2nd World Congress and School on Universal Logic

DDL: Embracing Action Formalism into Description Logic

Zhongzhi Shi   Liang Chang
shizz@ics.ict.ac.cn

Institute of Computing Technology
Chinese Academy of Sciences

2007/8/22
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
Intelligence encompasses abilities such as:

- understanding language
- perception
- learning
- reasoning
The Dream in AI

Look for a formal logic can describe:

- Perception
- Reasoning
- Behavior
- Decision making
Formal Logics

- First order logic
- Modal logic
- Temporal logic
- Prolog
- GOLOG
- Description logic
Semantic Web Layers

- May 2001
- Semantic Web
- 10 Feb 2004 W3C: OWL

by Tim Berners-Lee
Semantic Web Layers

- Query: SPARQL
- ontology: OWL
- Rules: RIF
- RDF-S
- Data interchange: RDF
- XML
- URI
- Unicode
- Crypto
- Proof
- Unifying Logic
- Trust
- User Interface & applications

2006
Web Science

by Tim Berners-Lee
Universal Logic

Beziau, Jean-Yves

- Universal Logic is not a new logic, but a general theory of logics, considered as mathematical structures.

- The name was introduced about ten years ago, but the subject is as old as the beginning of modern logic: Alfred Tarski and other Polish logicians such as Adolf Lindenbaum developed a general theory of logics at the end of the 1920s based on consequence operations and logical matrices.
Beziau, Jean-Yves: Main line of research of Uni. Logic

- Foundations of mathematics
- Different systems of logic
- General theory and tools for logics
Unifying Logic

Universal Logic?

Web Science

Unifying Logic
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
Logical Foundation of the Semantic Web

- OWL Lite: corresponding to the description logic SHIF(D)
  - Classification hierarchy
  - Simple constraints
- OWL DL: corresponding to the description logic SHOIN(D)
  - Maximal expressiveness
  - While maintaining tractability
  - Standard formalisation
- OWL Full:
  - Very high expressiveness
  - Loosing tractability
  - Non-standard formalisation
  - All syntactic freedom of RDF (self-modifying)

OWL: Ontology language recommended by W3C
Decidable Subset of First-Order Logic
Model theoretic semantics by mapping to abstract domain
Provides Primitives for defining Conceptual Knowledge
  - Concept Expressions (Formulas with 1 free variable) for describing Sets of Objects
    - Boolean Operators: \( C \cap D, C \cup D, \neg C \)
    - Quantifiers: \( (\exists R.C), (\forall P.C) \)
    - Cardinality Constraints: \( (= n R), (> n R), (< n R), (\geq n R), (\leq n R) \)
  - Axioms define relations between concepts
    - Subsumption: \( C \subseteq D \)
    - Equivalence: \( C \equiv D \)
    - Disjointness: \( C \cap D \subseteq \perp \)
Short History of Description Logics

- Phase 1:
  - Incomplete systems (Back, Classic, Loom, \ldots)
  - Based on structural algorithms
- Phase 2:
  - Development of tableau algorithms and complexity results
  - Tableau-based systems for Pspace logics (e.g., Kris, Crack)
  - Investigation of optimisation techniques
- Phase 3:
  - Tableau algorithms for very expressive DLs
  - Highly optimised tableau systems for ExpTime logics (e.g., FaCT, DLP, Racer)
  - Relationship to modal logic and decidable fragments of FOL
<table>
<thead>
<tr>
<th>Constructor</th>
<th>Syntax</th>
<th>Semantics</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic concept</td>
<td>$A$</td>
<td>$A^I \subseteq \triangle^I$</td>
<td>Human</td>
</tr>
<tr>
<td>Atomic Relation</td>
<td>$R$</td>
<td>$R^I \subseteq \triangle^I \times \triangle^I$</td>
<td>has-child</td>
</tr>
<tr>
<td>$\cap$</td>
<td>$C \cap D$</td>
<td>$C^I \cap D^I$</td>
<td>Human $\cap$ Male</td>
</tr>
<tr>
<td>$\cup$</td>
<td>$C \cup D$</td>
<td>$C^I \cup D^I$</td>
<td>Doctor $\cup$ Lawyer</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$\neg C$</td>
<td>$\triangle^I \setminus C$</td>
<td>$\neg$ Male</td>
</tr>
<tr>
<td>$\exists$</td>
<td>$\exists R.C$</td>
<td>${x</td>
<td>\exists y. &lt;x,y&gt; \in R^I \land y \in C^I}$</td>
</tr>
<tr>
<td>$\forall$</td>
<td>$\forall R.C$</td>
<td>${x</td>
<td>\forall y. &lt;x,y&gt; \in R^I \Rightarrow y \in C^I}$</td>
</tr>
</tbody>
</table>
Examples

woman \equiv person \cap female
man \equiv person \cap \neg woman
mother \equiv woman \cap \exists hasChild\cdot person
father \equiv man \cap \exists hasChild\cdot person
DL Knowledge Base

DL Knowledge Base (KB) normally separated into 2 parts:

- TBox is a set of axioms describing structure of domain (i.e., a conceptual schema), e.g.:
  - HappyFather ≡ Man ⊓ ∃hasChild.Female ⊓ …
  - Elephant ⊆ Animal ⊓ Large ⊓ Grey
  - transitive(ancestor)

- ABox is a set of axioms describing a concrete situation (data), e.g.:
  - John:HappyFather
  - <John,Mary>:hasChild

- Separation has no logical significance
  - But may be conceptually and implementationally convenient.
### Tableau Rules for ALC

<table>
<thead>
<tr>
<th>Name</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>if ((C_1 \cap C_2)(x) \in S), and (C_1(x) \notin S) or (C_2(x) \notin S), then (S := S \cup {C_1(x), C_2(x)})</td>
</tr>
<tr>
<td>□</td>
<td>if ((C_1 \sqcup C_2)(x) \in S), and (C_1(x) \notin S), (C_2(x) \notin S), then (S := S \cup {C_1(x)}) or (S := S \cup {C_2(x)})</td>
</tr>
<tr>
<td>∃</td>
<td>if ((\exists R.C)(x) \in S), and not exist (y) (R(x,y) \in S), (C(y) \in S), then (S) has (z), (S := S \cup {C(z), R(x,z)})</td>
</tr>
<tr>
<td>∀</td>
<td>if ((\forall R.C)(x) \in S), then (S := S \cup {C(y) \mid R(x,y) \in S\ and\ C(y) \notin S})</td>
</tr>
</tbody>
</table>
DL Architecture

Knowledge Base

Tbox (schema)
- Man ⊑ Human ⊑ Male
- Happy-Father ⊑ Man ⊑ 9 has-child
- Female ⊑ ...

Abox (data)
- John : Happy-Father
- John, Mary : has-child

Inference System

Interface
Ontology language: OWL

- **OWL Light**
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction
  - (in)equality
  - cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric
  - hasValue
  - someValuesFrom
  - allValuesFrom

- **Language Layers**
  - RDF Schema
    - Full
    - DL
    - Lite

- **OWL DL**
  - Negation
  - Disjunction
  - Full Cardinality
  - Enumerated types

- **OWL Full**
  - Allow meta-classes etc
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
Embracing actions into description logics, with the feature that actions are treated as citizens.

Therefore, there are three kinds of first-class citizens:
- Concepts
- Formulas
- Actions

Here we investigate a dynamic description logic D-ALCO@, corresponding to the description logic ALCO@.
Primitive symbols of D-ALCO@ are a set $N_C$ of concept names, a set $N_R$ of role names, and a set $N_I$ of individual names.

**Concepts** are formed with the syntax rule:

$$C, C' \rightarrow C_i \{p\} \@ p C \neg C | C \sqcup C' \exists R.C | < \pi > C \quad (1)$$

where $C_i \in N_C$, $p \in N_I$, $R \in N_R$, and $\pi$ is an action.

**Abbreviations:**

$$C \cap C' = \neg (\neg C \sqcup \neg C')$$
$$\forall R.C = \neg \exists R.C$$
$$T = C \sqcup \neg C$$
$$\bot = \neg T$$
Syntax

- **Formulas** are formed with the syntax rule:

\[ \varphi, \varphi' \rightarrow C(p) \mid R(p, q) \mid \neg \varphi \mid \varphi \lor \varphi' \mid < \pi > \varphi \]  

(2)

where \( C \) is a concept, \( p, q \in N_I \), \( R \in N_R \), and \( \pi \) is an action.

- **Abbreviations:**

\[
\begin{align*}
\varphi \land \varphi' &= \neg(\neg \varphi \lor \neg \varphi') \\
[\pi] \varphi &= \neg < \pi > \neg \varphi \\
\varphi \rightarrow \varphi' &= \neg \varphi \lor \varphi' \\
\text{true} &= \varphi \lor \neg \varphi \\
\text{false} &= \neg \text{true}
\end{align*}
\]
An atomic action is a pair $(P, E)$, where,

- $P$ is a finite set of formulas, describing the precondition of the action,

- $E$ is a finite set of formulas, describing the effect of the action, with each formula be of form $A(p)$, $\neg A(p)$, $R(p, q)$, or $\neg R(p, q)$, where $A \in N_C$, $R \in N_R$, and $p, q \in N_I$.

- let $P = \{\phi_1, \ldots, \phi_n\}$ and $E = \{\phi_1, \ldots, \phi_m\}$, then $P$ and $E$ subject to the constraint that $\phi_1 \land \ldots \land \phi_n \rightarrow \phi_k$ for all $k$ with $1 \leq k \leq m$. 
Actions are formed with the syntax rule:

\[ \pi, \pi' \longrightarrow (P, E) | \varphi? | \pi \cup \pi' | \pi; \pi' | \pi^* \] (3)

where \((P, E)\) is an atomic action, \(\varphi\) is a formula.

Abbreviations:

if \(\varphi\) then \(\pi\) else \(\pi' = (\varphi?; \pi) \cup ((\neg \varphi)?; \pi')\)
while \(\varphi\) do \(\pi = (\varphi?; \pi)^* ; (\neg \varphi)?\)
A model for dynamic description logic is a pair $M=\langle W, I \rangle$, where $W$ is a set of states, $I$ associates with each state $w \in W$ an interpretation:

$$I(w) = (\triangle^I, C_0^{I(w)}, \ldots, R_0^{I(w)}, \ldots, p_0^I, \ldots),$$

with $C_i^{I(w)} \subseteq \triangle^I$ for each $C_i \in N_c$,

$R_i^{I(w)} \subseteq \triangle^I \times \triangle^I$ for each $R_i \in N_R$,

and $p_i^I \in \triangle^I$ for each $p_i \in N_I$;

Furthermore, each action $\pi$ is interpreted as a binary relation $\pi^I \subseteq W \times W$. 
Semantics: Let $M=\langle W, I \rangle$ be a model and $w$ a state in $W$, then:
- the value $C^I(w)$ of a concept $C$ is defined inductively as:

1. $\{p\}^I(w) = \{p^I\}$;
2. If $p^I \in C^I(w)$ then $(\@_p C)^I(w) = \triangle^I$, else $(\@_p C)^I(w) = \emptyset$;
3. $(\neg C)^I(w) = \triangle^I - C^I(w)$;
4. $(C \sqcup D)^I(w) = C^I(w) \cup D^I(w)$;
5. $(\exists R.C)^I(w) = \{x | \exists y. ((x, y) \in R^I(w) \land y \in C^I(w))\}$;
6. $(< \pi > C)^I(w) = \{p | \exists w' \in W. ((w, w') \in \pi^I \land p \in C^I(w'))\}$;
the truth-relation \((M, w) \models \phi\) (or simply \(w \models \phi\) if \(M\) is understood) for a formula \(\phi\) is defined inductively as:

\[
\begin{align*}
(7) \quad & (M, w) \models C(p) \iff p^I \in C^I(w); \\
(8) \quad & (M, w) \models R(p, q) \iff (p^I, q^I) \in R^I(w); \\
(9) \quad & (M, w) \models \neg \varphi \iff (M, w) \models \varphi \text{ not holds}; \\
(10) \quad & (M, w) \models \varphi \lor \psi \iff (M, w) \models \varphi \text{ or } (M, w) \models \psi; \\
(11) \quad & (M, w) \models <\pi > \varphi \iff \exists w' \in W.((w, w') \in \pi^I \land (M, w') \models \varphi);
\end{align*}
\]
the binary relation $\pi^I$ for an action $\pi$ is defined inductively as:

(12) Let $S$ be a formula set, then, $(M, w) \models S$ iff $(M, w) \models \varphi_i$ for all $\varphi_i \in S$;

(13) Let $(P, E)$ be an atomic action with $E = \{\phi_1, \ldots, \phi_m\}$, then, $(P, E)^I = \{(w_1, w_2) \in W \times W \mid (M, w_1) \models P, C^I(w_2) = C^I(w_1) \cup C^+ - C^- \text{ for each primitive concept name } C \in N_{CP},$ and $R^I(w_2) = R^I(w_1) \cup R^+ - R^- \text{ for each role name } R \in N_R\}$, where,

- $C^+ = \{ p^I \mid C(p) \in E \}$,
- $C^- = \{ p^I \mid \neg C(p) \in E \}$,
- $R^+ = \{ (p^I, q^I) \mid R(p, q) \in E \}$,
- $R^- = \{ (p^I, q^I) \mid \neg R(p, q) \in E \}$;

(14) $(\varphi?)^I = \{(w_1, w_1) \in W \times W \mid (M, w_1) \models \varphi\};$

(15) $(\pi \cup \pi')^I = \pi^I \cup \pi'^I$;

(16) $(\pi; \pi')^I = \pi^I \circ \pi'^I$;

(17) $(\pi^*)^I = \text{reflexive transitive closure of } \pi^I$. 
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
Knowledge Base: \( K = \langle T, Ac, A \rangle \)

1. \( T \), also called TBox, is a finite set composed of *terminological axioms*, with each axiom be of form \( D \equiv C \), where \( D \) is a new defined concept name, \( C \) is a concept.

   Starting with concept names contained in the set \( N_C \), many new concept names can be inductively defined.

   Because actions can be used as model operators for the construction of concepts, concepts with dynamic meanings can be described. E.g.,

   \[
   \text{PhdCandidate} = \text{student} \sqcap \exists \text{forDegree}.\text{doctorDegree} \\
   \sqcap <\text{PhdDefend}>\text{doctor}
   \]
2 Ac, also called ActionBox, is a finite set composed of action axioms, with each axiom be of form $\alpha (v_1, \ldots, v_n) \equiv \pi$, where $\alpha$ is a new defined action name, $v_1, \ldots, v_n$ are individuals that the action operate on, and $\pi$ is an action.

Both atomic action and complex action can be defined with action axioms. Therefore, actions can be described and published as a sort of knowledge.

E.g., graduate($v$) = (\{student($v$)\}, \{\neg student($v$)\}) ;

buyBicycle($u$, $v$) = (\{bicycle($v$), \neg owns($u$, $v$), wants($u$, $v$), hasMoney($u$)\}, \{owns($u$, $v$)\});
A, also called ABox, is a finite set composed of *individual assertions*, with each assertion be of form $C(p)$, $R(p, q)$, or $\neg R(p, q)$;

A concrete situation is described with these assertions, by describing the properties of all the concerned individuals.
Semantics of the Knowledge Base

Let $K = \langle T, Ac, A \rangle$ be a knowledge base, $M = (W, I)$ a model and $w$ a state in $W$. Then,

1. $(M,w)$ satisfies a terminological axioms $D \equiv C$, written as $(M,w) \models D \equiv C$, if and only if $D^I(w) = C^I(w)$;
2. $(M,w)$ satisfies the TBox $T$, written as $(M,w) \models T$, if and only if $(M,w) \models D \equiv C$ for all $D \equiv C \in T$;
3. $(M,w)$ satisfies the ABox $A$, written as $(M,w) \models A$, if and only if $(M,w) \models \phi$ for all $\phi \in A$;
4. $M$ satisfies the TBox $T$, written as $M \models T$, if and only if $(M,w) \models T$ for all $w \in M$;
5. $(M,w)$ satisfies the knowledge base $K$, written as $(M,w) \models K$, if and only if $M \models T$ and $(M,w) \models A$. 

2007/8/22 Zhongzhi Shi: DDL
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
This prefixed tableau calculus is an elaborated combination of:

- the standard tableau calculus for \textit{ALCO@},
- the prefixed tableaux for propositional dynamic logic, and
- the embodiment of the possible models approach for interpreting actions.

The satisfiability problem and other reasoning problems can be realized based on this calculus.
Some notations:

- A prefix $\sigma \cdot \varepsilon$ is composed of a sequential action $\sigma$ and a set of effects $\varepsilon$, and is formed inductively with the syntax rule:

$$\sigma \cdot \varepsilon \rightarrow (\emptyset, \emptyset) \cdot \emptyset \sigma; (P, E) \cdot (\varepsilon - \{\neg \varphi | \varphi \in E\}) \cup E$$  \hspace{1cm} (4)

where $(\emptyset, \emptyset)$ and $(P, E)$ are atomic actions, $\sigma ; (P, E)$ is a sequential action, and $(\sigma - \{\neg \phi | \phi \in E\}) \cup E$ is a set of effects.

- A prefixed formula is a pair $\sigma \cdot \varepsilon : \phi$, where $\sigma \cdot \varepsilon$ is a prefix, $\phi$ is a formula.

- Prefixed tableau calculus for the dynamic description logic is shown in Figure 1, 2, 3, and 4. A tableau rule from them could be applied in the condition that the premise of this rule is hold.
### prefixed tableau calculus

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R(\neg\cdot)</td>
<td>If (\sigma.\varepsilon : (\neg(-C))(x) \in S), and (\sigma.\varepsilon : C(x) \notin S), then (S_1 := {\sigma.\varepsilon : C(x)} \cup S).</td>
</tr>
<tr>
<td>R({\cdot})</td>
<td>If (\sigma.\varepsilon : {q}(x) \in S), and (x \neq q \notin S), (q \neq x \notin S), then (S_1 := {x = q} \cup S[x/q]), here (S[x/q]) is obtained from (S) by replacing each occurrence of (x) in (S_{PF}, S_{nSC}) and (S_{VR}) by (q).</td>
</tr>
<tr>
<td>R(\neg{\cdot})</td>
<td>If (\sigma.\varepsilon : (\neg{q})(x) \in S), and and (x \neq q \notin S), (q \neq x \notin S), then (S_1 := S \cup {x \neq q}).</td>
</tr>
<tr>
<td>R(\exists\cdot)</td>
<td>If (\sigma.\varepsilon : (\exists_q C)(x) \in S), and (\sigma.\varepsilon : C(q) \notin S), then (S_1 := {\sigma.\varepsilon : C(q)} \cup S).</td>
</tr>
<tr>
<td>R(\neg\exists\cdot)</td>
<td>If (\sigma.\varepsilon : (\neg\exists_q C)(x) \in S), and (\sigma.\varepsilon : (\neg C)(q) \notin S), then (S_1 := {\sigma.\varepsilon : (\neg C)(q)} \cup S).</td>
</tr>
<tr>
<td>R(\cup\cdot)</td>
<td>If (\sigma.\varepsilon : (C_1 \cup C_2)(x) \in S), and (\sigma.\varepsilon : C_1(x) \notin S), (\sigma.\varepsilon : C_2(x) \notin S), then (S_1 := \sigma.\varepsilon : C_1(x) \cup S), (S_2 := \sigma.\varepsilon : C_2(x) \cup S).</td>
</tr>
<tr>
<td>R(\neg\cup\cdot)</td>
<td>If (\sigma.\varepsilon : (\neg(C_1 \cup C_2))(x) \in S), and (\sigma.\varepsilon : (\neg C_1)(x) \notin S) or (\sigma.\varepsilon : (\neg C_2)(x) \notin S), then (S_1 := {\sigma.\varepsilon : (\neg C_1)(x), \sigma.\varepsilon : (\neg C_2)(x)} \cup S).</td>
</tr>
<tr>
<td>R(\exists\cdot)</td>
<td>If ((\emptyset, \emptyset).\emptyset : (\exists R.C)(x) \in S), there is no (y) such that ((\emptyset, \emptyset).\emptyset : R(x, y) \in S) and ((\emptyset, \emptyset).\emptyset : C(y) \in S), then (S_1 := {(\emptyset, \emptyset).\emptyset : C(z), (\emptyset, \emptyset).\emptyset : R(x, z)} \cup S), (z) is a new individual name.</td>
</tr>
<tr>
<td>R(\neg\exists\cdot)</td>
<td>If ((\emptyset, \emptyset).\emptyset : (\neg\exists R.C)(x) \in S), there is a (y) with ((\emptyset, \emptyset).\emptyset : R(x, y) \in S) and ((\emptyset, \emptyset).\emptyset : (\neg C)(y) \notin S), then (S_1 := {(\emptyset, \emptyset).\emptyset : (\neg C)(y) \mid (\emptyset, \emptyset).\emptyset : R(x, y) \in S) and ((\emptyset, \emptyset).\emptyset : (\neg C)(y) \notin S}).</td>
</tr>
</tbody>
</table>

Figure 1: Rules for concepts
### prefixed tableau calculus

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_{\neg f})</td>
<td>If (\sigma.e : \neg(C(x)) \in S) and (\sigma.e : (\neg C)(x) \notin S), then (S_1 := {\sigma.e : (\neg C)(x)} \cup S).</td>
</tr>
<tr>
<td>(R_{\neg \neg f})</td>
<td>If (\sigma.e : \neg(\neg \varphi) \in S) and (\sigma.e : \varphi \notin S), then (S_1 := {\sigma.e : \varphi} \cup S).</td>
</tr>
<tr>
<td>(R_v)</td>
<td>If (\sigma.e : \varphi \lor \psi \in S), (\sigma.e : \varphi \notin S), and (\sigma.e : \psi \notin S), then (S_1 := {\sigma.e : \varphi} \cup S), (S_2 := {\sigma.e : \psi} \cup S).</td>
</tr>
<tr>
<td>(R_{\neg v})</td>
<td>If (\sigma.e : \neg(\varphi \lor \psi) \in S), and (\sigma.e : \neg \varphi \notin S) or (\sigma.e : \neg \psi \notin S), then (S_1 := {\sigma.e : \neg \varphi, \sigma.e : \neg \psi} \cup S).</td>
</tr>
<tr>
<td>(R_{&lt;;&gt; f})</td>
<td>If (\sigma.e : &lt; \pi_1; \pi_2 &gt; \varphi \in S) and (\sigma.e : &lt; \pi_1; &lt; \pi_2 &gt; \varphi \notin S), then (S_1 := {\sigma.e : &lt; \pi_1; &lt; \pi_2 &gt; \varphi} \cup S).</td>
</tr>
<tr>
<td>(R_{\neg&lt;;&gt; f})</td>
<td>If (\sigma.e : \neg &lt; \pi_1; \pi_2 &gt; \varphi \in S), and (\sigma.e : \neg &lt; \pi_1; &lt; \pi_2 &gt; \varphi \notin S), then (S_1 := {\sigma.e : \neg &lt; \pi_1; &lt; \pi_2 &gt; \varphi} \cup S).</td>
</tr>
<tr>
<td>(R_{&lt;;&gt; f})</td>
<td>If (\sigma.e : &lt; \varphi ? &gt; \varphi \in S), and (\sigma.e : &lt; \pi_1; &lt; \pi_2 &gt; \varphi \notin S), then (S_1 := {\sigma.e : &lt; \varphi ? &gt; \varphi} \cup S).</td>
</tr>
<tr>
<td>(R_{\neg&lt;;&gt; f})</td>
<td>If (\sigma.e : \neg &lt; \varphi ? &gt; \varphi \in S), and (\sigma.e : &lt; \varphi ? &gt; \varphi \notin S), then (S_1 := {\sigma.e : \neg &lt; \varphi ? &gt; \varphi} \cup S).</td>
</tr>
<tr>
<td>(R_{&lt;U&gt; f})</td>
<td>If (\sigma.e : &lt; \pi_1 \cup \pi_2 &gt; \varphi \in S), and (\sigma.e : &lt; \pi_2 &gt; \varphi \notin S), then (S_1 := {\sigma.e : &lt; \pi_1 \cup \pi_2 &gt; \varphi} \cup S), (S_2 := {\sigma.e : &lt; \pi_2 &gt; \varphi} \cup S).</td>
</tr>
<tr>
<td>(R_{\neg&lt;U&gt; f})</td>
<td>If (\sigma.e : \neg &lt; \pi_1 \cup \pi_2 &gt; \varphi \in S), and (\sigma.e : \neg &lt; \pi_1 &gt; \varphi \notin S) or (\sigma.e : \neg &lt; \pi_2 &gt; \varphi \notin S), then (S_1 := {\sigma.e : \neg &lt; \pi_1 &gt; \varphi, \sigma.e : \neg &lt; \pi_2 &gt; \varphi} \cup S).</td>
</tr>
</tbody>
</table>

**Figure 2: Rules for formulas**
prefixed tableau calculus

R\textsubscript{atom}\textsubscript{f} If \(\sigma.\varepsilon : (P, E) > \varphi \in S\), and \(\{\sigma.\varepsilon : \phi | \phi \in P\} \notin S\) or there is no prefix \(\sigma_i.\varepsilon_i\) such that both \(\varepsilon_i = (\varepsilon - \{\neg \varphi | \varphi \in E\}) \cup E\) and \(\sigma_i.\varepsilon_i : \varphi \in S\),
then, introduce a prefix \(\sigma'.\varepsilon': = \sigma; (P, E). (\varepsilon - \{\neg \varphi | \varphi \in E\}) \cup E\), and
generate a branch \(S_1 := \{\sigma.\varepsilon : \phi | \phi \in P\} \cup \{\sigma'.\varepsilon' : \varphi\} \cup \{\sigma'.\varepsilon' : \psi | \psi \in \varepsilon'\} \cup S\).

R\textsubscript{\neg atom}\textsubscript{f} If \(\sigma.\varepsilon : \neg (P, E) > \varphi \in S\), and \(\{\sigma : \neg \phi | \phi \in P\} \cap S = \emptyset\),
there is a prefix \(\sigma_i.\varepsilon_i\) with \(\varepsilon_i = (\varepsilon - \{\neg \varphi | \varphi \in E\}) \cup E\) and \(\sigma_i.\varepsilon_i : \neg \varphi \notin S\),
then, \(S_1 := \{\sigma_i.\varepsilon_i : \neg \varphi\} \cup S\), and
for each \(\phi \in P\) generate a branch \(S_\phi := \{\sigma.\varepsilon : \neg \phi\} \cup S\).

Figure 3: Forward generating rules

R\textsubscript{Back1} If \(\sigma.\varepsilon : D(x) \in S\), \(D\) is of form \(\exists R.C\) or \(\neg \exists R.C\), where \(C\) is a concept,
and \((\emptyset, \emptyset).\emptyset : D\text{Regress}(\sigma.\varepsilon)(x) \notin S\),
then, \(S_1 := \{(\emptyset, \emptyset).\emptyset : D\text{Regress}(\sigma.\varepsilon)(x)\} \cup S\).

R\textsubscript{Back2} If \(\sigma.\varepsilon : \varphi \in S\), \(\varphi\) is of form \(R(x, y), \neg R(x, y), C(x)\), or \((\neg C)(x)\) with \(C \in N_{CP}\),
and \(\varphi \notin \varepsilon\), \((\emptyset, \emptyset).\emptyset : \varphi \notin S\),
then, \(S_1 := \{(\emptyset, \emptyset).\emptyset : \varphi\} \cup S\).

Figure 4: Backward mapping rules
A formula $\phi$ is **satisfiable** if and only if there is a model $M = (W, I)$ and a state $w \in W$ such that $(M, w) \models \phi$.

Algorithm 1 (Deciding the satisfiability of a formula) Let $T$ and $Ac$ be acyclic TBox and ActionBox respectively, and $\varphi$ be a formula. Then, the satisfiability of $\varphi$ with respect to $T$ and $Ac$ is decided with the following steps:

1. Replace each occurrence of defined concept names in $\varphi$ with their definitions, result in a formula $\varphi'$.
2. Construct a branch $S' := \{((\emptyset, \emptyset), \emptyset : \varphi')\}$; If $S'$ is contradictory, exit the algorithm with the result “$\varphi$ is unsatisfiable”.
3. Construct an empty stack $SS$, push the branch $S'$ into $SS$.
4. Pop a branch from $SS$, let it be $S$, then:
   - if $S$ is not completed, then find a rule to apply to $S$, with the constraint that $R_{<atom>_f}$-rules can only be examined for applying as while as no other rules can be applied; For every new generated branch, replace each occurrence of defined concept names in it with the corresponding definitions, then, add the replaced branch into $SS$ if it is not contradictory;
   - if $S$ is completed but not ignorable, then exit the algorithm with the result “$\varphi$ is satisfiable”;
   - if $S$ is ignorable then discard it.
5. If $SS$ is empty, then exit the algorithm with the result “$\varphi$ is unsatisfiable”, else goto step 4.
Theorem 1: The satisfiability-checking algorithm is terminable, sound, and complete.

Theorem 2: Let \( \phi \) be a satisfiable formula. Then there exists a model \( M = (W, I) \) and a state \( w \in W \) such that \( (M, w) \models \phi \) and 

\[
\text{size}(M) \leq 2^{p(\text{size}(\phi)) \times \text{count_exist}(\phi)^{\text{count_exist}(\phi)}}
\]

where \( p \) is a polynomial, \( \text{count_exist}(\phi) \) is the number of "\( \exists \)" occurring in \( \phi \).
other reasoning problems on formulas

- **Entailment**: A formula $\phi_2$ is entailed by a formula $\phi_1$, if and only if $(M, w) |= \phi_2$ for every model $M = (W, I)$ and every state $w \in W$ such that $(M, w) |= \phi_1$.

- **Equivalence**: A formula $\phi_2$ is equivalent to a formula $\phi_1$, if and only if both $\phi_1$ entails $\phi_2$ and $\phi_2$ entails $\phi_1$.

- **Evaluation**: A formula $\phi$ is holds on a situation specified by an ABox $A$, if and only if $(M, w) |= \phi$ for every model $M = (W, I)$ and every state $w \in W$ such that $(M, w) |= A$.

All of these reasoning problems can be realized based on the satisfiability problem.
reasoning problems
on concepts

- **Satisfiability**: A concept $C$ is satisfiable if and only if there is a model $M=(W, I)$ and a state $w \in W$ such that $C^{I(w)} \neq \emptyset$.

- **Subsumption**: A concept $C_2$ is subsumed by a concept $C_1$, if and only if $C_2^{I(w)} \subseteq C_1^{I(w)}$ for every model $M=(W, I)$ and every state $w \in W$.

- **Equivalence**: A concept $C_2$ is equivalent to a concept $C_1$, if and only if $C_2^{I(w)} = C_1^{I(w)}$ for every model $M=(W, I)$ and every state $w \in W$.

- **Disjointness**: A concept $C_2$ is disjoint with a concept $C_1$, if and only if $C_2^{I(w)} \cap C_1^{I(w)} = \emptyset$ for every model $M=(W, I)$ and every state $w \in W$.

All of these reasoning problems can be realized based on the satisfiability problem.
reasoning problems
on actions

- **Executability**: An action \( \pi \) is executable on a situation specified by an ABox \( A \), if and only if for every model \( M=(W, I) \) and every state \( w_1 \in W \) such that \( (M, w_1) \models A \): there exists a state \( w_2 \in W \) such that \( (w_1, w_2) \in \pi^I \).

- **Realizability**: An action \( \pi \) is realizable, if and only if there exist a model \( M=(W, I) \) and two states \( w_1, w_2 \in W \) such that \( (w_1, w_2) \in \pi^I \).

- **Projection**: To decide whether a formula \( \phi \) really holds after executing an action \( \pi \) under certain situation specified by an ABox \( A \).
subsumption: An action $\pi_2$ is subsumed by an action $\pi_1$, if and only if $\pi_2^I \subseteq \pi_1^I$ for every model $M=(W, I)$.

Equivalence: An action $\pi_2$ is subsumed by an action $\pi_1$, if and only if $\pi_2^I = \pi_1^I$ for every model $M=(W, I)$.

All of these reasoning problems can be realized based on the satisfiability problem.
reasoning problems on knowledge base

- **Consistency**: A knowledge base $K = \langle T, Ac, A \rangle$ is consistent, if and only if there exist a model $M = (W, I)$ and a state $w \in W$ such that $(M, w) \models K$, i.e., $M \models T$ and $(M, w) \models A$.

- **Entailment**: A knowledge base $K_2$ is entailed by a knowledge base $K_1$, if and only if $(M, w) \models K_2$ for every model $M = (W, I)$ and every state $w \in W$ such that $(M, w) \models K_1$.

- **Equivalence**: A knowledge base $K_2$ is equivalent to a knowledge base $K_1$, if and only if both $K_2$ entails $K_1$ and $K_1$ entails $K_2$.

- **Updating**: To construct a knowledge base $K_2$ that resulted from executing an action $\pi$ on a knowledge base $K_1$. 
DDL Reasoner

Diagram showing the process of reasoning in a DDL system, with components such as Editor, Parser & Translator, and Validation & Species Checking & Repairing. The diagram also includes a DL reasoner with subcomponents like reasoner engine, ABox query engine, tableau rules, regress & ABox update, and reasoning about formula, action, concept, and ontology.
DDL Reasoner

The result is: OR(AND(Phd Candidate(x), NOT(Not hold Degree(x, doctor Degree))))

Reasoning type: Precendence calculation for Action.
The action for reasoning is: PhdDefend(x), graduate(x).
The result is: AND(OR(AND(Phd Candidate(x), NOT(Not hold Degree(x, doctor Degree))))), NOT(AND(NOT(NOT (stu

--- for test: the new concept assertion is:
NOT(AND(OR(student, nominal)), NOT(nominal))(x)

--- for test: the init node is as follows:
Formula set:
NOT(student)(x)
OR()
RoleTable:
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
SWSBroker: Semantic Web Services Platform:

Service Composition (SWSBroker)

Composition Editor

Plans

Pattern Learning

Service execution engine

Intention

Desire

DDL Reasoner

AI Planning

Composition pattern

Acquire

Ontology mapping

Ontology-Based Knowledge base

Context Knowledge

Knowledge

Actions

Ontology Editor

Knowledge Management (KMSphere)
SWSBroker Interface
Outline

- Introduction
- Description Logic
- Dynamic Description Logic
- DDL-Based Knowledge Representation
- DDL-Based Reasoning
- Applications
- Conclusions
Conclusions

- DDL: embracing action formalism into DL
  - atomic actions are described by their preconditions and effects, which are represented over ontologies expressed in DL;
  - complex actions can be constructed with the help of standard action constructors;
  - actions can be used as modal operators to construct concepts and formulas.
- A prefixed tableau calculus was provided, based on which a terminable, sound, and complete satisfiability checking algorithm was designed.
Conclusions

- Typical reasoning problems on formula, concept, action, and knowledge base were studied.
- A DDL reasoner was developed to support all these reasoning problems.
- The DDL reasoner was combined with AI planner in SWSBroker, which supports the description, matching, and composition of semantic web services.
Conclusions

- Contributions of DDL:
  - Provides an approach to extend DL for describing concepts with dynamic meanings;
  - Provides an approach to fill the gap between action formalisms based on first- or high-order logics and those based on propositional logics;
  - Provides an approach to combine the static descriptions of the information provided by ontologies with the dynamic descriptions of the computations provided by web services.
Thank You

Intelligence Science
http://www.intsci.ac.cn/

Enjoy Xi’an

Question!