5.

Structured Descriptions &

Tradeoff Between Expressiveness and Tractability

Outline

- Review
- Expressiveness & Tractability Tradeoff
- Modern Description Logics

Object Oriented Representations

- Key Representation Constructs
  - class, individual, slot and facet
  - subclass-of, instance-of
  - domain, range, cardinality, numeric-minimum, etc
- Key Reasoning Operations
  - Inheritance
  - Default values

Structured Descriptions

- Key Representation Constructs
  - Class, individual, role
  - Concept forming constructors (AND, ALL, EXISTS, FILLS…)
  - Role forming constructors (RESTR, …)
- Key Reasoning Operations
  - Subsumption
  - Classification

Outline

- Review
- Expressiveness & Tractability Tradeoff
  - Properties of reasoning procedures
  - An example description language
  - What makes reasoning hard?
  - Working around reasoning difficulties
- Modern Description Logics

Key Questions in KR&R

- Why restrict the representation language?
- Why not represent anything that needs to be represented using whatever representation language is needed?
- Why not use English as a representation language?

Properties of Reasoning Procedures

- A reasoning procedure is ***sound*** if and only if any sentence that can be derived from a KB using that procedure is logically implied by that procedure
- A reasoning procedure is ***complete*** if and only if any sentence logically implied by a KB can be derived using that procedure
- A reasoning procedure is ***intractable*** if its execution time scales exponentially with the size of the KB

Simple description logic

Consider the language FL defined by:

<concept> ::= atom
| [AND <concept> ... <concept>]
| [ALL <role> <concept>]
| [SOME <role>]   (= [EXISTS 1 <role>])

<role> ::= atom
| [RESTR <role> <concept>]

Example: [ALL :Child [AND Female Student]]

an individual whose children are female students

[ALL [RESTR :Child Female] Student]

an individual whose female children are students

there may or may not be male children and they may or may not be students

Interpretation $\mathfrak{I} = \langle D, I \rangle$ as before, but with

$I[[\text{RESTR } r\ c]] = \{ (x,y) \mid (x,y) \in I[r]$ and $y \in I[c] \}$

So [RESTR :Child Female] is the :Child relation restricted to females = :Daughter

Subsumption defined as usual

Computing subsumption

First for FL^{-} = FL without the `RESTR` operator

- put the concepts into normalized form
- to see if C subsumes D make sure that
  1. for every $p \in C$, $p \in D$
  2. for every [SOME r] $\in C$, [SOME r] $\in D$
  3. for every [ALL s c] $\in C$, find an [ALL s d] $\in D$ such that c subsumes d.

[AND $p_1 \dots p_k$
[SOME r_1] ... [SOME r_m]
[ALL s_1 c_1] ... [ALL s_n c_n]]

Can prove that this method is sound and complete relative to definition based on interpretations

Running time:

- normalization is $O(n^2)$
- structural matching: for each part of C, find a part of D. Again $O(n^2)$

What about all of FL, including `RESTR`?

Subsumption in FL

- cannot settle for part-by-part matching

  [ALL [RESTR :Friend [AND Male Doctor]] [AND Tall Rich]]

  subsumes

  [AND [ALL [RESTR :Friend Male] [AND Tall Bachelor]]
  [ALL [RESTR :Friend Doctor] [AND Rich Surgeon]]]

- complex interactions

  [SOME [RESTR r [AND a b]]]

  subsumes

  [AND [SOME [RESTR r [AND c d]]] [ALL [RESTR r c] [AND a e]]
  [ALL [RESTR r [AND d e]] b]]

In general: FL is powerful enough to encode *all* of propositional logic.

There is a mapping Ω from CNF wffs to FL where

$\models (\alpha \supset \beta)$ iff $\Omega(\alpha)$ is subsumed by $\Omega(\beta)$

But $\models (\alpha \supset (p \wedge \neg p))$ iff α is unsatisfiable

Conclusion: there is no good algorithm for FL unless P=NP

Moral

Even small doses of expressive power can come at a significant computational price

Questions:

- what properties of a representation language control its difficulty?
- how far can expressiveness be pushed without losing good algorithms
- when is easy reasoning adequate for KR purposes?

These questions remain unanswered, but some progress:

- need for case analyses is a major factor
- tradeoff for DL languages is reasonably well understood
- best addressed (perhaps) by looking at working systems

Useful approach:

- find reasoning tasks that are tractable
- analyze difficulty in extending them

Limited languages

Many reasoning problems that can be formulated in terms of FOL entailment (KB $\models^?$ α) admit very specialized methods because of the restricted form of either KB or α

although problem could be solved using full resolution, there is no need

Example 1: Horn clauses

- SLD resolution provides more focussed search
- in propositional case, a linear procedure is available

Example 2: Description logics

Can do DL subsumption using Resolution

Introduce predicate symbols for concepts, and "meaning postulates" like

$$\forall x[P(x) \equiv \forall y(\mathrm{Friend}(x,y) \supset \mathrm{Rich}(y)) \\ \wedge \forall y(\mathrm{Child}(x,y) \supset \\ \forall z(\mathrm{Friend}(y,z) \supset \mathrm{Happy}(z)))]$$

[AND [ALL :Friend Rich]
[ALL :Child
[ALL :Friend Happy]]]

Then ask if MP $\models \forall x[P(x) \supset Q(x)]$

Equations

Example linear equations

Let E be the usual axioms for arithmetic:

$$\forall x \forall y(x+y = y+x),\ \forall x(x+0 = x),\ \ldots$$

Peano axioms

Then we get the following:

$$E \models (x+2y=4 \wedge x-y=1) \supset (x=2 \wedge y=1)$$

Can "solve" linear equations using Resolution!

But there is a much better way:

Gauss-Jordan method with back substitution

- subtract (2) from (1): $3y = 3$
- divide by 3: $y = 1$
- substitute in (1): $x = 2$

In general, a set of linear equations can be solved in $O(n^3)$ operations

This idea obviously generalizes!

always advantageous to use a specialized procedure when it is available, rather than a general method like Resolution

Approach to KR&R System Development

- Given a problem identify a combination of representation and reasoning methods that can solve the problem
- Design a way of combining them into one mechanism

Hybrid reasoning

Want to be able to incorporate a number of special-purpose efficient reasoners into a single scheme such as Resolution

Resolution will be the glue that holds the reasoners together

Simple form: <u>semantic attachment</u>

- attach procedures to functions and predicates

  e.g. numbers: procedures on plus, LessThan, ...

- ground terms and atomic sentences can be *evaluated* prior to Resolution
  - $P(\text{factorial}(4), \text{times}(2,3)) \twoheadrightarrow P(24, 6)$
  - $\text{LessThan}(\text{quotient}(36,6), 5) \vee \alpha \twoheadrightarrow \alpha$
- much better than reasoning directly with axioms

More complex form: <u>theory resolution</u>

- build theory into unification process (the way paramodulation builds in =)
- extended notion of complimentary literals

  $\{\alpha, \text{LessThan}(2,x)\}$ and $\{\text{LessThan}(x,1), \beta\}$ resolve to $\{\alpha,\beta\}$

Outline

- ✓ Review
- ✓ Expressiveness & Tractability Tradeoff
- Modern Description Logics
  - New notation and naming schemes
  - Thorough complexity analysis
  - Tableau reasoners
  - Research on description graphs

Phases of Description Logic Research

- Phase 0 (1965-1980): Pre-DL phase
  - Semantic networks, frames, structured inheritance networks
- Phase 1 (1980-1990): Structural subsumption algorithms
  - Implementation of systems
    - KL-ONE, K-Rep, Krypton, Back, LOOM
- Phase 2 (1990-1995) Tableau based algorithms
  - Focus on propositionally closed DLs
  - Thorough analysis of complexity of reasoning
- Phase 3 (1995-2000) Very expressive DLs
  - Improving Tableau-based methods or conversion to modal logic
- Phase 4 (2000-onwards)
  - Industrial strength system for very expressive DLs with applications to semantic web, bio-medical informatics

From Description Logics by Baader, Horrocks and Sattler, in KR&R Handbook

Modern Description Logics

- Well-specified formal semantics
  - Fragments of First Order Logic (often contained in C2)
  - Closely related to propositional modal logic
- Computational properties are well understood
- Reasoning services
  - Practical decision procedures for key problems: satisfiability, subsumption, query answering
  - Several implemented reasoning systems are available

Modern Notation

- A man that is married to a doctor, and all of whose children are either doctors or professors.
  - B&L notation

    [AND Man

    [EXISTS :married Doctor]

    [ALL :hasChild [OR Doctor Professor]]

  - Current Notation

$\text{Human} \sqcap \neg \text{Female} \sqcap (\exists \text{married.Doctor}) \sqcap (\forall \text{hasChild.}(\text{Doctor} \sqcup \text{Professor}))).$

The Description Logic $\mathcal{ALC}$

- Attributive Concept Language with Complements

  N_C – set of concept names

  N_R – set of role names

  N_O – set of individual objects
- The set of $\mathcal{ALC}$ concepts is the smallest set such that:
  - The following are concepts:
    - $\top$ (top is a concept)
    - $\bot$ (bottom is a concept)
    - Every $A \in N_C$ (all atomic concepts are concepts)
  - If C and D are concepts and $R \in N_R$ then the following are concepts
    - $C \sqcap D$ (the intersection of two concepts is a concept)
    - $C \sqcup D$ (the union of two concepts is a concept)
    - $\neg C$ (the complement of a concept is a concept)
    - $\forall R.C$ (the universal restriction of a concept by a role is a concept)
    - $\exists R.C$ (the existential restriction of a concept by a role is a concept)

The Description Logic $\mathcal{ALC}$

- Terminological Axioms (TBox)
  - A general concept inclusion axiom has the form $\mathbb{C} \sqsubseteq \mathbb{D}$ where C and D are concepts
  - Write $\mathbb{C} \equiv \mathbb{D}$ iff both $\mathbb{C} \sqsubseteq \mathbb{D}$ and $\mathbb{D} \sqsubseteq \mathbb{C}$
  - A TBox is a finite set of GCIs
- Assertional Axioms (ABox)
  - A concept assertion is a statement of the form a:C where a $\in$ N_o C is a concept
  - A role assertion is a statement of the form (a,b):R where a, b $\in$ N_o and R is a role
  - An ABox is a finite set of assertional axioms
- Knowledge Base
  - A KB is an ordered pair ($\mathcal{T}$**,** $\mathcal{A}$**)** for a TBox $\mathcal{T}$ and ABox $\mathcal{A}$

Naming Conventions

- $\mathcal{S}$: basic DL ($\mathcal{ALC}$) plus transitive roles (e.g., ancestor $\in \mathrm{R}_{+}$)
- $\mathcal{N}$: number restrictions (e.g., $\geqslant 2$hasChild, $\leqslant 3$hasChild)
- $\mathcal{Q}$: Qualified number restrictions (e.g., $\geqslant 2$hasChild.Doctor)
- $\mathcal{D}$: concrete domains (e.g., real, integer, string)
- $\mathcal{O}$: Nominals, ie, indvidual names (e.g.,Scientists $\sqcap$ ($\exists$hasMet.{Turing}))
- $\mathcal{I}$: inverse roles (e.g., isChildOf $\equiv$ hasChild^{-})
- $\mathcal{H}$: role hierarchy (e.g., hasDaughter $\sqsubseteq$ hasChild)

$\mathcal{SHOIN}$(D) : A $\mathcal{ALC}$ description logic with role hierarchies, nominals, inverse roles, and number restrictions

Also the logic of the language OWL-DL

Extensive Work on Computational Complexity

http://dl.kr.org

Complexity of reasoning in Description Logics

Note: the information here is (always) incomplete and updated often

Base description logic: $\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements

$\mathcal{ALC} ::= \bot \mid A \mid \neg C \mid C \wedge D \mid C \vee D \mid \exists R.C \mid \forall R.C$

Concept constructors:	Role constructors:
☑ $\mathcal{F}$ – functionality[2]: $(\leq 1\ R)$	☐ $\mathcal{I}$ – role inverse: R^-
☐ $\mathcal{N}$ – (unqualified) number restrictions: $(\geq n\ R)$, $(\leq n\ R)$	☑ $\cap$ – role intersection[3]: $R \cap S$
☐ $\mathcal{Q}$ – qualified number restrictions: $(\geq n\ R.C)$, $(\leq n\ R.C)$	☐ $\cup$ – role union: $R \cup S$
☐ $\mathcal{O}$ – nominals: $\{a\}$ or $\{a_1, \ldots, a_n\}$ ("one-of")	☐ $\neg$ – role complement: $\neg R$ full
☐ μ – least fixpoint operator: $\mu X.C$	☐ $\circ$ – role chain (composition): $R \circ S$
Forbid complex roles[5] in number restrictions[6]	☐ $*$ – reflexive-transitive closure[4]: R^*
	☐ id – concept identity: $id(C)$

trans reg

TBox (concept axioms):	RBox (role axioms):
○ empty TBox	☐ $\mathcal{S}$ – role transitivity: Tr(R)
○ acyclic TBox ($A \equiv C$, A is a concept name; no cycles)	☐ $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$
◉ general TBox ($C \sqsubseteq D$, for arbitrary concepts C and D)	☐ $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$
	☐ s – some additional features (check it to see)

OWL-Lite OWL-DL OWL 1.1

Reset

You have selected a Description Logic: $\mathcal{ALCF}(\cap)$

Complex roles in number restrictions are: forbidden

Complexity of reasoning problems[7]

Reasoning problem	Complexity[8]	Comments and references
Concept satisfiability	**ExpTime-complete**	See [77, Theorem 4.38].
ABox consistency	**ExpTime-complete**	See [77, Theorem 4.42].
Important properties of the description logic		
Finite model property	**Yes**	For the logic $\mathcal{ALCN}(\cap)$ with any TBoxes. Follows from Theorem 3.9 in [14].
Tree model property	☺	

Maintained by: Evgeny Zolin
Please see the list of updates

Any comments are welcome: EZolin@cs.man.ac.uk

Notes:

1. The letters $\mathcal{O}$, $\mathcal{I}$, and $\mathcal{Q}$ are customary written in various orders, e.g., $\mathcal{ALCQIO}$, but $\mathcal{SHOIQ}$. Here we do not reflect this tradition, but rather use a uniform naming scheme.
2. In literature, the letter $\mathcal{F}$ sometimes stands for feature (dis)agreement constructor (see [1, pp.88,488], [53]), rather than functionality (see [7, 54, 40, 46]).
3. The presence of role intersection operator is sometimes indicated by the letter $\mathcal{R}$ in literature, e.g. $\mathcal{ALCNR} := \mathcal{ALCN}(\cap)$.
4. Transitive closure is usually denoted as R^+. The operators $*$ and $^+$ are expressible in terms of each other via equalities: $R^+ = R \circ R^*$ and $R^* = id(\top) \cup R^+$. Note however that the former definition is not linearly bounded. Therefore, any complexity result for a logic with $^+$ immediately implies the same result for a logic with ($*$,$\circ$), but not *vice versa*.
5. In the selector "Allow/disallow complex roles in number restriction", a role (expression) is called *complex* if it contains any role operations other than inversion (i.e. inversion is harmless (with some rare exceptions, which are pointed out in the comments to those cases)). However, in literature it is usually hard or even impossible to determine whether this assumption holds by looking at the *name* of a logic. For instance, $\mathcal{ALCQI}_{reg}$ usually abbreviates a logic where only role names and their inverses are allowed in number restrictions; whereas in the logic $\mathcal{ALCN}(\circ)$, role composition is allowed in number restrictions. To avoid this ambiguity, the selector was introduced here explicitly.

Reasoning Tasks

Is an axiom/fact **entailed** by KB

- KB contains obvious errors

  $\mathcal{K} \models C \equiv \bot$ for some concept name C ?

- KB is consistent with intuitions

  $\mathcal{K} \models C \sqsubseteq D$ s.t. expert believes $C \not\sqsubseteq D$?
  $\mathcal{K} \models C \not\sqsubseteq D$ or $\mathcal{K} \models C \sqsubseteq D$ s.t. expert believes $C \sqsubseteq D$?

- KB entails unexpected equivalences

  $\mathcal{K} \models C \equiv D$ for concept names C and D ?

- KB entails query answers

  $\mathcal{K} \models (\text{Parent} \sqcap \exists \text{hasChild.Doctor}) \sqsubseteq \text{HappyParent}$?
  $\mathcal{K} \models \text{John:HappyParent}$?
  Retrieve all individuals a s.t. $\mathcal{K} \models \text{a:(Wizard} \sqcap \exists \text{ hasPet.Owl)}$

Slide adapted from Ian Horrocks

Reasoning Techniques

- **Direct**
  - Specially designed reasoning algorithms
  - Operate on the DL (more or less) directly
- **Indirect**
  - Translate into some equivalent problem in another formalism
  - Solve resulting problem using appropriate technology

Direct Reasoning Techniques

- Two basic classes of algorithm
  - **Model construction**
    - Prove entailment does not hold by constructing model of KB in which axiom/fact is false
      - tableau algorithms
        - » tableau expansion rules used to derive **new ABox facts**
  - **Proof derivation**
    - Prove entailment holds by deriving axiom/fact from KB
      - structural, completion, rule-based algorithms
        - » deduction rules used to derive **new TBox axioms**

Tableau Algorithms

- Currently the most widely used technique
  - Basis for reasoners such as FaCT++, HermiT, Pellet, Racer, …
  - Standard technique is to negate premise axiom/fact
- Most effective for schema reasoning
  - Large datasets may necessitate construction of large models
  - Query answering may require each possible answer to behecked
  - Optimizations can limit but not eliminate these problems

Tableau Algorithms

- Transform entailment to KB (un)satisfiability
  - $\mathcal{K} \models a{:}C$ iff $\mathcal{K} \cup \{a{:}(\neg C)\}$ is *not* satisfiable
  - $\mathcal{K} \models C \sqsubseteq D$ iff $\mathcal{K} \cup \{a{:}(C \sqcap \neg D)\}$ is *not* satisfiable (for new a)
- Start with facts explicitly asserted in ABox

  e.g., John:HappyParent, John hasChild Mary
- Use expansion rules to derive new **ABox facts**

  e.g., John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
- Construction fails if obvious contradiction (clash)

  e.g., Mary:Doctor, Mary:$\neg$Doctor

Expansion Rules for $\mathcal{ALC}$

$\sqcap$-rule: if 1. $a : (C_1 \sqcap C_2) \in \mathcal{A}$, and
2. $\{a : C_1, a : C_2\} \not\subseteq \mathcal{A}$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1, a : C_2\}$

$\sqcup$-rule: if 1. $a : (C_1 \sqcup C_2) \in \mathcal{A}$, and
2. $\{a : C_1, a : C_2\} \cap \mathcal{A} = \emptyset$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1\}$ and $\mathcal{A}_2 = \mathcal{A} \cup \{a : C_2\}$

$\exists$-rule: if 1. $a : (\exists S.C) \in \mathcal{A}$, and
2. there is no b such that $\{\langle a, b\rangle : S, b : C\} \subseteq \mathcal{A}$,
then set $\mathcal{A}_1 = \mathcal{A} \cup \{\langle a, d\rangle : S, d : C\}$, where d is new in $\mathcal{A}$

$\forall$-rule: if 1. $\{a : (\forall S.C), \langle a, b\rangle : S\} \subseteq \mathcal{A}$, and
2. $b : C \notin \mathcal{A}$
then set $\mathcal{A}_1 = \mathcal{A} \cup \{b : C\}$

- some rules are nondeterministic, e.g., $\sqcup$, $\leq$
- implementations use backtracking search

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \exists\text{hasChild.Doctor})\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models \text{Mary:Doctor}$?

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \exists\text{hasChild.Doctor})\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \exists\text{hasChild.Doctor})\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \exists\text{hasChild.Doctor})\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models \text{Mary:Doctor}$?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.(Doctor} \sqcup \exists\text{hasChild.Doctor)}\}$
$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models$ Mary:Doctor

?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild}.\text{Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild}.(\text{Doctor} \sqcup \exists\text{hasChild}.\text{Doctor})\}$

$\mathcal{A} = \{\text{John}:\text{HappyParent}, \text{John hasChild Mary}$

$\models \text{Mary}:\text{Doctor}$?

John:HappyParent, John hasChild Mary

✘ Mary:¬Doctor

John:Parent, John:∀hasChild.(Doctor ⊔ ∃hasChild.Doctor)

John:Person, John:∃hasChild.Person

Mary:(Doctor ⊔ ∃hasChild.Doctor)

John hasChild a, a:Person, a:(Doctor ⊔ ∃hasChild.Doctor)

✘ Mary:Doctor

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild.Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild.(Doctor} \sqcup \exists \text{hasChild.Doctor)}\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models \text{Mary:Doctor}$?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
Mary:$\exists$hasChild.Doctor

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
Mary:$\exists$hasChild.Doctor
Mary hasChild b, b:Doctor, b:Person

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.}(\text{Doctor} \sqcup \exists\text{hasChild.Doctor})\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}$

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
Mary:$\exists$hasChild.Doctor
Mary hasChild b, b:Doctor, b:Person
a:Doctor

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T} = \{\text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists\text{hasChild.Person},$
$\quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall\text{hasChild.(Doctor} \sqcup \exists\text{hasChild.Doctor)}\}$

$\mathcal{A} = \{\text{John:HappyParent, John hasChild Mary, } \mathbf{Mary{:}\forall hasChild.\bot}$

$\models \text{Mary:Doctor}$?

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}
$\mathcal{A}$ = {John:HappyParent, John hasChild Mary, **Mary:$\forall$hasChild.$\perp$**

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary, Mary:$\forall$hasChild.$\perp$

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}

$\mathcal{A}$ = {John:HappyParent, John hasChild Mary, **Mary:$\forall$hasChild.$\bot$**

$\models$ Mary:Doctor ?

John:HappyParent, John hasChild Mary, Mary:$\forall$hasChild.$\bot$
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
Mary:$\exists$hasChild.Doctor
Mary hasChild b, b:Doctor, b:Person

Slide adapted from Ian Horrocks

Expansion Example

$\mathcal{T}$ = {Doctor $\sqsubseteq$ Person, Parent $\equiv$ Person $\sqcap$ $\exists$hasChild.Person,
HappyParent $\equiv$ Parent $\sqcap$ $\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)}
$\mathcal{A}$ = {John:HappyParent, John hasChild Mary, **Mary:$\forall$hasChild.$\bot$**

$\models$ Mary:Doctor ? ✓

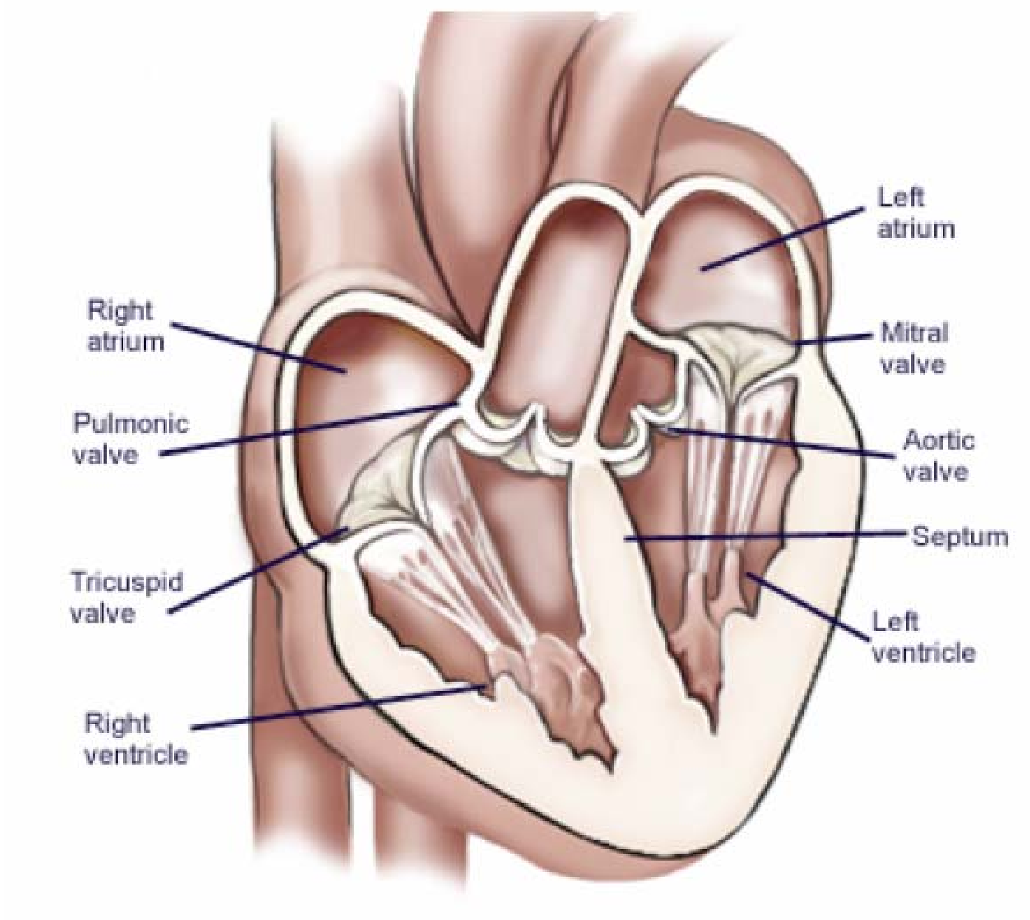
John:HappyParent, John hasChild Mary, Mary:$\forall$hasChild.$\bot$
Mary:$\neg$Doctor
John:Parent, John:$\forall$hasChild.(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John:Person, John:$\exists$hasChild.Person
Mary:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
John hasChild a, a:Person, a:(Doctor $\sqcup$ $\exists$hasChild.Doctor)
Mary:$\exists$hasChild.Doctor
Mary hasChild b, b:Doctor, b:Person
✗ b:$\bot$

Slide adapted from Ian Horrocks

Highly Optimized Implementations

- Blocking (to avoid infinite loops)
- Lazy unfolding
- Simplification and rewriting
- Search optimization
- Caching
- Detecting tractable fragments
- Heuristics
- etc

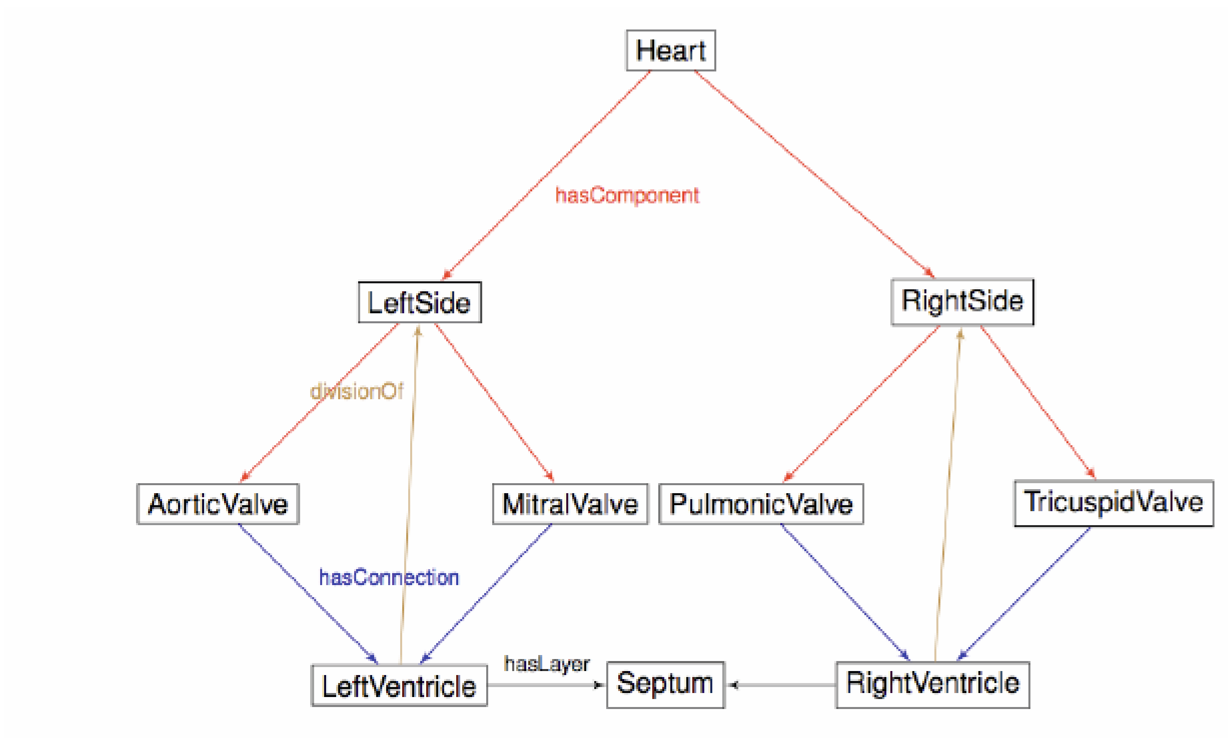
Current Research
Representing Physical Structures



Slide adapted from Ian Horrocks

Current Research

- DLs poor at representing non-tree structures



Slide adapted from Ian Horrocks

Related Conferences

DL 2011 : 24th International Workshop on Description Logics

SHARE

Link: http://dl.kr.org/dl2011

When	Jul 13, 2011 - Jul 16, 2011
Where	Barcelona, Spain
Submission Deadline	May 1, 2011
Notification Due	Jun 5, 2011
Final Version Due	Jun 19, 2011

Categories logic

OWLED 2011
OWL: Experiences and Directions

Eigth International Workshop
San Francisco, California, USA
June 5-6 2011

Co-located with SemTech 2011

Summary

- Review
- Expressiveness & Tractability Tradeoff
  - Properties of reasoning procedures
  - An example description language
  - What makes reasoning hard?
  - Working around reasoning difficulties
- Modern Description Logics
  - New notation and naming schemes
  - Thorough complexity analysis
  - Tableau reasoners
  - Research on description graphs

Reading

- Required
  - Chapter 16 of the B&L Textbook
  - Wikipedia page on Description Logics
    - http://en.wikipedia.org/wiki/Description_logic