Description Logic: A Formal Foundation for Ontology Languages and Tools

Part 1: Languages

Ian Horrocks
<ian.horrocks@comlab.ox.ac.uk>
Information Systems Group
Oxford University Computing Laboratory
Contents

• Motivation
• Brief review of (first order) logic
• Description Logics as fragments of FOL
• Description Logic syntax and semantics
• Brief review of relevant complexity notions
• Description Logics and OWL
• Ontology applications
• Ontologies –v- databases
DL Basics
What Are Description Logics?
What Are Description Logics?

- Decidable fragments of First Order Logic

Thank you for listening

Any questions?
What Are Description Logics?

• A family of logic based Knowledge Representation formalisms
  – Originally descended from semantic networks and KL-ONE
  – Describe domain in terms of concepts (aka classes), roles (aka properties, relationships) and individuals

[Quillian, 1967]
What Are Description Logics?

• Modern DLs (after Baader et al) distinguished by:
  – Fully fledged logics with formal semantics
    • Decidable fragments of FOL (often contained in $C_2$)
    • Closely related to Propositional Modal/Dynamic Logics & Guarded Fragment
  – Computational properties well understood (worst case complexity)
  – Provision of inference services
    • Practical decision procedures (algorithms) for key problems
      (satisfiability, subsumption, query answering, etc)
    • Implemented systems (highly optimised)

• The basis for widely used ontology languages
Web Ontology Language OWL (2)

• **W3C** recommendation(s)

• Motivated by **Semantic Web** activity
  
  Add meaning to web content by annotating it with terms defined in ontologies

• Supported by **tools and infrastructure**
  
  – APIs (e.g., OWL API, Thea, OWLink)
  – Development environments
    (e.g., Protégé, Swoop, TopBraid Composer, Neon)
  – Reasoners & Information Systems
    (e.g., Pellet, Racer, HermiT, Quonto, …)

• Based on **Description Logics** (**SHOIN / SROIQ**)
and now:

A Word from our Sponsors
Crash Course in (simplified) FOL

• Syntax
  – Non-logical symbols (signature)
    • Constants: Felix, MyMat
    • Predicates(arity): Animal(1), Cat(1), has-color(2), sits-on(2)
  – Logical symbols:
    • Variables: x, y
    • Operators: ∧, ∨, →, ¬, …
    • Quantifiers: ∃, ∀
    • Equality: =
  – Formulas:
    • Cat(\text{Felix}), \text{Mat(\text{MyMat})}, \text{sits-on(\text{Felix, MyMat})}
    • Cat(x), Cat(x) ∨ \text{Human}(x), ∃y.\text{Mat}(y) ∧ \text{sits-on}(x, y)
    • ∀x.\text{Cat}(x) → \text{Animal}(x), ∀x.\text{Cat}(x) → (∃y.\text{Mat}(y) ∧ \text{sits-on}(x, y))
Crash Course in (simplified) FOL

- Semantics
Crash Course in (simplified) FOL

- Semantics
Crash Course in (simplified) FOL

• Semantics
Crash Course in (simplified) FOL

• Semantics

Why should I care about semantics? -- In fact I heard that a little goes a long way!
Crash Course in (simplified) FOL

• Semantics

Why should I care about semantics? -- In fact I heard that a little goes a long way!
Crash Course in (simplified) FOL

• Semantics

Why should I care about semantics? -- In fact I heard that a little goes a long way!

Well, from a philosophical POV, we need to specify the relationship between statements in the logic and the existential phenomena they describe.
Crash Course in (simplified) FOL

- Semantics

Why should I care about semantics? -- In fact I heard that a little goes a long way!

Well, from a philosophical POV, we need to specify the relationship between statements in the logic and the existential phenomena they describe.

That's OK, but I don't get paid for philosophy.
Crash Course in (simplified) FOL

• Semantics

Why should I care about semantics? -- In fact I heard that a little goes a long way!

Well, from a philosophical POV, we need to specify the relationship between statements in the logic and the existential phenomena they describe.

That’s OK, but I don’t get paid for philosophy.

From a practical POV, in order to specify, build and test (ontology-based) tools/systems we need to precisely define relationships (like entailment) between logical statements – this defines the intended behaviour of tools/systems.
Crash Course in (simplified) FOL

• Semantics

In FOL we define the semantics in terms of models (a model theory). A model is supposed to be an analogue of (part of) the world being modeled. FOL uses a very simple kind of model, in which “objects” in the world (not necessarily physical objects) are modeled as elements of a set, and relationships between objects are modeled as sets of tuples.
Crash Course in (simplified) FOL

• Semantics

In FOL we define the semantics in terms of models (a model theory). A model is supposed to be an analogue of (part of) the world being modeled. FOL uses a very simple kind of model, in which “objects” in the world (not necessarily physical objects) are modeled as elements of a set, and relationships between objects are modeled as sets of tuples.

Note that this is exactly the same kind of model as used in a database: objects in the world are modeled as values (elements) and relationships as tables (sets of tuples).
Crash Course in (simplified) FOL

- Semantics
  - Model: a pair \( \langle D, \cdot^I \rangle \) with \( D \) a non-empty set and \( \cdot^I \) an interpretation
    - \( C^I \) is an element of \( D \) for \( C \) a constant
    - \( v^I \) is an element of \( D \) for \( v \) a variable
    - \( P^I \) is a subset of \( D^n \) for \( P \) a predicate of arity \( n \)
  - E.g., \( D = \{a, b, c, d, e, f\} \), and
    - \( \text{Felix}^I = a \)
    - \( \text{MyMat}^I = b \)
    - \( \text{Cat}^I = \{a, c\} \)
    - \( \text{Mat}^I = \{b, e\} \)
    - \( \text{Animal}^I = \{a, c, d\} \)
    - \( \text{sits-on}^I = \{\langle a, b \rangle, \langle c, e \rangle\} \)
Crash Course in (simplified) FOL

• Semantics
  – Evaluation: truth value in a given model \( M = \langle D, \cdot^I \rangle \)
    - \( P(t_1, \ldots, t_n) \) is \textit{true} iff \( \langle t_1^I, \ldots, t_n^I \rangle \in P^I \)
    - \( A \land B \) is \textit{true} iff \( A \) is \textit{true} and \( B \) is \textit{true}
      \( \neg A \) is \textit{true} iff \( A \) is not \textit{true}
  – E.g.,

\[
\begin{align*}
\text{Cat}(Felix) & \quad \text{true} \\
\text{Cat}(MyMat) & \quad \text{false} \\
\neg \text{Mat}(Felix) & \quad \text{true} \\
\text{sits-on}(Felix, MyMat) & \quad \text{true} \\
\text{Mat}(Felix) \lor \text{Cat}(Felix) & \quad \text{true}
\end{align*}
\]

\[
\begin{align*}
D &= \{a, b, c, d, e, f\} \\
Felix^I &= a \\
MyMat^I &= b \\
\text{Cat}^I &= \{a, c\} \\
\text{Mat}^I &= \{b, e\} \\
\text{Animal}^I &= \{a, c, d\} \\
\text{sits-on}^I &= \{\langle a, b \rangle, \langle c, e \rangle\}
\end{align*}
\]
Crash Course in (simplified) FOL

• Semantics
  – Evaluation: truth value in a given model $M = \langle D, \cdot^I \rangle$
    • $\exists x.A$ is true iff exists $\cdot^I'$ s.t. $\cdot^I$ and $\cdot^I'$ differ only w.r.t. $x$, and $A$ is true w.r.t. $\langle D, \cdot^I' \rangle$
    • $\forall x.A$ is true iff for all $\cdot^I'$ s.t. $\cdot^I$ and $\cdot^I'$ differ only w.r.t. $x$, $A$ is true w.r.t. $\langle D, \cdot^I' \rangle$

E.g.,
\[
\begin{align*}
\exists x.\text{Cat}(x) & \quad \text{true} \\
\forall x.\text{Cat}(x) & \quad \text{false} \\
\exists x.\text{Cat}(x) \land \text{Mat}(x) & \quad \text{false} \\
\forall x.\text{Cat}(x) \rightarrow \text{Animal}(x) & \quad \text{true} \\
\forall x.\text{Cat}(x) \rightarrow (\exists y.\text{Mat}(y) \land \text{sits-on}(x, y)) & \quad \text{true}
\end{align*}
\]

\[
\begin{array}{ll}
D = \{a, b, c, d, e, f\} \\
\text{Felix}^I = a \\
\text{MyMat}^I = b \\
\text{Cat}^I = \{a, c\} \\
\text{Mat}^I = \{b, e\} \\
\text{Animal}^I = \{a, c, d\} \\
\text{sits-on}^I = \{\langle a, b\rangle, \langle c, e\rangle\}
\end{array}
\]
Crash Course in (simplified) FOL

- **Semantics**
  - Given a model $M$ and a formula $F$, $M$ is a model of $F$ (written $M \models F$) iff $F$ evaluates to true in $M$.
  - A formula $F$ is **satisfiable** iff there exists a model $M$ s.t. $M \models F$.
  - A formula $F$ **entails** another formula $G$ (written $F \models G$) iff every model of $F$ is also a model of $G$ (i.e., $M \models F$ implies $M \models G$).

E.g.,

\[
\begin{align*}
M \models & \exists x. \text{Cat}(x) \\
M \not\models & \forall x. \text{Cat}(x) \\
M \not\models & \exists x. \text{Cat}(x) \land \text{Mat}(x) \\
M \models & \forall x. \text{Cat}(x) \rightarrow \text{Animal}(x) \\
M \models & \forall x. \text{Cat}(x) \rightarrow (\exists y. \text{Mat}(y) \land \text{sits-on}(x, y))
\end{align*}
\]

\[
D = \{a, b, c, d, e, f\} \\
\text{Felix}^I = a \\
\text{MyMat}^I = b \\
\text{Cat}^I = \{a, c\} \\
\text{Mat}^I = \{b, e\} \\
\text{Animal}^I = \{a, c, d\} \\
\text{sits-on}^I = \{\langle a, b \rangle, \langle c, e \rangle\}
\]
Crash Course in (simplified) FOL

• Semantics
  
  – Given a model $M$ and a formula $F$, $M$ is a model of $F$ (written $M \models F$) iff $F$ evaluates to true in $M$.
  
  – A formula $F$ is **satisfiable** iff there exists a model $M$ s.t. $M \models F$.
  
  – A formula $F$ **entails** another formula $G$ (written $F \models G$) iff every model of $F$ is also a model of $G$ (i.e., $M \models F$ implies $M \models G$).

E.g.,

<table>
<thead>
<tr>
<th>Formula</th>
<th>Truth Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall x. \text{Cat}(x)$</td>
<td>$\models$</td>
</tr>
<tr>
<td>$\exists x. \text{Cat}(x)$</td>
<td>$\models$</td>
</tr>
<tr>
<td>$(\forall x. \text{Cat}(x) \rightarrow \text{Animal}(x)) \land \text{Cat}(\text{Felix})$</td>
<td>$\models$</td>
</tr>
<tr>
<td>$(\forall x. \text{Cat}(x) \rightarrow \text{Animal}(x)) \land \neg \text{Animal}(\text{Felix})$</td>
<td>$\not\models$</td>
</tr>
<tr>
<td>$\text{Cat}(\text{Felix})$</td>
<td>$\not\models$</td>
</tr>
<tr>
<td>$\neg \text{Cat}(\text{Felix})$</td>
<td>$\not\models$</td>
</tr>
<tr>
<td>$\text{sits-on}(\text{Felix}, \text{Mat1}) \land \text{sits-on}(\text{Tiddles}, \text{Mat2})$</td>
<td>$\not\models$</td>
</tr>
<tr>
<td>$\neg \text{sits-on}(\text{Felix}, \text{Mat2})$</td>
<td>$\not\models$</td>
</tr>
<tr>
<td>$\text{sits-on}(\text{Felix}, \text{Mat1}) \land \text{sits-on}(\text{Tiddles}, \text{Mat1})$</td>
<td>$\models$</td>
</tr>
<tr>
<td>$\exists^{\geq 2} x. \text{sits-on}(x, \text{Mat1})$</td>
<td>$\not\models$</td>
</tr>
</tbody>
</table>
Decidable Fragments

• FOL (satisfiability) well known to be undecidable
  – A sound, complete and terminating algorithm is impossible

• Interesting decidable fragments include, e.g.,
  – C2: FOL with 2 variables and Counting quantifiers \((\exists^{\geq n}, \exists^{\leq n})\)
    • Counting quantifiers abbreviate pairwise (in-) equalities, e.g.:
      \[ \exists^{\geq 3} x. \text{Cat}(x) \] equivalent to
      \[ \exists x, y, z. \text{Cat}(x) \land \text{Cat}(y) \land \text{Cat}(z) \land x \neq y \land x \neq z \land y \neq z \]
      \[ \exists^{\leq 2} x. \text{Cat}(x) \] equivalent to
      \[ \forall x, y, z. \text{Cat}(x) \land \text{Cat}(y) \land \text{Cat}(z) \rightarrow x = y \lor x = z \lor y = z \]
  – Propositional modal and description logics
  – Guarded fragment
Back to our Scheduled Program
DL Syntax

• **Signature**
  - **Concept** (aka class) names, e.g., Cat, Animal, Doctor
    • Equivalent to FOL unary predicates
  - **Role** (aka property) names, e.g., sits-on, hasParent, loves
    • Equivalent to FOL binary predicates
  - **Individual** names, e.g., Felix, John, Mary, Boston, Italy
    • Equivalent to FOL constants
DL Syntax

• Operators
  – Many kinds available, e.g.,
    • Standard FOL Boolean operators (\(\cap, \cup, \neg\))
    • Restricted form of quantifiers (\(\exists, \forall\))
    • Counting (\(\geq, \leq, =\))
    • …
DL Syntax

• Concept expressions, e.g.,
  – Doctor ⊔ Lawyer
  – Rich ⊓ Happy
  – Cat ⊓ ∃sits-on.Mat

• Equivalent to FOL formulae with one free variable
  – Doctor(x) ∨ Lawyer(x)
  – Rich(x) ∧ Happy(x)
  – ∃y.(Cat(x) ∧ sits-on(x, y))
DL Syntax

• Special concepts
  – $\top$ (aka top, Thing, most general concept)
  – $\bot$ (aka bottom, Nothing, inconsistent concept)

used as abbreviations for
  – $(A \sqcup \neg A)$ for any concept $A$
  – $(A \sqcap \neg A)$ for any concept $A$
DL Syntax

• Role expressions, e.g.,
  – loves
  – hasParent o hasBrother

• Equivalent to FOL formulae with two free variables
  – loves(y, x)
  – ∃z.(hasParent(x, z) ∧ hasBrother(z, y))
DL Syntax

• “Schema” Axioms, e.g.,
  - `Rich ⊑ ¬Poor`  (concept inclusion)
  - `Cat ⊓ ∃sits-on.Mat ⊑ Happy`  (concept inclusion)
  - `BlackCat ≡ Cat ⊓ ∃hasColour.Black`  (concept equivalence)
  - `sits-on ⊑ touches`  (role inclusion)
  - `Trans(part-of)`  (transitivity)

• Equivalent to (particular form of) FOL sentence, e.g.,
  - `∀x.(Rich(x) → ¬Poor(x))`
  - `∀x.(Cat(x) ∧ ∃y.(sits-on(x,y) ∧ Mat(y)) → Happy(x))`
  - `∀x.(BlackCat(x) ↔ (Cat(x) ∧ ∃y.(hasColour(x,y) ∧ Black(y))))`
  - `∀x,y.(sits-on(x,y) → touches(x,y))`
  - `∀x,y,z.((sits-on(x,y) ∧ sits-on(y,z)) → sits-on(x,z))`
DL Syntax

• “Data” **Axioms** (aka Assertions or Facts), e.g.,
  – BlackCat(Felix) (concept assertion)
  – Mat(Mat1) (concept assertion)
  – Sits-on(Felix,Mat1) (role assertion)

• Directly equivalent to FOL “ground facts”
  – Formulae with no variables
DL Syntax

• A set of axioms is called a **TBox**, e.g.:

\[
\{\text{Doctor} \sqsubseteq \text{Person}, \\
\text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}, \\
\text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}
\}
\]

• A set of facts is called an **ABox**, e.g.:

\[
\{\text{HappyParent}(\text{John}), \\
\text{hasChild}(\text{John}, \text{Mary})\}
\]

• A **Knowledge Base** (KB) is just a TBox plus an Abox
  – Often written \( \mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle \)

**Note**
Facts sometimes written
- John:HappyParent,
- John hasChild Mary,
- \langle John, Mary \rangle: hasChild
The DL Family

• Many different DLs, often with “strange” names
  – E.g., $EL$, $ALC$, $SHIQ$

• Particular DL defined by:
  – Concept operators ($\cap$, $\cup$, $\neg$, $\exists$, $\forall$, etc.)
  – Role operators ($\cdot$, $\circ$, etc.)
  – Concept axioms ($\sqsubseteq$, $\equiv$, etc.)
  – Role axioms ($\sqsubseteq$, $\text{Trans}$, etc.)
The DL Family

• E.g., $\mathcal{EL}$ is a well known “sub-Boolean” DL
  – Concept operators: $\land$, $\neg$, $\exists$
  – No role operators (only atomic roles)
  – Concept axioms: $\sqsubseteq$, $\equiv$
  – No role axioms

• E.g.:

  \begin{align*}
  \text{Parent} & \equiv \text{Person} \land \exists \text{hasChild}.\text{Person}
  \end{align*}
The DL Family

- \textbf{\textit{ALC}} is the smallest propositionally closed DL
  - Concept operators: $\cap$, $\cup$, $\neg$, $\exists$, $\forall$
  - No role operators (only atomic roles)
  - Concept axioms: $\subseteq$, $\equiv$
  - No role axioms

- E.g.:

$$\text{ProudParent} \equiv \text{Person} \sqcap \forall \text{hasChild.}(\text{Doctor} \sqcup \exists \text{hasChild.}\text{Doctor})$$
The DL Family

- *S* used for *ALC* extended with (role) transitivity axioms
- **Additional letters** indicate various extensions, e.g.:  
  - *H* for role hierarchy (e.g., hasDaughter ⊆ hasChild)  
  - *R* for role box (e.g., hasParent ◦ hasBrother ⊆ hasUncle)  
  - *O* for nominals/singleton classes (e.g., {Italy})  
  - *I* for inverse roles (e.g., isChildOf ≡ hasChild⁻)  
  - *N* for number restrictions (e.g., ≥2hasChild, ≤3hasChild)  
  - *Q* for qualified number restrictions (e.g., ≥2hasChild.Doctor)  
  - *F* for functional number restrictions (e.g., ≤1hasMother)
- E.g., *SHIQ* = *S* + role hierarchy + inverse roles + QNRs
The DL Family

• Numerous other extensions have been investigated
  – Concrete domains (numbers, strings, etc)
  – DL-safe rules (Datalog-like rules)
  – Fixpoints
  – Role value maps
  – Additional role constructors ($\cap$, $\cup$, $\neg$, $\circ$, $\text{id}$, …)
  – Nary (i.e., predicates with arity >2)
  – Temporal
  – Fuzzy
  – Probabilistic
  – Non-monotonic
  – Higher-order
  – …
DL Semantics

Via translation to FOL, or directly using FO model theory:

- **Interpretation function** $\mathcal{I}$
- **Interpretation domain** $\Delta^\mathcal{I}$

**Individuals** $i^\mathcal{I} \in \Delta^\mathcal{I}$
- John
- Mary

**Concepts** $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$
- Lawyer
- Doctor
- Vehicle

**Roles** $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$
- hasChild
- owns
DL Semantics

- Interpretation function extends to **concept expressions** in the obvious(ish) way, e.g.:

\[
(C \equiv D) = C \cap D
\]
\[
(C \equiv D) = C \cup D
\]
\[
(\neg C) = \Delta \setminus C
\]
\[
\{x\} = \{x\}
\]
\[
(\exists R.C) = \{x \mid \exists y.\langle x, y \rangle \in R \land y \in C\}
\]
\[
(\forall R.C) = \{x \mid \forall y.\langle x, y \rangle \in R \Rightarrow y \in C\}
\]
\[
(\leq n R) = \{x \mid \#\{y \mid \langle x, y \rangle \in R\} \leq n\}
\]
\[
(\geq n R) = \{x \mid \#\{y \mid \langle x, y \rangle \in R\} \geq n\}
\]
DL Semantics

- Given a model $M = \langle D, I \rangle$
  - $M \models C \subseteq D$ iff $C^I \subseteq D^I$
  - $M \models C \equiv D$ iff $C^I = D^I$
  - $M \models C(a)$ iff $a^I \in C^I$
  - $M \models R(a, b)$ iff $\langle a^I, b^I \rangle \in R^I$
  - $M \models \langle T, A \rangle$ iff for every axiom $ax \in T \cup A$, $M \models ax$
DL Semantics

• Satisfiability and entailment
  
  – A KB $\mathcal{K}$ is satisfiable iff there exists a model $M$ s.t. $M \models \mathcal{K}$
  
  – A concept $C$ is satisfiable w.r.t. a KB $\mathcal{K}$ iff there exists a model $M = \langle D, \cdot^1 \rangle$ s.t. $M \models \mathcal{K}$ and $C^1 \neq \emptyset$
  
  – A KB $\mathcal{K}$ entails an axiom $\text{ax}$ (written $\mathcal{K} \models \text{ax}$) iff for every model $M$ of $\mathcal{K}$, $M \models \text{ax}$ (i.e., $M \models \mathcal{K}$ implies $M \models \text{ax}$)
DL Semantics

E.g.,
\[ T = \{ \text{Doctor} \sqsubseteq \text{Person}, \text{Parent} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Person}, \]
\[ \quad \text{HappyParent} \equiv \text{Parent} \sqcap \forall \text{hasChild}.(\text{Doctor} \sqcup \exists \text{hasChild}.\text{Doctor}) \} \]
\[ A = \{ \text{John:HappyParent}, \text{John hasChild Mary}, \text{John hasChild Sally}, \]
\[ \quad \text{Mary:}\neg\text{Doctor}, \text{Mary hasChild Peter}, \text{Mary:(} \leq 1 \text{ hasChild}) \}

\[ \checkmark - \mathcal{K} \models \text{John:Person} ? \]
\[ \checkmark - \mathcal{K} \models \text{Peter:Doctor} ? \]
\[ \checkmark - \mathcal{K} \models \text{Mary:HappyParent} ? \]
\[ - \text{What if we add “Mary hasChild Jane”?} \]
\[ \quad \mathcal{K} \models \text{Peter = Jane} \]
\[ - \text{What if we add “HappyPerson} \equiv \text{Person} \sqcap \exists \text{hasChild}.\text{Doctor”?} \]
\[ \quad \mathcal{K} \models \text{HappyPerson} \sqsubseteq \text{Parent} \]
DL and FOL

• Most DLs are subsets of C2
  – But reduction to C2 may be (highly) non-trivial
    • Trans(R) naively reduces to $\forall x, y, z. R(x, y) \land R(y, z) \rightarrow R(x, z)$

• Why use DL instead of C2?
  – Syntax is succinct and convenient for KR applications
  – Syntactic conformance guarantees being inside C2
    • Even if reduction to C2 is non-obvious
  – Different combinations of constructors can be selected
    • To guarantee decidability
    • To reduce complexity
  – DL research has mapped out the decidability/complexity landscape in great detail
    • See Evgeny Zolin’s DL Complexity Analyzer
      http://www.cs.man.ac.uk/~ezolin/dl/
Complexity of reasoning in Description Logics

Base description logic: Attributive Language with Complements

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

**Concept constructors:**
- $\exists R$: functionality ($\leq 1$ R)
- $\forall R$: (unqualified) number restrictions ($\geq n$ R), ($\leq n$ R)
- $\forall R.C$: qualified number restrictions ($\geq n$ R.C), ($\leq n$ R.C)
- $\exists R.C$: nominals: \{a\} or \{a_1, \ldots, a_n\} ("one-of" constructor)
- $\mu$ - least fixpoint operator: $\mu X.C$
- $R \subseteq S$: role-value-maps
- $f = g$: agreement of functional role chains ("same-as")

**Role constructors:**
- $\top$: role inverses: $R^-$
- $\bigcap R$: role intersection ($\forall R.S$)
- $\bigcup R$: role union: $R \cup S$
- $\neg R$: role complement: $\neg R$
- $\circ$: role chain (composition): $R \circ S$
- $\star$: reflexive-transitive closure ($\forall R.S$)
- $\mu X.C$: concept identity: $id(C)$
- $\mu X.C$: complex roles in number restrictions

**TBox** is internalized in extensions of ALCQIO, see [75, Lemma 4.12], [54, p.3]
- Empty TBox
- Acyclic TBox ($A \equiv C$, $A$ is a concept name; no cycles)
- General TBox ($C \subseteq D$ for arbitrary concepts $C$ and $D$)

**Role axioms (RBox):**
- $\top$: Role transitivity: $\text{Trans}(R)
- $\bigcup R$: Role hierarchy: $R \subseteq S$
- $\bigcap R$: Complex role inclusions: $R \subseteq S$, $R \subseteq S$
- $\star$: some additional features

---

Complexity of reasoning problems

<table>
<thead>
<tr>
<th>Reasoning problem</th>
<th>Complexity</th>
<th>Comments and references</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept satisfiability</td>
<td>NExpTime-complete</td>
<td><strong>Hardness</strong> of even ALCQIO is proved in [76, Corollary 4.13]. In that paper, the result is formulated for ALCQIO, but only number restrictions of the form ($\leq 1$R) are used in the proof. A different proof of the NExpTime-hardness for ALCQIO is given in [54] (even with 1 nominal, and role inverses not used in number restrictions). Upper bound for SHOIQ is proved in [77, Corollary 6.31] with numbers coded in unary (for binary coding, the upper bound remains an open problem for all logics in between ALCQIO and SHIQ). <strong>Important:</strong> in number restrictions, only simple roles (i.e. which are neither transitive nor have a transitive subrole) are allowed; otherwise we gain undecidability even in SHIQ; see [46]. <strong>Remark:</strong> recently [47] it was observed that, in many cases, one can use transitive roles in number restrictions – and still have a decidable logic! So the above notion of a simple role could be substantially extended.</td>
</tr>
<tr>
<td>ABox consistency</td>
<td>NExpTime-complete</td>
<td>By reduction to concept satisfiability problem in presence of nominals shown in [69, Theorem 3.7].</td>
</tr>
</tbody>
</table>
Complexity Measures

• **Taxonomic** complexity
  Measured w.r.t. total size of “schema” axioms

• **Data** complexity
  Measured w.r.t. total size of “data” facts

• **Query** complexity
  Measured w.r.t. size of query

• **Combined** complexity
  Measured w.r.t. total size of KB (plus query if appropriate)
Complexity Classes

• LogSpace, PTime, NP, PSpace, ExpTime, etc
  – worst case for a given problem w.r.t. a given parameter
  – X-hard means at-least this hard (could be harder);
    in X means no harder than this (could be easier);
    X-complete means both hard and in, i.e., exactly this hard
  • e.g., SROIQ KB satisfiability is 2NExpTime-complete w.r.t.
    combined complexity and NP-hard w.r.t. data complexity

• Note that:
  – this is for the worst case, not a typical case
  – complexity of problem means we can never devise a more
    efficient (in the worst case) algorithm
  – complexity of algorithm may, however, be even higher
    (in the worst case)
DLs and Ontology Languages
DLs and Ontology Languages

- W3C’s OWL 2 (like OWL, DAML+OIL & OIL) based on DL
  - OWL 2 based on SROIQ, i.e., ALC extended with transitive roles, a role box nominals, inverse roles and qualified number restrictions
  - OWL 2 EL based on EL
  - OWL 2 QL based on DL-Lite
  - OWL 2 EL based on DLP
- OWL was based on SHOIN
  - only simple role hierarchy, and unqualified NRs
# Class/Concept Constructors

<table>
<thead>
<tr>
<th>OWL Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${john} \sqcup {mary}$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq nP$</td>
<td>$\leq 1$ hasChild</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq nP$</td>
<td>$\geq 2$ hasChild</td>
</tr>
</tbody>
</table>
### Ontology Axioms

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>subClassOf</td>
<td>$C_1 \sqsubseteq C_2$</td>
<td>Human $\sqsubseteq$ Animal $\sqcap$ Biped</td>
</tr>
<tr>
<td>equivalentClass</td>
<td>$C_1 \equiv C_2$</td>
<td>Man $\equiv$ Human $\sqcap$ Male</td>
</tr>
<tr>
<td>subPropertyOf</td>
<td>$P_1 \sqsubseteq P_2$</td>
<td>hasDaughter $\sqsubseteq$ hasChild</td>
</tr>
<tr>
<td>equivalentProperty</td>
<td>$P_1 \equiv P_2$</td>
<td>cost $\equiv$ price</td>
</tr>
<tr>
<td>transitiveProperty</td>
<td>$P^+ \sqsubseteq P$</td>
<td>ancestor$^+$ $\sqsubseteq$ ancestor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OWL Syntax</th>
<th>DL Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>$a : C$</td>
<td>John : Happy-Father</td>
</tr>
<tr>
<td>property</td>
<td>$\langle a, b \rangle : R$</td>
<td>$\langle$John, Mary$\rangle :$ has-child</td>
</tr>
</tbody>
</table>

- An **Ontology** is *usually* considered to be a TBox
  - but an **OWL** ontology is a mixed set of TBox and ABox axioms
Other OWL Features

- XSD datatypes and (in OWL 2) facets, e.g.,
  - integer, string and (in OWL 2) real, float, decimal, datetime, …
  - minExclusive, maxExclusive, length, …
  - PropertyAssertion(hasAge Meg "17"^^xsd:integer )
  - DatatypeRestriction(xsd:integer xsd:minInclusive "5"^^xsd:integer xsd:maxExclusive "10"^^xsd:integer )

These are equivalent to (a limited form of) DL concrete domains

- Keys
  - E.g., HasKey(Vehicle Country LicensePlate)
    - Country + License Plate is a unique identifier for vehicles

This is equivalent to (a limited form of) DL safe rules
E.g., Person $\sqcap \forall$hasChild.(Doctor $\sqcap \exists$hasChild.Doctor):

```xml
<owl:Class>
  <owl:intersectionOf rdf:parseType="collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType="collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```
Complexity/Scalability

• From the complexity navigator we can see that:
  – OWL (aka SHOIN) is \text{NExpTime-complete}
  – OWL Lite (aka SHIF) is \text{ExpTime-complete} (oops!)
  – OWL 2 (aka SROIQ) is \text{2NExpTime-complete}
  – OWL 2 EL (aka EL) is \text{PTIME-complete} (robustly scalable)
  – OWL 2 RL (aka DLP) is \text{PTIME-complete} (robustly scalable)
    • And implementable using rule based technologies
e.g., rule-extended DBs
  – OWL 2 QL (aka DL-Lite) is in \text{AC}^0 \text{ w.r.t. size of data}
    • same as DB query answering -- nice!
Why (Description) Logic?

- OWL exploits results of 20+ years of DL research
  - Well defined (model theoretic) semantics

<table>
<thead>
<tr>
<th>Constructor</th>
<th>DL Syntax</th>
<th>Example</th>
<th>FOL Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersectionOf</td>
<td>$C_1 \sqcap \ldots \sqcap C_n$</td>
<td>Human $\sqcap$ Male</td>
<td>$C_1(x) \wedge \ldots \wedge C_n(x)$</td>
</tr>
<tr>
<td>unionOf</td>
<td>$C_1 \sqcup \ldots \sqcup C_n$</td>
<td>Doctor $\sqcup$ Lawyer</td>
<td>$C_1(x) \vee \ldots \vee C_n(x)$</td>
</tr>
<tr>
<td>complementOf</td>
<td>$\neg C$</td>
<td>$\neg$ Male</td>
<td>$\neg C(x)$</td>
</tr>
<tr>
<td>oneOf</td>
<td>${x_1} \sqcup \ldots \sqcup {x_n}$</td>
<td>${john} \sqcup {mary}$</td>
<td>$x = x_1 \vee \ldots \vee x = x_n$</td>
</tr>
<tr>
<td>allValuesFrom</td>
<td>$\forall P.C$</td>
<td>$\forall$ hasChild.Doctor</td>
<td>$\forall y. P(x, y) \rightarrow C(y)$</td>
</tr>
<tr>
<td>someValuesFrom</td>
<td>$\exists P.C$</td>
<td>$\exists$ hasChild.Lawyer</td>
<td>$\exists y. P(x, y) \wedge C(y)$</td>
</tr>
<tr>
<td>maxCardinality</td>
<td>$\leq np$</td>
<td>$\leq 1$ hasChild</td>
<td>$\exists y. P(x, y)$</td>
</tr>
<tr>
<td>minCardinality</td>
<td>$\geq np$</td>
<td>$\geq 2$ hasChild</td>
<td>$\exists y. P(x, y)$</td>
</tr>
</tbody>
</table>
Why (Description) Logic?

- OWL exploits results of 20+ years of DL research
  - Well defined (model theoretic) semantics
  - Formal properties well understood (complexity, decidability)

I can’t find an efficient algorithm, but neither can all these famous people.

Why (Description) Logic?

- **OWL** exploits results of 20+ years of DL research
  - Well defined (model theoretic) **semantics**
  - **Formal properties** well understood (complexity, decidability)
  - Known reasoning algorithms

| □-rule | if 1. \((C_1 \cap C_2) \in \mathcal{L}(v), v\) is not indirectly blocked, and  
|        | 2. \(\{C_1, C_2\} \notin \mathcal{L}(v)\)  
|        | then \(\mathcal{L}(v) \rightarrow \mathcal{L}(v) \cup \{C_1, C_2\}\).  |
| □-rule | if 1. \((C_1 \cup C_2) \in \mathcal{L}(v), v\) is not indirectly blocked, and  
|        | 2. \(\{C_1, C_2\} \cap \mathcal{L}(v) = \emptyset\)  
|        | then \(\mathcal{L}(v) \rightarrow \mathcal{L}(v) \cup \{E\}\) for some \(E \in \{C_1, C_2\}\).  |
| ∃-rule | if 1. ∃r.C \(\in \mathcal{L}(v_1)\), \(v_1\) is not blocked, and  
|        | 2. \(v_1\) has no safe r-neighbour \(v_2\) with \(C \in \mathcal{L}(v_1)\),  
|        | then create a new node \(v_2\) and an edge \((v_1, v_2)\)  
|        | with \(\mathcal{L}(v_2) = \{C\}\) and \(\mathcal{L}(v_1, v_2) = \{r\}\).  |
| ∀-rule | if 1. ∀r.C \(\in \mathcal{L}(v_1)\), \(v_1\) is not indirectly blocked, and  
|        | 2. there is an r-neighbour \(v_2\) of \(v_1\) with \(C \notin \mathcal{L}(v_2)\)  
|        | then \(\mathcal{L}(v_2) \rightarrow \mathcal{L}(v_2) \cup \{C\}\).  |
| ∀+.-rule | if 1. ∀r.C \(\in \mathcal{L}(v_1)\), \(v_1\) is not indirectly blocked, and  
|        | 2. there is some role \(r'\) with\( Trans(r')\) and \(r' \sqsubseteq r\)  
|        | 3. there is an r'-neighbour \(v_2\) of \(v_1\) with \(\forall r'.C \notin \mathcal{L}(v_2)\)  
|        | then \(\mathcal{L}(v_2) \rightarrow \mathcal{L}(v_2) \cup \{\forall r'.C\}\).  |
| choose-rule | if 1. \(\exists n. r.C \in \mathcal{L}(v_1)\), \(v_1\) is not indirectly blocked, and  
|            | 2. there is an r-neighbour \(v_2\) of \(v_1\) with \(\{C, \neg C\} \cap \mathcal{L}(v_2) = \emptyset\)  
|            | then \(\mathcal{L}(v_2) \rightarrow \mathcal{L}(v_2) \cup \{E\}\) for some \(E \in \{C, \neg C\}\).  |
| ≥.-rule | if 1. \(\geq n. r.C \in \mathcal{L}(v)\), \(v\) is not blocked, and  
|           | 2. there are not \(n\) safe r-neighbours \(v_1, \ldots, v_n\) of \(v\)  
|           | with \(C \in \mathcal{L}(v_i)\) and \(v_i \neq v_j\) for \(1 \leq i < j \leq n\).  |
Why (Description) Logic?

• OWL exploits results of 20+ years of DL research
  – Well defined (model theoretic) **semantics**
  – **Formal properties** well understood (complexity, decidability)
  – Known **reasoning algorithms**
  – **Scalability** demonstrated by **implemented systems**
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:
Tools, Tools, Tools

Major benefit of OWL has been huge increase in range and sophistication of tools and infrastructure:

• Editors/development environments
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners

Hermit

Racer

KAON2

FaCT++

Pellet

CEL
Tools, Tools, Tools

Major benefit of OWL has been huge increase in range and sophistication of tools and infrastructure:

• Editors/development environments

• Reasoners

• Explanation, justification and pinpointing
Tools, Tools, Tools

**Major benefit** of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners
- Explanation, justification and pinpointing
- Integration and modularisation
Tools, Tools, Tools

Major benefit of OWL has been huge increase in range and sophistication of tools and infrastructure:

- Editors/development environments
- Reasoners
- Explanation, justification and pinpointing
- Integration and modularisation
- APIs, in particular the **OWL API**
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience

**3D Analysis of Patterns of Gene Expression**

**Ontology of Zebrafish Developmental Anatomy**

<table>
<thead>
<tr>
<th></th>
<th>20 somite</th>
<th>...Head</th>
<th>Perihiral Nervous System</th>
</tr>
</thead>
<tbody>
<tr>
<td>trigeminal (V) ganglion</td>
<td>...Head</td>
<td></td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>Rohon-Beard neurons</td>
<td>20 somite</td>
<td></td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>primary motorneurons</td>
<td>20 somite</td>
<td>...Head</td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>brain</td>
<td>...Head</td>
<td>14 somite</td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>hindbrain</td>
<td>14 somite</td>
<td>...Head</td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>midbrain</td>
<td>14 somite</td>
<td>...Head</td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>forebrain</td>
<td>14 somite</td>
<td>...Head</td>
<td>Central Nervous System</td>
</tr>
<tr>
<td>ear</td>
<td>20 somite</td>
<td>...Head</td>
<td>Auditory</td>
</tr>
<tr>
<td>eye</td>
<td>14 somite</td>
<td>...Head</td>
<td>Visual</td>
</tr>
</tbody>
</table>

**Integration of Heterogeneous gene expression data**
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience, geography
Motivating Applications

• OWL playing **key role** in increasing number & range of applications
  – eScience, geography, engineering,

  Experience of OWL in use has identified restrictions:
  – on expressivity
  – on scalability

These restrictions are problematic in some applications
Research has now shown how some restrictions can be overcome
– W3C OWL WG is updating OWL accordingly
Motivating Applications

- OWL playing key role in increasing number & range of applications
  - eScience, geography, engineering, medicine

Experience of OWL in use has identified restrictions:
- on expressivity
- on scalability

These restrictions are problematic in some applications.

Research has now shown how some restrictions can be overcome:
- W3C OWL WG is updating OWL accordingly
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience, geography, engineering, medicine, biology

Experience of OWL in use has identified restrictions:
- on expressivity
- on scalability

These restrictions are problematic in some applications.

Research has now shown how some restrictions can be overcome:
- W3C OWL WG is updating OWL accordingly
Motivating Applications

- OWL playing **key role** in increasing number & range of applications
  - eScience, geography, engineering, medicine, biology, defence, …

Experience of OWL in use has identified restrictions:
- on expressivity
- on scalability

These restrictions are problematic in some applications.

Research has now shown how some restrictions can be overcome:
- W3C OWL WG is updating OWL accordingly.
NHS £6.2 £12 Billion IT Programme

Key component is “Care Records Service”

• “Live, interactive patient record service accessible 24/7”
• Patient data distributed across local and national DBs
  – Diverse applications support radiology, pharmacy, etc
  – Applications exchange “semantically rich clinical information”
  – Summaries sent to national database
• SNOMED-CT ontology provides clinical **vocabulary**
  – Data uses terms drawn from ontology
  – New terms with well defined meaning can be added “on the fly”
Ontology -v- Database
Obvious Database Analogy

• Ontology axioms analogous to DB schema
  – Schema describes structure of and constraints on data
• Ontology facts analogous to DB data
  – Instantiates schema
  – Consistent with schema constraints
• But there are also important differences…
Obvious Database Analogy

Database:

- Closed world assumption (CWA)
  - Missing information treated as false
- Unique name assumption (UNA)
  - Each individual has a single, unique name
- Schema behaves as constraints on structure of data
  - Define legal database states

Ontology:

- Open world assumption (OWA)
  - Missing information treated as unknown
- No UNA
  - Individuals may have more than one name
- Ontology axioms behave like implications (inference rules)
  - Entail implicit information
Database -v- Ontology

E.g., given the following ontology/schema:

- \( \text{HogwartsStudent} \equiv \text{Student} \sqcap \exists \text{attendsSchool.Hogwarts} \)
- \( \text{HogwartsStudent} \subseteq \forall \text{hasPet.(Owl or Cat or Toad)} \)
- \( \text{hasPet} \equiv \text{isPetOf}^- \) (i.e., hasPet inverse of isPetOf)
- \( \exists \text{hasPet.} \top \subseteq \text{Human} \) (i.e., domain of hasPet is Human)
- \( \text{Phoenix} \sqsubseteq \forall \text{isPetOf.Wizard} \) (i.e., only Wizards have Phoenix pets)
- \( \text{Muggle} \sqsubseteq \neg \text{Wizard} \) (i.e., Muggles and Wizards are disjoint)
Database -v- Ontology

And the following facts/data:

HarryPotter: Wizard
DracoMalfoy: Wizard
HarryPotter hasFriend RonWeasley
HarryPotter hasFriend HermioneGranger
HarryPotter hasPet Hedwig

Query: Is Draco Malfoy a friend of HarryPotter?

- DB: No
- Ontology: Don’t Know

OWA (didn’t say Draco was not Harry’s friend)
Database -v- Ontology

And the following facts/data:

HarryPotter: Wizard
DracoMalfoy: Wizard
HarryPotter hasFriend RonWeasley
HarryPotter hasFriend HermioneGranger
HarryPotter hasPet Hedwig

Query: How many friends does Harry Potter have?

- DB: 2
- Ontology: at least 1

No UNA (Ron and Hermione may be 2 names for same person)
Database -v- Ontology

And the following facts/data:

- HarryPotter: Wizard
- DracoMalfoy: Wizard
- HarryPotter hasFriend RonWeasley
- HarryPotter hasFriend HermioneGranger
- HarryPotter hasPet Hedwig

\[\text{RonWeasley} \neq \text{HermioneGranger}\]

**Query**: How many friends does Harry Potter have?

- DB: 2
- Ontology: at least 2

OWA (Harry may have more friends we didn’t mention yet)
Database -v- Ontology

And the following facts/data:

HarryPotter: Wizard
DracoMalfoy: Wizard
HarryPotter hasFriend RonWeasley
HarryPotter hasFriend HermioneGranger
HarryPotter hasPet Hedwig

RonWeasley ≠ HermioneGranger

 anállysis:

HarryPotter: ∀hasFriend.{RonWeasley} U {HermioneGranger}

Query: How many friends does Harry Potter have?

- DB: 2
- Ontology: 2!
Database -v- Ontology

**Inserting** new facts/data:

Dumbledore: Wizard
Fawkes: Phoenix
Fawkes isPetOf Dumbledore

What is the response from DBMS?

– Update rejected: *constraint violation*

  Domain of hasPet is Human; Dumbledore is not Human (CWA)

What is the response from Ontology reasoner?

– **Infer** that Dumbledore is Human (domain restriction)

– Also infer that Dumbledore is a Wizard (only a Wizard can have a pheonix as a pet)
DB Query Answering

• Schema plays **no role**
  – Data must explicitly satisfy schema constraints

• Query answering amounts to **model checking**
  – I.e., a “look-up” against the data

• Can be very **efficiently implemented**
  – Worst case complexity is low (logspace) w.r.t. size of data
Ontology Query Answering

• Ontology axioms play a powerful and crucial role
  – Answer may include implicitly derived facts
  – Can answer conceptual as well as extensional queries
    • E.g., Can a Muggle have a Phoenix for a pet?

• Query answering amounts to theorem proving
  – I.e., logical entailment

• May have very high worst case complexity
  – E.g., for OWL, NP-hard w.r.t. size of data
    (upper bound is an open problem)
  – Implementations may still behave well in typical cases
  – Fragments/profiles may have much better complexity
Ontology Based Information Systems

• Analogous to relational database management systems
  – Ontology \approx schema; instances \approx data

• Some important (dis)advantages
  + (Relatively) easy to maintain and update schema
    • Schema plus data are integrated in a logical theory
  + Query answers reflect both schema and data
  + Can deal with incomplete information
  + Able to answer both intensional and extensional queries
  – Semantics can seem counter-intuitive, particularly w.r.t. data
    • Open -v- closed world; axioms -v- constraints
  – Query answering (logical entailment) may be much more difficult
    • Can lead to scalability problems with expressive logics
Ontology Based Information Systems

• Analogous to relational database management systems
  – Ontology ≈ schema; instances ≈ data

• Some important (dis)advantages
  + (Relatively) easy to maintain and update schema
  + Schema plus data are integrated in a logical theory
  + Query answers reflect both schema and data
  + Can deal with incompleteness
  + Able to answer both intensional and extensional queries
  – Semantics can seem counter-intuitive, particularly w.r.t. data
    • Open -v- closed world; axioms -v- constraints
    – Query answering (logical entailment) may be much more difficult
      • Can lead to scalability problems with expressive logics