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:

\[<\text{concept}> ::= \text{atom} \quad \text{\mid} \quad [\text{AND} \ <\text{concept}> \ ... \ <\text{concept}>] \quad \text{\mid} \quad [\text{RESTR} \ <\text{role}> \ <\text{concept}>] \]

\[<\text{role}> ::= \text{atom} \quad \text{\mid} \quad \text{ALL} \ <\text{role}> \ <\text{concept}> \quad \text{\mid} \quad \text{SOME} \ <\text{role}> \quad (= \exists 1 \ <\text{role}>)]\]

Example:

\[[\text{ALL} \ :\text{Child} \ [\text{AND} \ \text{Female} \ \text{Student}] ]\]

an individual whose children are female students

\[[\text{ALL} \ [\text{RESTR} \ :\text{Child} \ \text{Female}] \ \text{Student}]\]

an individual whose female children are students

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

Interpretation \( \mathcal{I} = \langle D, I \rangle \) as before, but with

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

So \([\text{RESTR} \ :\text{Child} \ \text{Female}]\) 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 $[\text{AND } p_1 \ldots p_k]$
- to see if $C$ subsumes $D$ make sure that $[\text{SOME } r_1] \ldots [\text{SOME } r_m]$
- $[\text{ALL } s_1 c_1] \ldots [\text{ALL } s_n c_n]]$

1. for every $p \in C$, $p \in D$
2. for every $[\text{SOME } r] \in C$, $[\text{SOME } r] \in D$
3. for every $[\text{ALL } s c] \in C$, find an $[\text{ALL } s d] \in D$ such that $c$ subsumes $d$.

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

\[
\text{[ALL [RESTR :Friend [AND Male Doctor]] [AND Tall Rich]]}
\]

subsumes

\[
\text{[AND [ALL [RESTR :Friend Male] [AND Tall Bachelor]]}
\]

\[
\text{[ALL [RESTR :Friend Doctor] [AND Rich Surgeon]]]}
\]

- complex interactions

\[
\text{[SOME [RESTR r [AND a b]]]}
\]

subsumes

\[
\text{[AND [SOME [RESTR r [AND c d]]] [ALL [RESTR r c] [AND a e]]}
\]

\[
\text{[ALL [RESTR r [AND d e]] b]]}
\]

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

There is a mapping \( \Omega \) from CNF wffs to FL where

\(|= (\alpha \supset \beta) \iff \Omega(\alpha) \text{ is subsumed by } \Omega(\beta)|

But \(|= (\alpha \supset (p \land \neg p)) \iff \alpha \text{ 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 ⊨ 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

∀x[P(x) ⇔ ∀y(Friend(x,y) ⊃ Rich(y))]
∧ ∀y(Child(x,y) ⊃
    ∀z(Friend(y,z) ⊃ Happy(z)))]

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

Then ask if MP |= ∀x[P(x) ⊃ Q(x)]
Equations

Example  linear equations

Let $E$ be the usual axioms for arithmetic:
\[ \forall x \forall y (x + y = y + x), \quad \forall x (x + 0 = x), \quad \text{Peano axioms} \]

Then we get the following:
\[ E \models (x + 2y = 4 \land x - y = 1) \supset (x = 2 \land y = 1) \]

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: semantic attachment

- attach procedures to functions and predicates
  
  e.g. numbers: procedures on plus, LessThan, ...

- ground terms and atomic sentences can be evaluated prior to Resolution

  \[ P(\text{factorial}(4), \text{times}(2,3)) \Rightarrow P(24, 6) \]

  \[ \text{LessThan}(\text{quotient}(36,6), 5) \lor \alpha \Rightarrow \alpha \]

- much better than reasoning directly with axioms

More complex form: theory resolution

- build theory into unification process (the way paramodulation builds in =)

- extended notion of complimentary literals

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

Adapted from Ian Horrocks
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
  Human ⊨ ¬ Female ⊨ (∃married.Doctor) ⊨ (∀hasChild.(Doctor ⊨ 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
    - $C \cap D$ (the intersection of two concepts is a concept)
    - $C \cup D$ (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 $C \sqsubseteq D$ where $C$ and $D$ are concepts
  - Write $C \equiv D$ iff both $C \sqsubseteq D$ and $D \sqsubseteq C$
  - A TBox is a finite set of GCIls

- **Assertional Axioms (ABox)**
  - A concept assertion is a statement of the form $a:C$ where $a \in N_0$
  - C is a concept
  - A role assertion is a statement of the form $(a,b):R$ where $a, b \in N_0$ 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

\[ S \]: basic DL (\( \mathcal{ALC} \)) plus transitive roles (e.g., ancestor \( \in R^+ \))

\[ N \]: number restrictions (e.g., \( \geq 2 \text{hasChild}, \leq 3 \text{hasChild} \))

\[ Q \]: Qualified number restrictions (e.g., \( \geq 2 \text{hasChild.Doctor} \))

\[ D \]: concrete domains (e.g., real, integer, string)

\[ O \]: Nominals, ie, individual names (e.g., Scientists \( \sqcap (\exists \text{hasMet.}\{\text{Turing}\}) \))

\[ I \]: inverse roles (e.g., isChildOf \( \equiv \text{hasChild}^\sim \))

\[ H \]: role hierarchy (e.g., hasDaughter \( \sqsubseteq \) hasChild)

\[ SHOIN(D) \]: 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

Base description logic: Atriibutive Language with Complements

\[ \text{ALC} ::= \top \mid A \mid \neg C \mid C \land D \mid C \lor D \mid \exists R.C \mid \forall R.C \]

Concept constructors:
- \( \exists R \) - functionality\(^2\): \((\leq 1) R\)
- \( \exists \neg \) - (unqualified) number restrictions: \((2n) R\), \((3n) R\)
- \( \exists Q \) - qualified number restrictions: \((2n R.C)\), \((3n R.C)\)
- \( \exists \) - nominals: \(\{a\}\) or \(\{a_1, \ldots, a_n\}\) ("one-of")
- \( \exists \mu \) - least fixpoint operator: \(\mu X.C\)

Role constructors:
- \( I \) - role inverse: \(R^\neg\)
- \( \neg R \) - role intersection: \(R \land S\)
- \( \forall R\) - role union: \(R \lor S\)
- \( \
eg \) - role complement: \(\neg R\)
- \( \land \) - role chain (composition): \(R \land S\)
- \( \mu \) - reflexive-transitive closure: \(R^*\)
- \( \mu \) - concept identity: \(id(C)\)

TBox (concept axioms):
- empty TBox
- acyclic TBox (A \(\subseteq\) C, A is a concept name; no cycles)
- general TBox (C \(\subseteq\) D, for arbitrary concepts C and D)

RBox (role axioms):
- \( \exists R \) - role transitivity: \(Tr(R)\)
- \( \forall R \) - role hierarchy: \(R \subseteq S\)
- \( \forall R \) - complex role inclusions: \(R \subseteq R, \forall R \subseteq S\)
- \( s \) - some additional features (check it to see)

You have selected a Description Logic: \(\text{ALCF}(\top)\)
Complex roles in number restrictions are: forbidden

Complexity of reasoning problems\(^2\)

<table>
<thead>
<tr>
<th>Reasoning problem</th>
<th>Complexity(^2)</th>
<th>Comments and references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept satisfiability</td>
<td>ExpTime-complete</td>
<td>See [77, Theorem 4.38].</td>
</tr>
<tr>
<td>ABox consistency</td>
<td>ExpTime-complete</td>
<td>See [77, Theorem 4.42].</td>
</tr>
</tbody>
</table>

Important properties of the description logic

- Finite model property: Yes
  For the logic \(\text{ALCM}(\top)\) with any TBoxes. Follows from Theorem 3 in [14].
- Tree model property: No

Notes:
1. The letters \( I, J, \) and \( Q \) are customary written in various orders, e.g., \(\text{ALCQI}\), but \(\text{SNARK}\) here we do not reflect this tradition, but rather use a uniform naming scheme.
2. In literature, the letter \( F \) sometimes stands for feature (dis)agreement constructor (see [1, pp.88,489], [3]), rather than functionality (see [1, 55, 45, 46]).
3. The presence of role intersection operator is sometimes indicated by the letter \( R \) in literature, e.g., \(\text{ALCQI}^R\) = \(\text{ALCN}(\top)\).
4. Transitive closure is usually denoted as \(R^*\). The operators \( * \) and \( \ast \) are expressible in terms of each other via equalities: \(R^\ast = R \circ R^\ast\) and \(R^\ast = id(T) \cup R^\ast\). 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 \( \ast \), 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, \(\text{ALCQI}^\top\) usually abbreviates a logic where only role names and their inverses are allowed in number restrictions; whereas in the logic \(\text{ALCN}(\top)\), role composition is allowed in number restrictions. To avoid this ambiguity, the selector was introduced here explicitly.

Maintained by: Evgeny Zolin
Please see the list of updates

Any comments are welcome: E.Zolin@cs.man.ac.uk
Reasoning Tasks

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

- KB contains **obvious errors**
  \[ \mathcal{K} \models C \equiv \bot \text{ for some concept name } C \quad \? \]

- KB is **consistent with intuitions**
  \[ \mathcal{K} \models C \subseteq D \text{ s.t. expert believes } C \not\subseteq D \quad ? \]
  \[ \mathcal{K} \models C \not\subseteq D \text{ or } \mathcal{K} \models C \subseteq D \text{ s.t. expert believes } C \subseteq D \quad ? \]

- KB **entails unexpected equivalences**
  \[ \mathcal{K} \models C \equiv D \text{ for concept names } C \text{ and } D \quad ? \]

- KB **entails query answers**
  \[ \mathcal{K} \models (\text{Parent} \sqcap \exists \text{hasChild.Doctor}) \subseteq \text{HappyParent} \quad ? \]
  \[ \mathcal{K} \models \text{John:HappyParent} \quad ? \]
  \[ \text{Retrieve all individuals } a \text{ s.t. } \mathcal{K} \models a:(\text{Wizard} \sqcap \exists \text{hasPet.Owl}) \quad \]
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

Slide adapted from Ian Horrocks
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 be checked
  - Optimizations can limit but not eliminate these problems

Slide adapted from Ian Horrocks
Tableau Algorithms

- Transform entailment to KB (un)satisfiability
  - $\mathcal{K} \models a:C$ iff $\mathcal{K} \cup \{a: (\neg C)\}$ is not satisfiable
  - $\mathcal{K} \models C \subseteq D$ iff $\mathcal{K} \cup \{a: (C \cap \neg D)\}$ 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:∀hasChild.(Doctor ∨ ∃hasChild.Doctor)

- Construction fails if obvious contradiction (clash)
  - e.g., Mary:Doctor, Mary:¬Doctor

Slide adapted from Ian Horrocks
Expansion Rules for $\mathcal{ALC}$

- □-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 $\mathcal{A}_1 = \mathcal{A} \cup \{a : C_1, a : C_2\}$

- ⊖-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 $\mathcal{A}_2 = \mathcal{A} \cup \{a : C_2\}$

- ∃-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 $\mathcal{A}$

- ∀-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., ⊖, ≤
- implementations use backtracking search

Slide adapted from Ian Horrocks
Expansion Example

\[ T = \{ \text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}, \]
\[ \quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \} \]
\[ A = \{ \text{John:HappyParent, John hasChild Mary} \}
\]
\[ \models \text{Mary:Doctor} \quad ? \]
Expansion Example

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

\[ \models \text{Mary:Doctor} \quad ? \]

\[ \text{John:HappyParent, John hasChild Mary} \]
Expansion Example

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

\[ \vdash \text{Mary:Doctor} \quad ? \]

\begin{align*}
\text{John:HappyParent, John hasChild Mary} \\
\text{Mary:}\neg\text{Doctor}
\end{align*}

Slide adapted from Ian Horrocks
Expansion Example

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

\[ \vdash \text{Mary:Doctor} \]

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

Slide adapted from Ian Horrocks
Expansion Example

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

\[ \models \text{Mary}:\text{Doctor} \]

\[ \text{John}:\text{HappyParent}, \text{John hasChild Mary} \]
\[ \text{Mary}:\neg \text{Doctor} \]
\[ \text{John}:\text{Parent}, \text{John} \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \]
\[ \text{John}:\text{Person}, \text{John} \exists \text{hasChild}.\text{Person} \]
Expansion Example

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

\[ \equiv \text{Mary:Doctor} \quad ? \]

\begin{align*}
\text{John:HappyParent, John hasChild Mary} \\
\text{Mary:}\neg\text{Doctor} \\
\text{John:Parent, John:}\forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \\
\text{John:Person, John:}\exists \text{hasChild}.\text{Person} \\
\text{Mary:}(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor})
\end{align*}

Slide adapted from Ian Horrocks
Expansion Example

\[ \mathcal{T} = \{\text{Doctor} \subseteq \text{Person}, \; \text{Parent} \equiv \text{Person} \land \exists \text{hasChild}.\text{Person}, \]
\[ \quad \text{HappyParent} \equiv \text{Parent} \land \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor})\} \]
\[ \mathcal{A} = \{\text{John:HappyParent, John hasChild Mary}\} \]

\[ \vdash \text{Mary:Doctor} \]

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

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

\[ \vdash \text{Mary:Doctor} \]

\[ \text{John:HappyParent, John hasChild Mary} \]

\[ \text{Mary:}\neg\text{Doctor} \]

\[ \text{John:Parent, John:}\forall\text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \]

\[ \text{John:Person, John:}\exists\text{hasChild}.\text{Person} \]

\[ \text{Mary:(Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \]

\[ \text{John hasChild a, a:Person, a:(Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \]

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

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

$\vdash \text{Mary}:\text{Doctor}$

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)

Slide adapted from Ian Horrocks
Expansion Example

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

\[ \models \text{Mary:Doctor} \ ? \]

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

Slide adapted from Ian Horrocks
Expansion Example

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

\[ \models \text{Mary:Doctor} \]

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

Slide adapted from Ian Horrocks
Expansion Example

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

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

$\not\models \text{Mary:Doctor}$

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:∃hasChild.Doctor
Mary hasChild b, b:Doctor, b:Person
a:Doctor

Slide adapted from Ian Horrocks
Expansion Example

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

\[ \vdash \text{Mary:Doctor} \]
Expansion Example

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

\[ \models \text{Mary:Doctor} \]

\[ \text{John:HappyParent, John hasChild Mary, Mary:}\forall\text{hasChild.}\perp \]

Slide adapted from Ian Horrocks
Expansion Example

\[ T = \{ \text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}, \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \} \]
\[ A = \{ \text{John:HappyParent}, \text{John hasChild Mary}, \text{Mary:}\forall \text{hasChild}.\perp \]
Expansion Example

\[ T = \{ \text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}, \]
\[ \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \} \]
\[ \mathcal{A} = \{ \text{John}: \text{HappyParent}, \text{John hasChild Mary}, \text{Mary}: \forall \text{hasChild}.\bot \]
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
Current Research

• DLs poor at representing non-tree structures

Slide adapted from Ian Horrocks
Related Conferences

DL 2011: 24th International Workshop on Description Logics

<table>
<thead>
<tr>
<th>When</th>
<th>Jul 13, 2011 - Jul 16, 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where</td>
<td>Barcelona, Spain</td>
</tr>
<tr>
<td>Submission Deadline</td>
<td>May 1, 2011</td>
</tr>
<tr>
<td>Notification Due</td>
<td>Jun 6, 2011</td>
</tr>
<tr>
<td>Final Version Due</td>
<td>Jun 19, 2011</td>
</tr>
</tbody>
</table>

Categories: logic

OWLED 2011
OWL: Experiences and Directions

Eighth 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