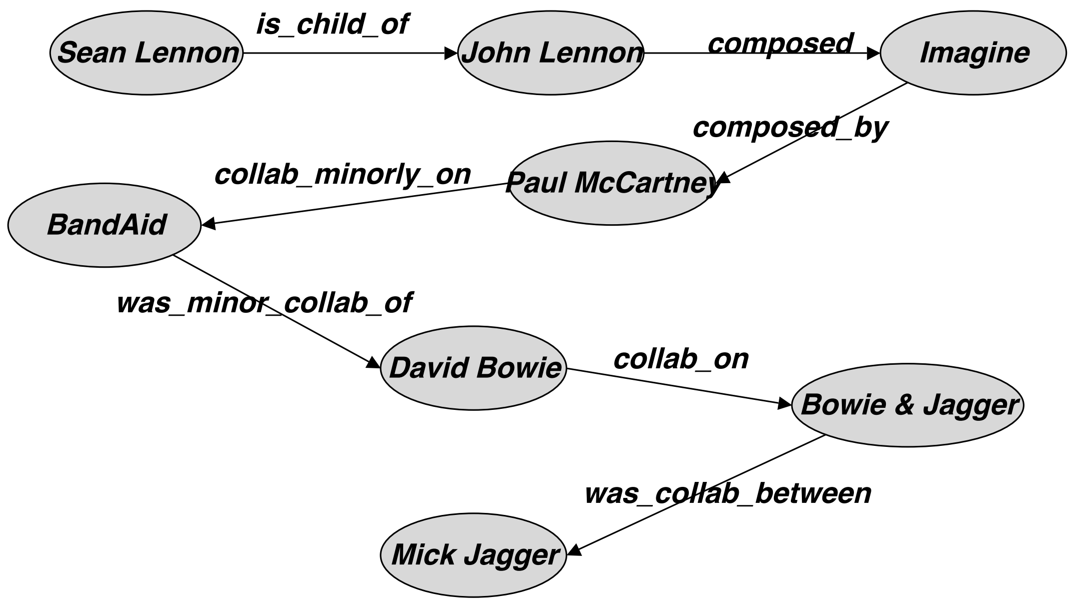
Porphyry 3rd AD.



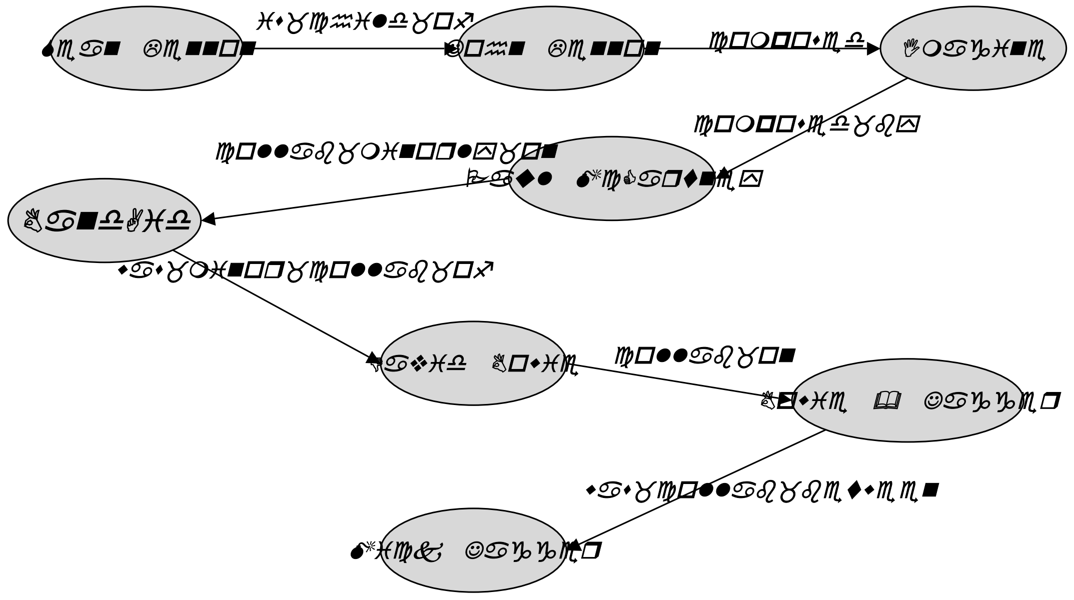
C.S.Peirce 1890's

SEMANTIC NETWORKS
in Artificial Intelligence/Cognitive Science
Quillian 1966
...

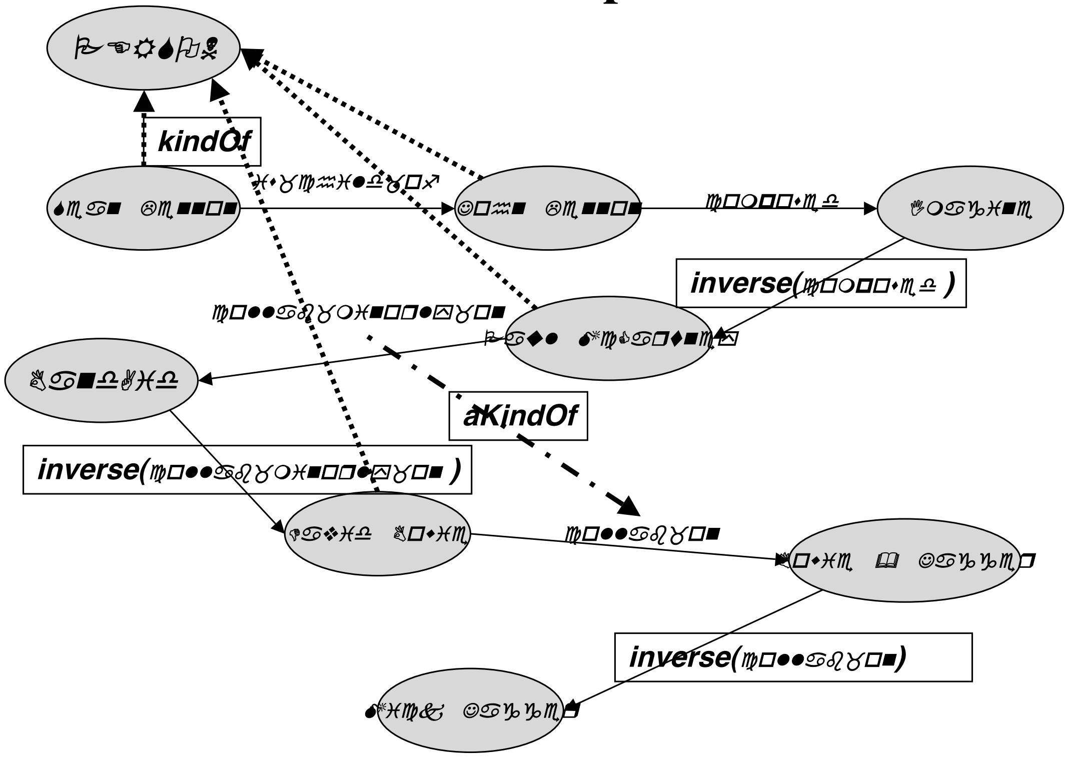
What we say to computers



What computers “hear”



What we would like computers to understand



Outline

1. Motivation
2. >> Fundamental notions of DLs + syntax
3. Specification of reasoning + some formal properties
4. Applications of DLs

Description Logics

- **A *precise* notation for representing "noun phrases"**
  [Brachman 70's: KL-ONE]

Fundamental ontology: conceptual model is populated by

- *individuals*
- related by binary relationships (called *roles & features*)
- grouped into classes (*concepts*)

So we need the ability to describe concepts, relationships, individuals.

First Order Logic would be fine, but it is impossible to reason with it decidably.

Description Logics (cont'd)

Fundamental observation 1: In addition to **primitive** concepts, such as **PERSON , CHAIR**, ... there are **defined** concepts

- some have names:
  - ***"person with age between 13 and 17"*** **≡ TEENAGER**
  - ***"person who eats only non-meat foods"*** **≡ VEGETARIAN**
- others are describable only by relative clauses or compound nouns:
  - ***"person who has at least 3 children"***
  - ***"towns located in MA or NH or VT,.."***
    (NEW_ENGLAND_TOWNS)

Description Logics (cont'd)

Fundamental observation 2: Both primitive and defined concepts can have additional assertions made about them, representing *necessary* conditions.

A standard way to make such assertions is to use

is-a / is-subconcept-of / is-subsumed-by / is-a-kind-of

PERSON is-a ANIMATE
PERSON is-a ("age having an integer value")
TEENAGER is-a LIKES_MTV

Description Logics (cont'd)

We need a language for defining concepts. ***(Based on empirical experience on what has been useful in many applications):***

- ***atomic/primitive concepts:*** **PERSON, COURSE, BOOK**
- ***boolean combinations of these:***
  - **AFRICAN <u>and</u> HERBIVORE**
  - **PERSON <u>or</u> CORPORATION**
- ***concepts defined by enumeration of individuals:*** {Masc,Fem}
- ***concepts from "concrete domains"*** *(numbers, strings, ...)*
- ***primitive binary relationships*** **graduateOf, locatedIn, likes, hasPart**
- ***sets of objects satisfying restrictions on their role fillers***
  - ***objects <u>all</u> of whose*** **locatedIn** ***values are in*** **NEW_ENGLAND_TOWNS**
  - ***objects <u>some</u> of whose*** **graduateOf** ***values are in*** **UNIVERSITY**
  - ***objects with <u>at least</u> 3*** **hasPart** ***fillers***
  - ***objects whose*** **firstName** ***<u>same as</u>*** **father'*s* firstName**
  - ***objects whose*** **name** ***values <u>include</u>*** "Jr."

Description Logics - syntax (1)

Just like {**and,or,not**} are logical *formula* constructors, DLs offer ***concept constructors***. Will use term/prefix notation here:

• **AFRICAN and HERBIVORE**	·`and(AFRICAN, HERBIVORE)`
• **not ANIMATE**	·`not(ANIMATE)`
• **PERSON or CORPORATION**	·`or(PERSON, CORPORATION)`
• **PERSON and not TEEN**	·`and(PERSON, not(TEEN))`
• **{Masc,Fem}**	·`enum(Masc , Fem)`
• ***(numbers, Progr.Lang. values)***	·`INTEGER`
• ***objects whose* locatedIn *values are only in NEW_ENGLAND_TWN***	· `all(locIn,NEW_ENGLAND_TWN)`
• ***objects some of whose* graduateOf *values are in* UNIVERSITY**	·`some(graduateOf,UNIVERSITY)`
• ***objects with at least 3* hasPart *fillers***	·`at-least(3,hasPart)`
• **name *value is identical to* father's**	·`same-as([name],[father name])`

Description Logics -syntax

Can describe concepts of arbitrary complexity by nesting. (Unlike OO, etc. no need to name concepts)

"Courses taken by 60 to 90 students, who are all juniors or seniors, and taught by a CS professor"

```
· and(
      COURSE
      at-least(60, takers)
      at-most(90, takers)
      all(takers,  and(STUDENT
                       all(inYear , enum(3, 4))))
      exactly(1, taughtBy)
      all( taughtBy, and(PROFESSOR
                          fills(inDepartment , "CS"))))
```

Description Logics -*syntax variants*

"Persons who eat only non-meat stuff"

- `(and PERSON  (all eats (not MEAT)))`
- **PERSON ⊓ ∀eats.¬MEAT**

-
  ```
  <concept> <and>
            <primitive name="PERSON"/>
            <all>
                  <primrole name="eats"/>
                  <not> <primitive name="MEAT"/> </not>
            </all> </and> </concept>
  ```

-
  ```
  <owl:intersectionOf  rdf:parseType="Collection">
     <owl:Class  rdf:about="#PERSON" />
     <owl:Restriction>
        <owl:onProperty  rdf:resource="#eats" />
        <owl:allValuesFrom>
            <owl:complementOf  rdf:resource="#MEAT" />
        </owl:allValuesFrom/>
     </owl:Restriction>
   </owl:intersectionOf>
  ```

Description logics: roles/properties

DL Fundamental observation 3: Relationships are like concepts. Hence they can also be structured and defined, using ***role constructors***.

- *loves* is-a-kind-of *likes*
  `loves` ***`is-a`*** `likes`
- *childOf* is the inverse of *parentOf*
  `inverse(``parentOf)`
- *descendantOf* is the transitive closure of *childOf*
  `trans(``childOf)`
- *nephewOf* is the composition of *sonOf* and *siblingOf*
  `compose(``sonOf,siblingOf)`

Concept/Description *Languages:* summary

- Descriptions are composite, variable-free *terms*, which can be recursively built up from primitive symbols, using *constructors*
- There are constructors for both concepts and roles (binary relationships)
- There is a collection of constructors that have been empirically found useful over the years

So what can one do with descriptions?

Standard *"judgements"* about Descriptions

1. Does C <u>subsume</u> D? **D :< C** **D $\sqsubseteq$ C**

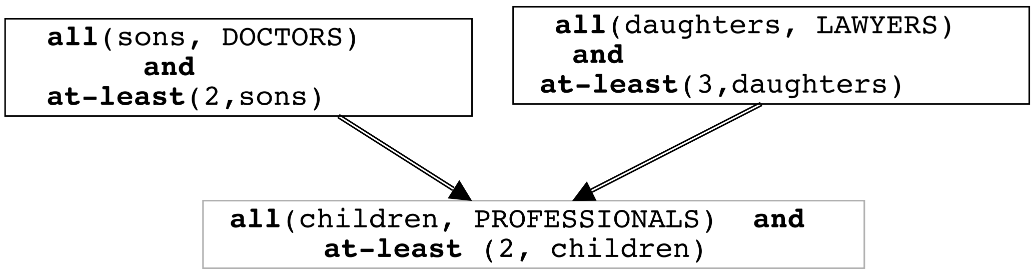
```
– and(PERSON, MALE) :< PERSON
– at-least(3, hasChildren):<at-least(1,hasChildren)
– and(all(p,C) , all(p,D))  :<  all(p , and(C,D))
– fills(loves, Eve)  :< at-least(1, likes)
```

2. Is concept C <u>incoherent</u>?

```
▪ and(PERSON
        at-least(3, hasDegree)
        all(hasDegree , enum("BA", "BS" ) )
```

Non-standard judgements

3. What is the <u>least common subsumer</u> of concepts C and D:
`lcs(C,D)` in the (infinite) lattice of all description terms!!! ***[B] (Useful in machine learning.)***



```
all(sons, DOCTORS)
      and
at-least(2,sons)
```

```
all(daughters, LAWYERS)
  and
at-least(3,daughters)
```

```
all(children, PROFESSIONALS)  and
      at-least (2, children)
```

NOTE: contrast with FOPC, where disjunction makes this pointless.

4. <u>Matching/Unification</u> ***[B] (Useful in printing relevant aspects)***

e.g., matching **`all`**`(hasParts, `**`?Y)`** against **`ARCH`** yields **`?Y ↔ BLOCK;`** But macthing is against "semantic complection" of **`ARCH`** **!**

How does one use DLs?

- **(A specific DL consists of a particular set of concept & role constructors)**
- **Then create a theory $\mathcal{T}$ of subsumption and definition assertions (or other kinds of assertions**

  e.g., `A` **`disjoint_from`** `B` ≡ **`and(A,B) :<`** $\bot$

  $\mathcal{T}$ is usually called a T-box ("ontology", "knowledge base")
- **As part of creating $\mathcal{T}$, concepts in it are**
  - automatically pre-classified into a subsumption hierarchy
  - tested for "reasonableness" (satisfiable)
- **$\mathcal{T}$ can then be queried to see if it entails other judgements**

DLs and individuals/nominals

- **Two new judgements**

```
Mimi : HAPPY              ind membership
sisterOf(Anna,Mimi)       roles relating inds
```

- **Create a theory *A* of assertions about individuals, usually called an *A-box ("database")***
- **As part of creating *A*, individuals in it are (often)**
  - automatically pre-classified under the *most specific named concept* in T-box taxonomy
  - tested for "reasonableness" (satisfiable)
  - some propagations cached
- ***A* can then be queried to see if it entails other judgements**

Sample Individual Reasoning

- ***Assertions*: *individuals can be asserted to satisfy descriptions***

```
Calvin : PERSON
Calvin : all(friendOf, the(age and(min(5),max(7))))
```

Open World Assumption

- ***Consistency checking*: *given additional assertion***

```
friendOf(Calvin, Susie)
```

verify that* Susie's age *is not known to be under 5 or over 7

- ***Propagation* -- *if* Susie's age *is not known, then infer partial information***

```
Susie : the(age ,  and(min(5),max(7)) )
```

- ***Individual Classification* -- *in either case, if we have a definition like***

CHILD $=_{def}$ **the**(age , **and**(**min**(0),**max**(12)))

then* Susie *is inferred to be a child

```
Susie : CHILD
```

Outline

Expressive power

- **Even the most expressive DL ever proposed =**
  FOL + counting quantifiers + fix point <u>but only 3 variable symbols</u>
  - so cannot represent 4-clique

```
happy(X) :- likes(X,Y1),likes(X,Y2),likes(Y1,Y2),...
```

- **But open-world assumption, ALL-restrictions, definitions, put it beyond Datalog**

- **Subsets of DL are variants of**
  - modal logic K
  - Propositional Dynamic Logic
  - Guarded Fragment of FOL

Some well known DLs

- **Classic (early 1990's, AT&T Bell Labs** *[B]*
  - low-order polynomial time reasoning
  - used in industrial application at AT&T to configure switching equipment
- **FaCT$\mathcal{SHIQ}$ (late 90's, Manchester)**
  - optimized tableaux implementation
  - used for large (5000 concept) medical ontology, which is not just a tree
  - although logic is EXPTIME-complete, in practice not a problem!?
- **OWL-DL**
  - the ontology language of the semantic web
  - $\mathcal{SHOIQ}$(C)

Some complexity results

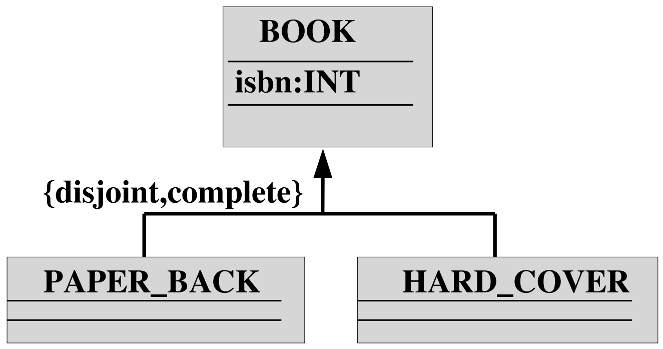
Constructors	**T-Box** (prim :< D)	(D :< C)	**cyclic**	**Subsumes?**	**Member?**
$\mathcal{AL}$(`and,all`)				*O(n²)*	
$\mathcal{AL}$	*			co-NP-complete	
CLASSIC with host individuals				*O(n³)*	
$\mathcal{ALE}$*(and,all,some)*				NP-compl.	PSPACE
$\mathcal{ALC}$ *(and,all,not)*				PSPACE-complete	
$\mathcal{ALC}$*(and,all,not)*		*		EXPTIME-complete	
$\mathcal{ALCNR}$(r-and,nrs)				PSPACE	PSPACE
$\mathcal{ALCNR}$,$\mathcal{SHIQ}$ NEXPTIME		*	*	*	NEXPTIME
$\mathcal{ALCQ}$, $\mathcal{ALCN}$*+complex roles but not* `r-and`				EXPTIME-complete	
$\mathcal{AL}$ & role `same-as`				*undecidable*	
$\mathcal{AL}$ *& func'n role* `same-as`			*	*poly-time*	

Outline

1. **Motivation**
2. **Fundamental notions of DLs**
3. **Syntax, semantics, some formal properties**

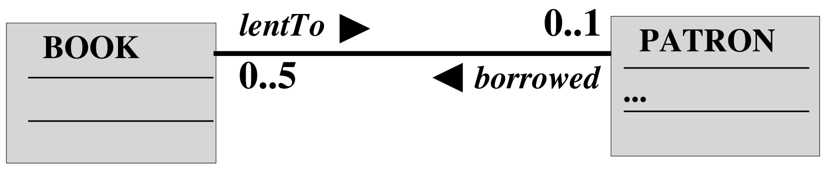
>> 4. **Application of DLs**
   - (representing UML class diagrams-- hence reasoning about consistency)
   - describing e-services/programs

Representing UML in *SHIQ / DL-lite*



```
BOOK :< the(isbn, INT)
PAPER_BACK :< BOOK
HARD_COVER :< BOOK
BOOK  :< or(PAPER_BACK,HARD_COVER)      ;;complete
and(PAPER_BACK,HARD_COVER) :< NOTHING  ;;disjoint
```

Representing UML in *SHIQ / DL-lite*



```
BOOK :< all(lentTo,PATRON)  and   at-most(1,lentTo)
borrowed =def= inverse(lentTo)
PATRON :< all(borrowed,BOOK) and at-most(5,borrowed)
```

An application: e-service description [B]

CORBA interface:

```
interface CAR{
 attrib CAR-MODEL model;
 attrib OWNER ownedBy;
 attrib MANUFACT madeBy;
 ...

 deliver( in MANUFACT src,
         in DEALER dest,
         in DATE time
        )signals (BadDealer);
 sell(...);
 destroy(...);
```

1. Create class for CAR with attributes and methods as properties:

```
CAR :<
      (model some   CAR_MODELS)
      (ownedBY some OWNER)
      (madeBy some MANUFACT)
      (deliver some  DELIVER)
```

SE application: e-service description

CORBA interface:

```
interface CAR{
 attrib CAR-MODEL model;
 attrib OWNER ownedBy;
 attrib MANUFACT madeBy;
 ...

 deliver( in MANUFACT src,
          in DEALER dest,
          in DATE time
         )signals (BadDealer);
 sell(...);
 destroy(...);
```

2. Reify methods, to describe parameters as attributes

```
DELIVER :<
      ACTION and
      (this some CAR)
      (src some MANUFACT)
      (dest some DEALER)
      (time some DATE)
```

SE application: e-service description

```
DELIVER :<
      ACTION and
      (this some CAR)
      (src some MANUFACT)
      (dest some DEALER)
      (time some DATE)
```

3. Describe service semantics by giving pre- and post-conditions, conditions for exceptions,...

```
CAR :<
      (model some   CAR_MODELS)
      (ownedBY some OWNER)
      (madeBy some   MANUFACT)
      (deliver some   DELIVER)
        //preconds include
            (madeBy same-as  deliver.src)
        //postconds include
            (ownedBy  same-as deliver.dest)
        //exception BadDealer signalled when
            (not (src overlaps dest.represents))
```

Pros and Cons of DLs

Pros

- **Has been found empirically useful to describe "natural" domains we talk about (can represent and reason with ER and UML diagrams)**
- **"Open World Assumption" helps with reasoning in the presence of incomplete knowledge**
- **Syntax avoids variables, quantifiers, and supports nested complex concepts without having to name them**
- **Distinguishes *definitions* from *primitive concepts*, and applies uniformly to relationships and concepts**
- **Intermediate in expressive power between propositional and full First Order Predicate Calculus**
- **Well-explored complexity picture for many combinations of constructors**

Pros and Cons of DLs

Cons

- **Expressive limitation:** 3FOL + fixed point logic
- **Poor at describing mathematical concepts** (algebraic equations and reasoning with them)
- **Cannot express even conjunctive queries (non-recursive Datalog)**
- **Vast majority of 'ontologies' being built are simple** (simple hierarchies of terms (e.g., DMOZ, Yahoo)**,** or at most UML)**. For these, OWL is overkill**

References

- ***Description Logic Handbook***, F. Baader et al, Cambridge Press, 2003
- Annual Description Logic workshops (20 so far). Electronic proceedings on web -- search for

`dblp DL 2006`

***some [Borgida...] papers*:**

- **"CLASSIC: A Structural Data Model for Objects",** ***SIGMOD Database Conf. 1989*** (*with* R. Brachman, D. McGuinness, L. Alperin Resnick)
- **"Description Logics in Information Management",** ***IEEE Trans. Knowl. & Data Engineering (*1995*)***
- **"On the Relative Expressiveness of Description Logics and Predicate Logics",** ***Artif. Intelligence Journal*** **(1996)**
- **"Adding more 'DL' to IDL: Towards More Knowledgeable Component Inter-Operability",** ***Int. Conf. on Software Engineering (ICSE) 1999*** (*with* Prem Devanbu)
- **"Explaining *ALC* subsumption",** ***ECAI'2000,*** (*with* E. Franconi, I. Horrocks, D. McGuinness)
- **"Distributed Description Logics",** ***Journal of Data Semantics 1(1)*****, 2004,** *(with* L.Serafini)
- **"On Concept Similarity" (DL'2006)** (*with* T.Walsh, H.Hirsh)
- **"Importing Knowledge from T-Boxes" (DL'2007)**

Outline

1. **Motivation**
2. **Fundamental properties of DLs**
3. **Syntax and reasoning with DLs**
4. **Using DLs in Information Management**
5. **Importing knowledge from DL KBs**

On Importing Knowledge

- It is important to **reuse** knowledge from previous KBs when building new ones.
- Study the notion

  "KB1 imports identifiers S={N,...} from KB$_{exp}$"

Basic Desiderata:

- behave as if all of KB$_{exp}$ was included in KB1
- *but* minimize import to make understanding easier and reasoning faster
- accept possibly additional names & axioms imported, not just *S*
  - *S*={Dog,Cat},
  - Dog :< Carnivore :< Animal, Cat :< Carnivore

On Importing Knowledge

"KB1 imports identifiers *S*={N,...} from KB$_{exp}$"

Approach 1: based on the notion of "module"

– **KB$_{exp}$** partitioned into modules ***M1***,... which are exported *a priori.*
– Each needed module is then imported as a unit (so imported concept name N comes with everything in its module)

I. Modules are created by hand, by the developer

II. **Automatic** modularization
- based on more or less syntactic (graph theoretic) grounds
- based on logical properties

On Importing Knowledge

"KB1 imports identifiers S={N,...} from KB$_{exp}$"

Approach 2: Use list S of names to customize material imported

III. Define and compute *import(KB1,S,KB$_{exp}$)*

IV. Use names in S to write special axioms ("bridge rules") connecting KB1 and KB$_{exp}$, and treat KB1 and KB$_{exp}$ as independent, communicating sources

Defining *import(S,KB2)*

Borgida [DL'07,WOMO'07]
Grau et al [WWW'07]
Issues

- **Axioms imported form a subset of**
  1. theorems(KB$_{exp}$)
  2. KB$_{exp}$
  3. expanded(KB$_{exp}$)
     - **to deal with dependence on syntax, avoid irrelevant material**
- **How to define "minimal amount of knowledge to be imported"**

$\varphi \;\mid \text{vocab}(KB \cup \{\varphi\}) \cap \text{vocab}(KB2) \subseteq S$ and $KB \cup KB2 \models \varphi$

  – just for this importing KB? or for all possible ones?
- **Influence of importing KB**
  – limit the places where symbols from S can appear (this may limit the set of axioms that need to be brought)

Computing *import*

- Even in very simple cases (hierarchies with disjointness), cost of *minimizing* makes problem co-NP hard
- [Grau et al] have syntactic condition on KB_{exp} ("locality") which allows import to be found effectively
- In general, problem related to "conservative extensions", and is hard

IV. Multi-logics with "connections"

Local Semantics

1. DDL (Distributed DL)
2. E-connections
3. P-DL
4. [Stuckenschmidt&Klein ISCW 04]

Characteristic:

- denotational semantics does not assume the same domain of interpretation for all ontologies

Distributed Desription Logics

Borgida & Serafini [J of Data Semantics 2004]
Serafini, Borgida & Tamlin [IJCAI 2005]

GIVEN: $\langle$ T1, T2, { T2 imports T1\$A} $\rangle$ + <u>very restricted use of these imported names in T2</u>! Only in axioms of the form

H $\sqsubseteq$ 1:A **/*A onto H*/** 1:B $\sqsubseteq$ G **/*B into G*/**

(and actually, $\sqsubseteq$ is not real subsumption: it is mediated by *domain relation* $\mathbf{r}_{12}$ connecting Domain1 to Domain2

1:teamA |---> {2:Pele, 2:Julinho,...}

RESULTS:

• *specification* of DDL entailment
$\langle$ T1,T2,imports $\rangle \models_{ddl}$ 2: E F $\sqsubseteq$
• *implementation* as distributed tableaux theorem prover
• *fixed point characterization* using H :< G1 $\vee$ ... $\vee$ Gn derived from bridge axioms and T1

E-connections

Grau, Parsia, & Sirin [ISWC 2004]

Analogy to DDL:

GIVEN: $\langle$ T1, T2, { T2 imports concept T1\$A} $\rangle$ + *R*
<u>Somewhat restricted use of these imported names in T2</u>!

Imported concepts can only be used in T2 to create new <u>restrictions</u> on the special roles in *R,* using a specific set of constructors. (But once defined, such concepts can be used anywhere in T2.)

RESULTS: *spec and implementation* for OWL-DL importers

➢Can *simulate DDL* by using *R* = { $\mathbf{r}_{12}^{-}$ }
"into": T1\$A :< ($\forall$ $\mathbf{r}_{12}$. G) $\equiv$ ($\exists$ $\mathbf{r}_{12}^{-}$. T1\$A) :< G
"onto": H :< ($\exists$ $\mathbf{r}_{12}^{-}$. T1\$B)

Summary

- **Exciting times in Description Logics** (too exciting for my taste ;-)
- **Lots of work on modularization**
- **Return to interest on low-expressivity DLs**
  - ***EL***
  - ***DL Lite***