2nd World Congress and School on

Universal Logic

中科院计算所

INSTITUTE OF COMPUTING TECHNOLOGY

DDL: Embracing Action Formalism into Description Logic

Zhongzhi Shi Liang Chang

shizz@ics.ict.ac.cn

Institute of Computing Technology

Chinese Academy of Sciences

Outline

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

Intelligence

Intelligence encompasses abilities such as:

- *understanding language*
- *perception*
- *learning*
- *reasoning*

The Dream in AI

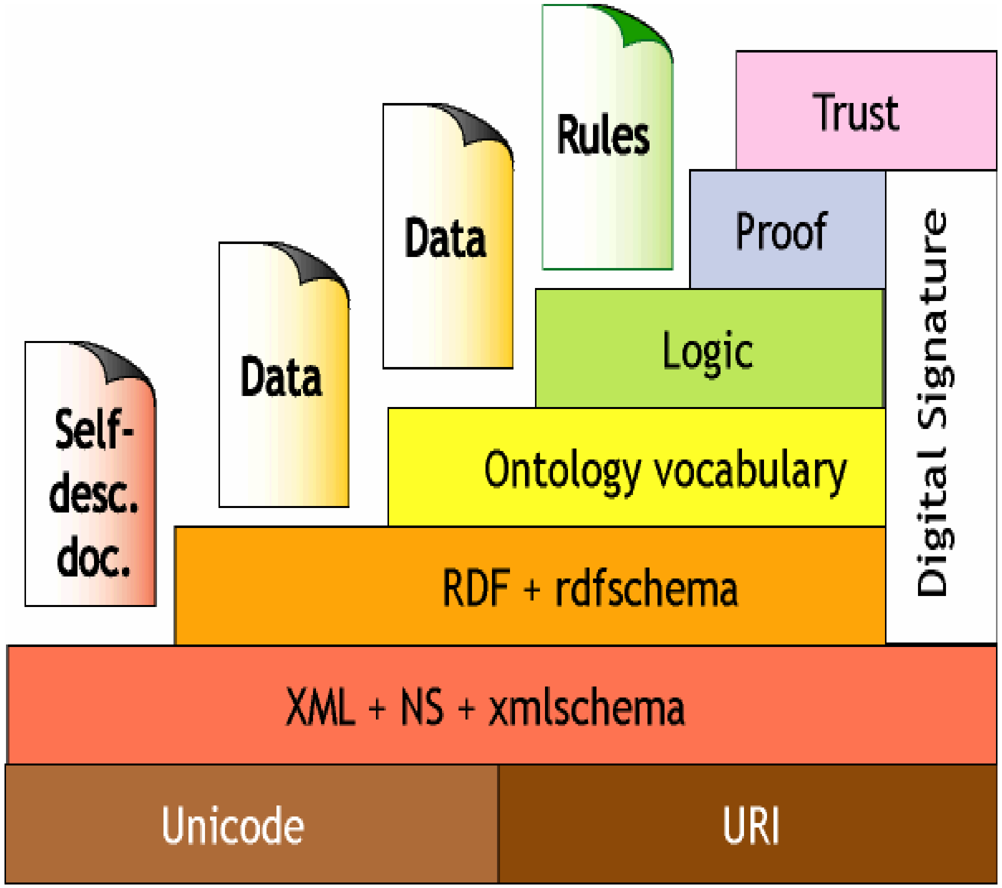
Look for a formal logic can describe:

- *Perception*
- *Reasoning*
- *Behavior*
- *Decision making*

Formal Logics

- First order logic
- Modal logic
- Temporal logic
- Prolog
- GOLOG
- Description logic

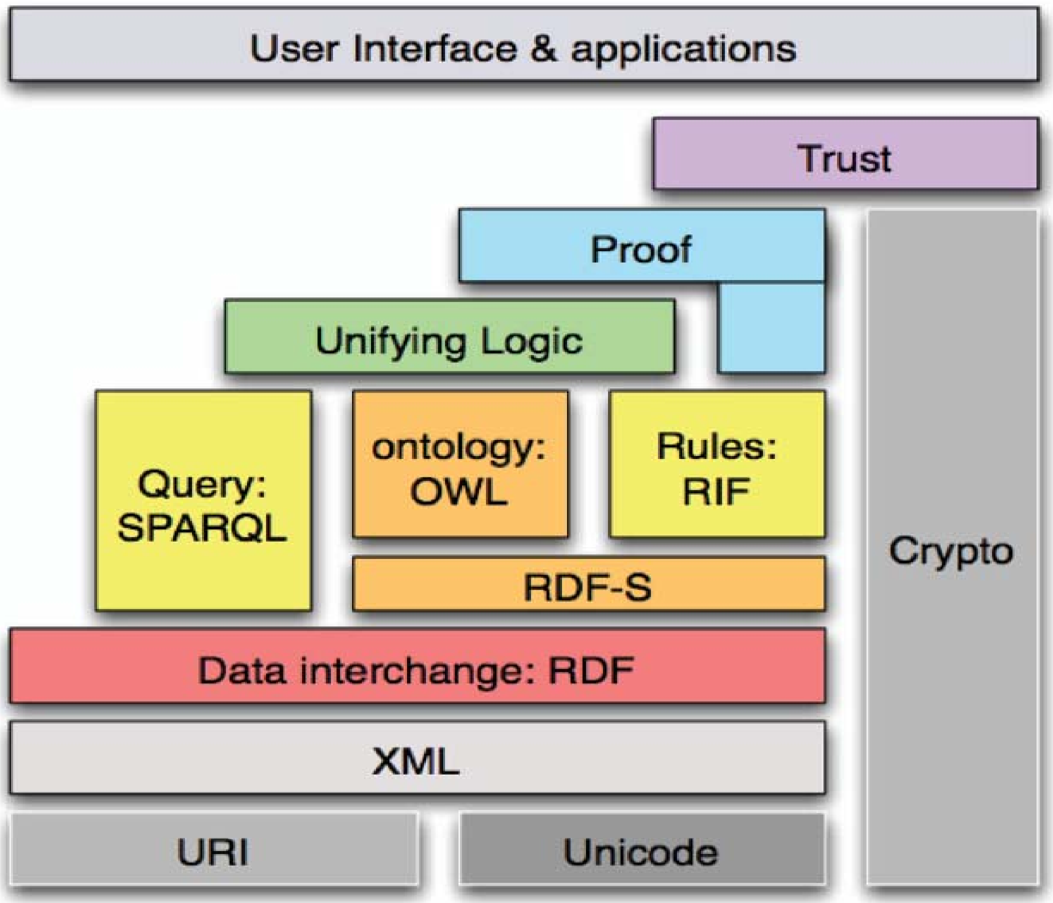
Semantic Web Layers



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

by Tim Berners-Lee

Semantic Web Layers



- 2006

Web Science

by Tim Berners-Lee

Universal Logic

Beziau, Jean-Yves

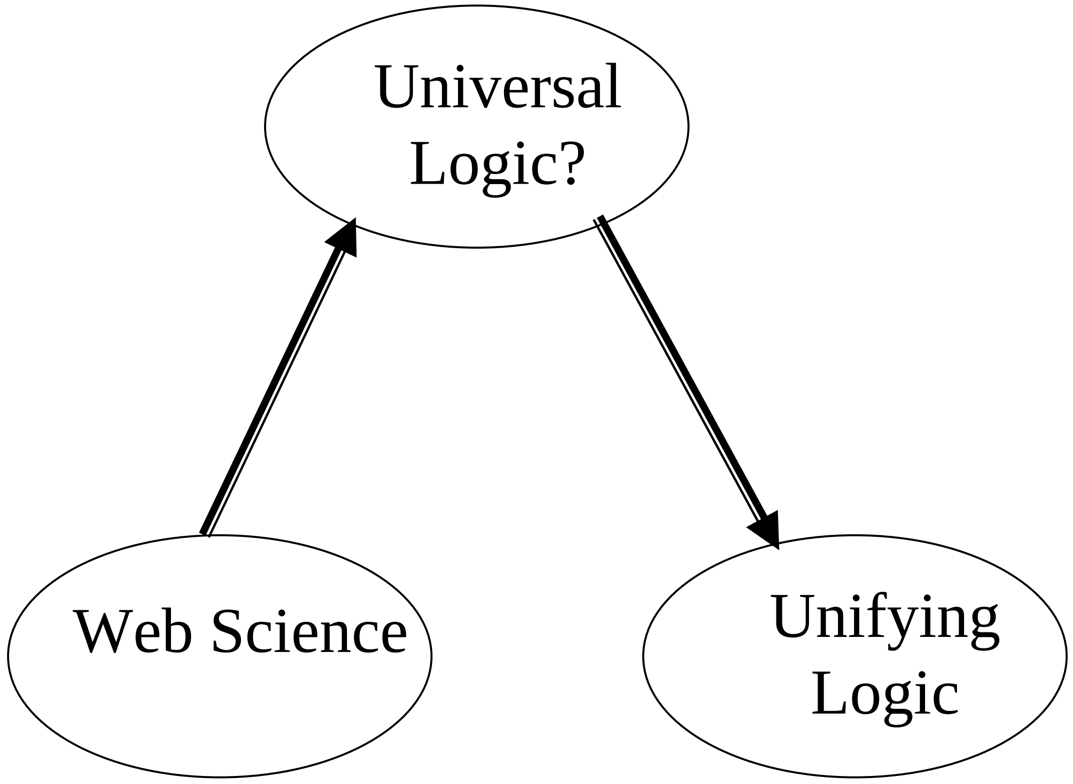
- Universal Logic is not a new logic, but a general theory of logics, considered as mathematical structures.
- The name was introduced about ten years ago, but the subject is as old as the beginning of modern logic: Alfred Tarski and other Polish logicians such as Adolf Lindenbaum developed a general theory of logics at the end of the 1920s based on consequence operations and logical matrices.

Universal Logic

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

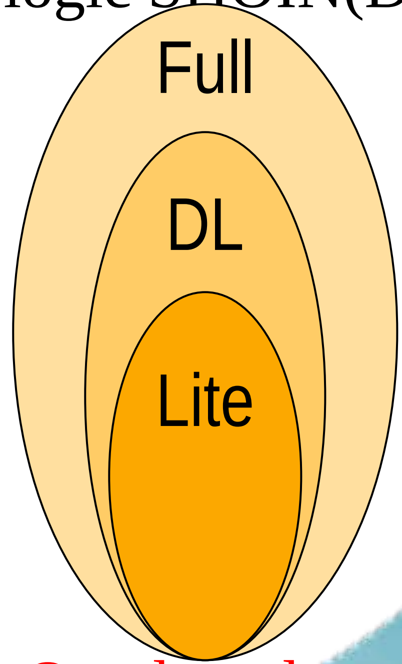


Outline

- 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

Description Logics

- 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, . . .)
  - 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

A Basic Description Logic: ALC

Constructor	Syntax	Semantics	Examples
Atomic concept	A	$A^I \subseteq \Delta^I$	Human
Atomic Relation	R	$R^I \subseteq \Delta^I \times \Delta^I$	has-child
$\sqcap$	$C \sqcap D$	$C^I \cap D^I$	Human $\sqcap$ Male
$\sqcup$	$C \sqcup D$	$C^I \cup D^I$	Doctor $\sqcup$ Lawyer
$\neg$	$\neg C$	$\Delta^I \backslash C$	$\neg$ Male
$\exists$	$\exists R.C$	$\{x \mid \exists y. <x,y> \in R^I \wedge y \in C^I\}$	$\exists$ has-child.Male
$\forall$	$\forall R.C$	$\{x \mid \forall y. <x,y> \in R^I \Rightarrow y \in C^I\}$	$\forall$ has-child.Doctor

Examples

$\text{woman} \equiv \text{person} \sqcap \text{female}$

$\text{man} \equiv \text{person} \sqcap \neg\text{woman}$

$\text{mother} \equiv \text{woman} \sqcap \exists\text{hasChild.person}$

$\text{father} \equiv \text{man} \sqcap \exists\text{hasChild.person}$

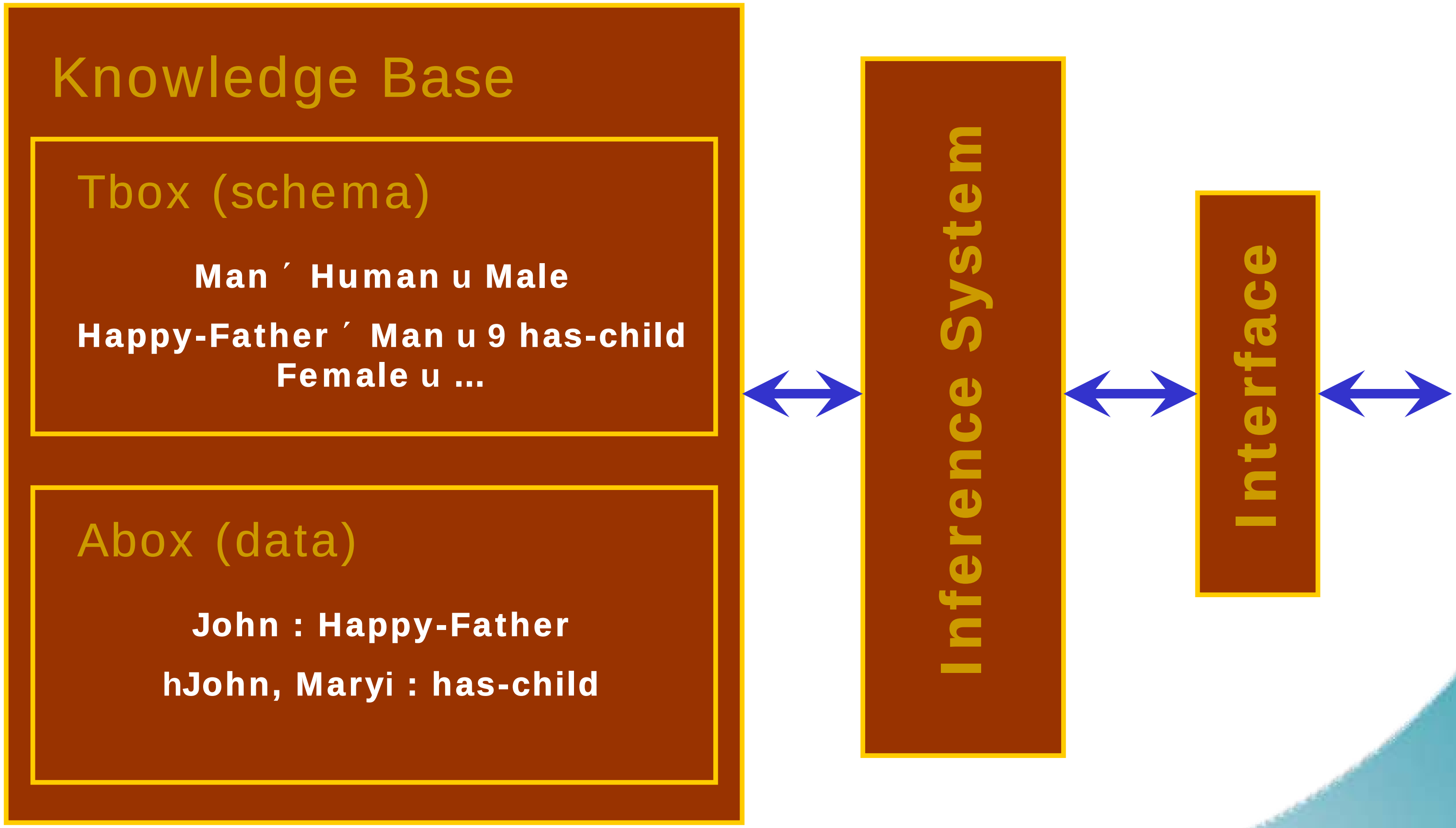
DL Knowledge Base

- 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 $\equiv$ Man $\sqcap$ $\exists$hasChild.Female $\sqcap$ …
    - Elephant $\sqsubseteq$ Animal $\sqcap$ Large $\sqcap$ 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

Name	Rule
$\sqcap$	if $(C_1 \sqcap C_2)(x) \in S$， and $C_1(x) \notin S$ or $C_2(x) \notin S$, Then $S := S \cup \{C_1(x), C_2(x)\}$
$\sqcup$	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 $S := S \cup \{C_2(x)\}$
$\exists$	if $(\exists R.C)(x) \in S$， and not exist y $R(x, y) \in S$、$C(y) \in S$， then S has z, $S := S \cup \{C(z), R(x, z)\}$
$\forall$	if $(\forall R.C)(x) \in S$， tehn $S := S \cup \{C(y) \mid R(x, y) \in S$ and $C(y) \notin S\}$

DL Architecture



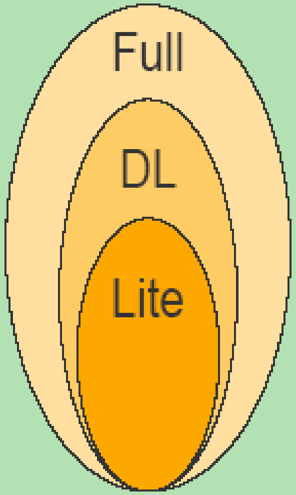
Ontology language: OWL

Language Layers

- **OWL Light**
  - (sub)classes, individuals
  - (sub)properties, domain, range
  - conjunction

  } **RDF Schema**

  - (in)equality
  - cardinality 0/1
  - datatypes
  - inverse, transitive, symmetric
  - hasValue
  - someValuesFrom
  - allValuesFrom



- **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

Dynamic Description Logic

- 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@.

Syntax

- 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:

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

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

- *Abbreviations:*

$$C \sqcap C' = \neg(\neg C \sqcup \neg C')$$
$$\forall R.C = \neg\exists R.C$$
$$\top = C \sqcup \neg C$$
$$\bot = \neg\top$$

Syntax

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

$$\varphi, \varphi' \longrightarrow C(p) | R(p,q) | \neg\varphi | \varphi \vee \varphi' | < \pi > \varphi \qquad (2)$$

where C is a concept, p, $q \in N_I$, $R \in N_R$, and π is an action.

- *Abbreviations:*

$$\begin{aligned} \varphi \wedge \varphi' &= \neg(\neg\varphi \vee \neg\varphi') \\ [\pi]\varphi &= \neg < \pi > \neg\varphi \\ \varphi \rightarrow \varphi' &= \neg\varphi \vee \varphi' \\ true &= \varphi \vee \neg\varphi \\ false &= \neg true \end{aligned}$$

Syntax

- 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 $\phi_1 \wedge \ldots \wedge \phi_n \rightarrow \Phi_k$ for all k with $1 \leqslant k \leqslant m$.

Syntax

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

$$\pi, \pi' \longrightarrow (P, E) | \varphi? | \pi \cup \pi' | \pi ; \pi' | \pi^* \tag{3}$$

where (*P,E*) is an atomic action, ϕ is a formula.

- *Abbreviations:*

$$\text{if } \varphi \text{ then } \pi \text{ else } \pi' = (\varphi?;\pi) \cup ((\neg\varphi)?;\pi')$$
$$\text{while } \varphi \text{ do } \pi = (\varphi?;\pi)^* \ ; (\neg\varphi)?$$

Semantics

- A **model** for dynamic description logic is a pair $M=(W, I)$, where W is a set of states, I associates with each state $w \in W$ an interpretation：

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

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

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

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

Furthermore, each action π is interpreted as a binary relation $\pi^I \subseteq W \times W$.

Semantics

- **Semantics**: Let M=(W, I) 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 $(@_p C)^{I(w)} = \Delta^I$, else $(@_p C)^{I(w)} = \emptyset$;
(3) $(\neg C)^{I(w)} = \Delta^I - C^{I(w)}$;
(4) $(C \sqcup D)^{I(w)} = C^{I(w)} \cup D^{I(w)}$;
(5) $(\exists R.C)^{I(w)} = \{x | \exists y.((x,y) \in R^{I(w)} \wedge y \in C^{I(w)})\}$;
(6) $(<\pi> C)^{I(w)} = \{p | \exists w' \in W.((w,w') \in \pi^I \wedge p \in C^{I(w')})\}$;

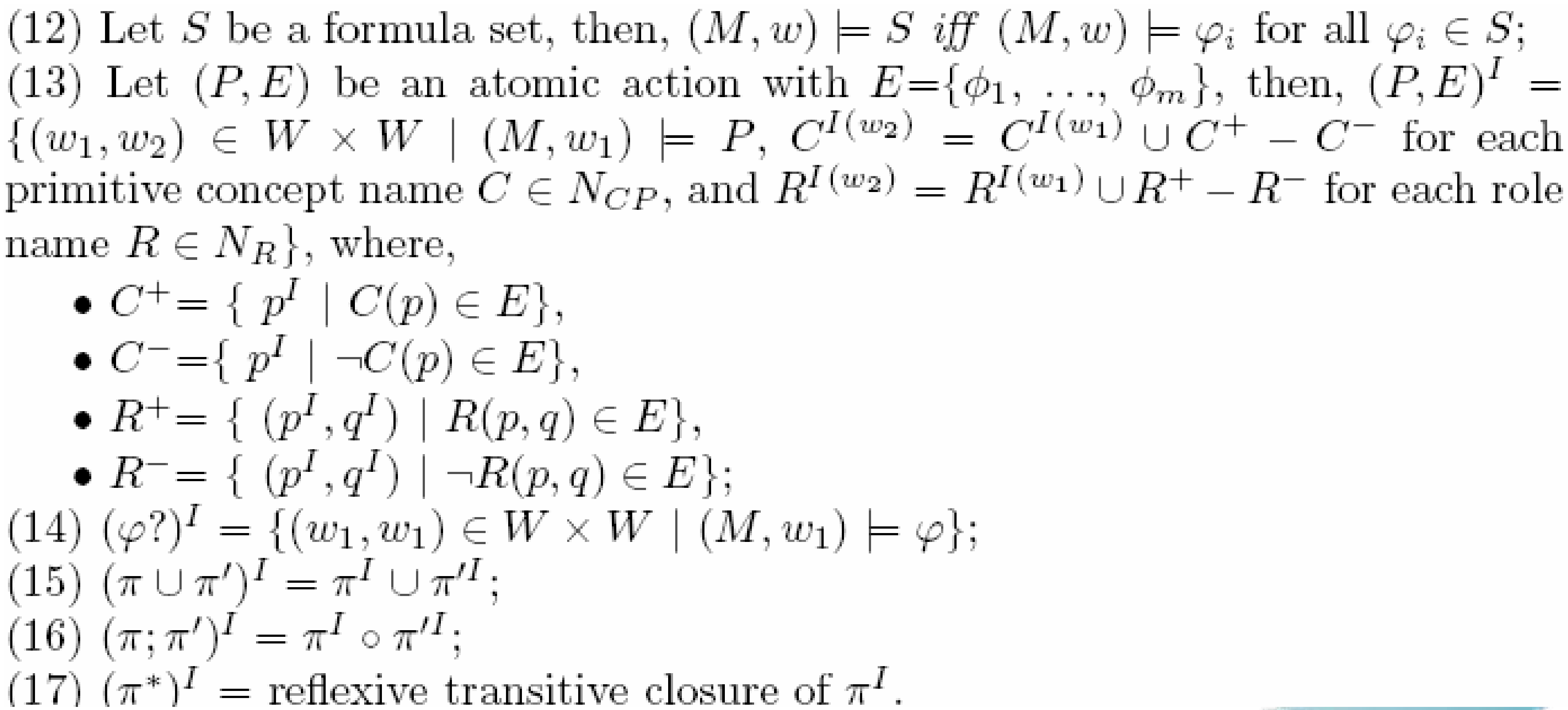
Semantics

- the truth-relation $(M,w) \models \phi$ (or simply $w \models \phi$ if M is understood) for a formula ϕ is defined inductively as:

(7) $(M,w) \models C(p)$ iff $p^I \in C^{I(w)}$;
(8) $(M,w) \models R(p,q)$ iff $(p^I, q^I) \in R^{I(w)}$;
(9) $(M,w) \models \neg\varphi$ iff $(M,w) \models \varphi$ not holds;
(10) $(M,w) \models \varphi \vee \psi$ iff $(M,w) \models \varphi$ or $(M,w) \models \psi$;
(11) $(M,w) \models < \pi > \varphi$ iff $\exists w' \in W.((w,w') \in \pi^I \wedge (M,w') \models \varphi)$;

Semantics

- the binary relation π^I for an action π *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, $(P, E)^I = \{(w_1, w_2) \in W \times W \mid (M, w_1) \models P,\ C^{I(w_2)} = C^{I(w_1)} \cup C^+ - C^-$ for each primitive concept name $C \in N_{CP}$, and $R^{I(w_2)} = R^{I(w_1)} \cup R^+ - R^-$ 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) $(\pi^*)^I$ = reflexive transitive closure of π^I.

Outline

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

Knowledge Base Based on Dynamic Description Logic

- **Knowledge Base:** $K = <T, Ac, A>$

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.,

$$\text{PhdCandidate} = \text{student} \sqcap \exists \text{forDegree.doctorDegree}$$

$$\sqcap <\text{PhdDefend}>\text{doctor}$$

Knowledge Base Based on Dynamic Description Logic

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 α is a new defined action name, $v_1, \ldots, v_n$ are individuals that the action operate on, and π 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)}, {¬student(v)}) ;

buyBicycle(u,v) = ({bicycle(v), ¬owns(u,v), wants(u,v), hasMoney(u)}, {owns(u,v)});

Knowledge Base Based on Dynamic Description Logic

③ *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 $K = <T, Ac, A>$ 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 $(M,w) \models A$.

Outline

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

prefixed tableau calculus

- This prefixed tableau calculus is an elaborated combination of:
  - the standard tableau calculus for *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.

prefixed tableau calculus

Some notations:

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

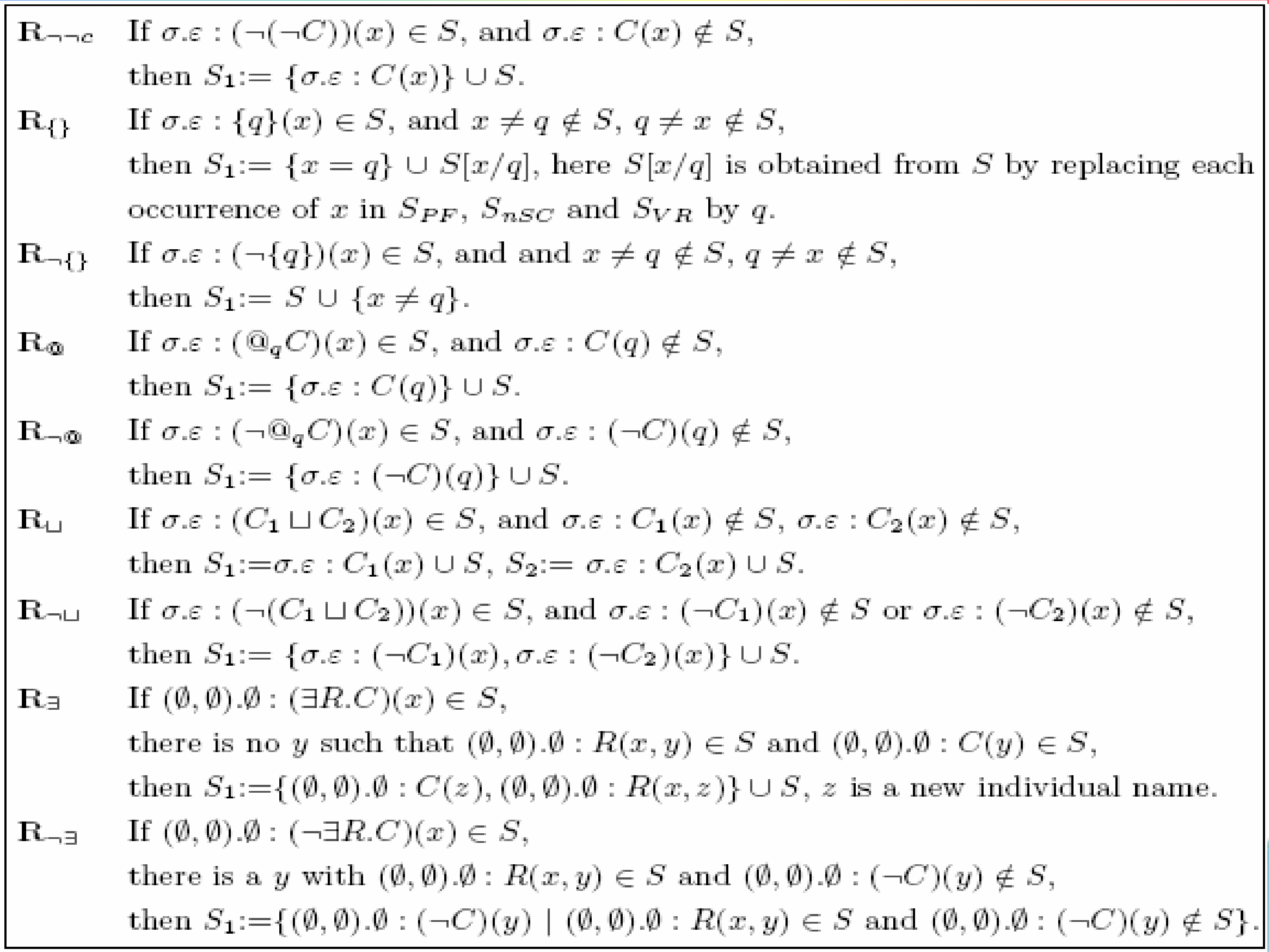
$$\sigma.\varepsilon \longrightarrow (\emptyset,\emptyset).\emptyset | \sigma;(P,E).(\varepsilon - \{\neg\varphi | \varphi \in E\}) \cup E \qquad (4)$$

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

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

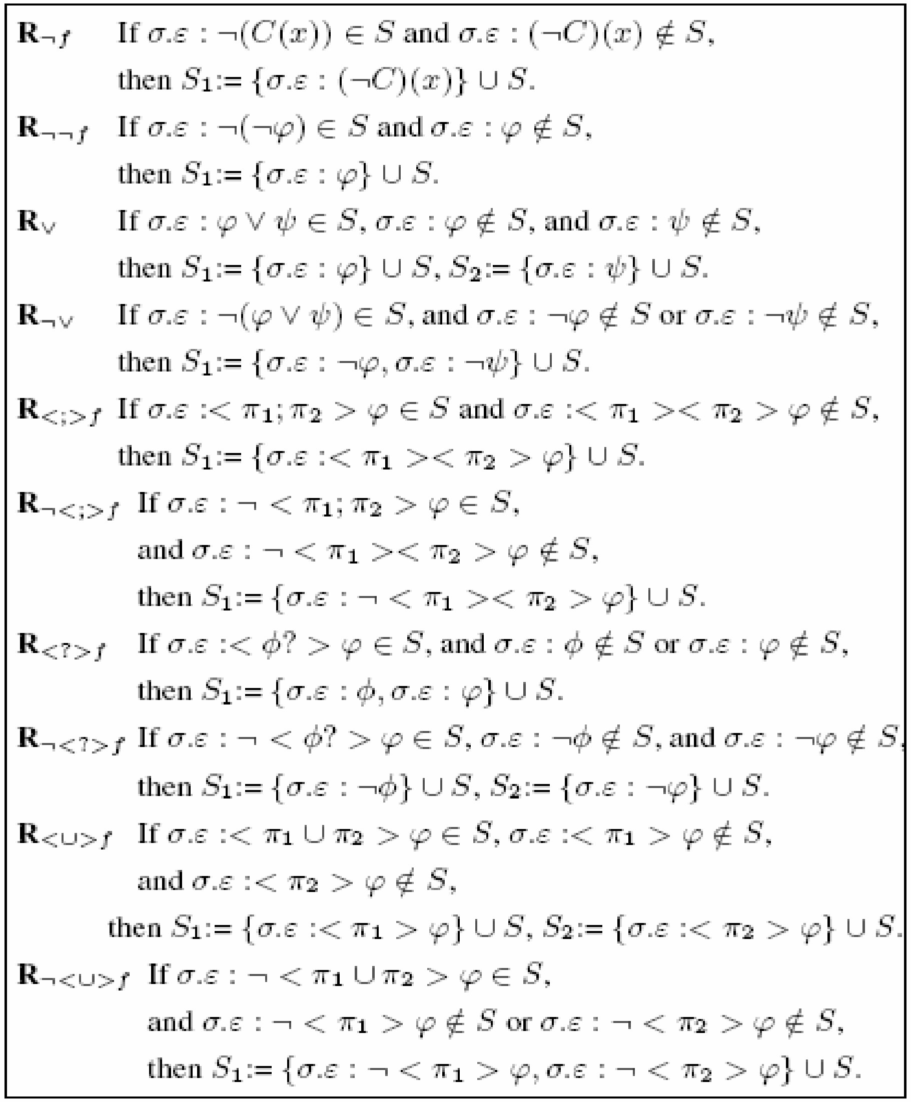
prefixed tableau calculus



$\mathbf{R}_{\neg\neg c}$	If $\sigma.\varepsilon : (\neg(\neg C))(x) \in S$, and $\sigma.\varepsilon : C(x) \notin S$, then $S_1 := \{\sigma.\varepsilon : C(x)\} \cup S$.
$\mathbf{R}_{\{\}}$	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.
$\mathbf{R}_{\neg\{\}}$	If $\sigma.\varepsilon : (\neg\{q\})(x) \in S$, and and $x \neq q \notin S$, $q \neq x \notin S$, then $S_1 := S \cup \{x \neq q\}$.
$\mathbf{R}_{@}$	If $\sigma.\varepsilon : (@_q C)(x) \in S$, and $\sigma.\varepsilon : C(q) \notin S$, then $S_1 := \{\sigma.\varepsilon : C(q)\} \cup S$.
$\mathbf{R}_{\neg @}$	If $\sigma.\varepsilon : (\neg @_q C)(x) \in S$, and $\sigma.\varepsilon : (\neg C)(q) \notin S$, then $S_1 := \{\sigma.\varepsilon : (\neg C)(q)\} \cup S$.
$\mathbf{R}_{\sqcup}$	If $\sigma.\varepsilon : (C_1 \sqcup C_2)(x) \in S$, and $\sigma.\varepsilon : C_1(x) \notin S$, $\sigma.\varepsilon : C_2(x) \notin S$, then $S_1 := \sigma.\varepsilon : C_1(x) \cup S$, $S_2 := \sigma.\varepsilon : C_2(x) \cup S$.
$\mathbf{R}_{\neg\sqcup}$	If $\sigma.\varepsilon : (\neg(C_1 \sqcup C_2))(x) \in S$, and $\sigma.\varepsilon : (\neg C_1)(x) \notin S$ or $\sigma.\varepsilon : (\neg C_2)(x) \notin S$, then $S_1 := \{\sigma.\varepsilon : (\neg C_1)(x), \sigma.\varepsilon : (\neg C_2)(x)\} \cup S$.
$\mathbf{R}_{\exists}$	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.
$\mathbf{R}_{\neg\exists}$	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 $(\emptyset,\emptyset).\emptyset : (\neg C)(y) \notin S\}$.

Figure 1: Rules for concepts

prefixed tableau calculus



$\mathbf{R}_{\neg f}$ If $\sigma.\varepsilon : \neg(C(x)) \in S$ and $\sigma.\varepsilon : (\neg C)(x) \notin S$,
then $S_1 := \{\sigma.\varepsilon : (\neg C)(x)\} \cup S$.

$\mathbf{R}_{\neg\neg f}$ If $\sigma.\varepsilon : \neg(\neg\varphi) \in S$ and $\sigma.\varepsilon : \varphi \notin S$,
then $S_1 := \{\sigma.\varepsilon : \varphi\} \cup S$.

$\mathbf{R}_{\vee}$ If $\sigma.\varepsilon : \varphi \vee \psi \in S$, $\sigma.\varepsilon : \varphi \notin S$, and $\sigma.\varepsilon : \psi \notin S$,
then $S_1 := \{\sigma.\varepsilon : \varphi\} \cup S$, $S_2 := \{\sigma.\varepsilon : \psi\} \cup S$.

$\mathbf{R}_{\neg\vee}$ If $\sigma.\varepsilon : \neg(\varphi \vee \psi) \in S$, and $\sigma.\varepsilon : \neg\varphi \notin S$ or $\sigma.\varepsilon : \neg\psi \notin S$,
then $S_1 := \{\sigma.\varepsilon : \neg\varphi, \sigma.\varepsilon : \neg\psi\} \cup S$.

$\mathbf{R}_{<;>f}$ If $\sigma.\varepsilon :< \pi_1; \pi_2 > \varphi \in S$ and $\sigma.\varepsilon :< \pi_1 >< \pi_2 > \varphi \notin S$,
then $S_1 := \{\sigma.\varepsilon :< \pi_1 >< \pi_2 > \varphi\} \cup S$.

$\mathbf{R}_{\neg<;>f}$ If $\sigma.\varepsilon : \neg < \pi_1; \pi_2 > \varphi \in S$,
and $\sigma.\varepsilon : \neg < \pi_1 >< \pi_2 > \varphi \notin S$,
then $S_1 := \{\sigma.\varepsilon : \neg < \pi_1 >< \pi_2 > \varphi\} \cup S$.

$\mathbf{R}_{<?>f}$ If $\sigma.\varepsilon :< \phi? > \varphi \in S$, and $\sigma.\varepsilon : \phi \notin S$ or $\sigma.\varepsilon : \varphi \notin S$,
then $S_1 := \{\sigma.\varepsilon : \phi, \sigma.\varepsilon : \varphi\} \cup S$.

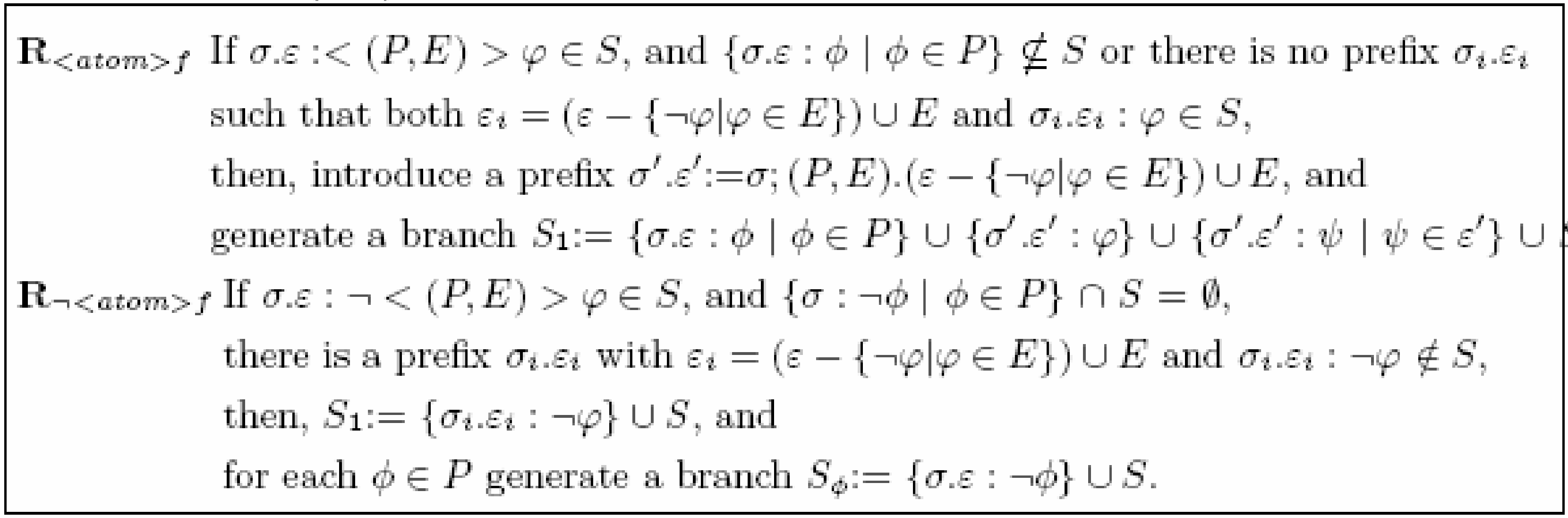
$\mathbf{R}_{\neg<?>f}$ If $\sigma.\varepsilon : \neg < \phi? > \varphi \in S$, $\sigma.\varepsilon : \neg\phi \notin S$, and $\sigma.\varepsilon : \neg\varphi \notin S$
then $S_1 := \{\sigma.\varepsilon : \neg\phi\} \cup S$, $S_2 := \{\sigma.\varepsilon : \neg\varphi\} \cup S$.

$\mathbf{R}_{<\cup>f}$ If $\sigma.\varepsilon :< \pi_1 \cup \pi_2 > \varphi \in S$, $\sigma.\varepsilon :< \pi_1 > \varphi \notin S$,
and $\sigma.\varepsilon :< \pi_2 > \varphi \notin S$,
then $S_1 := \{\sigma.\varepsilon :< \pi_1 > \varphi\} \cup S$, $S_2 := \{\sigma.\varepsilon :< \pi_2 > \varphi\} \cup S$.

$\mathbf{R}_{\neg<\cup>f}$ If $\sigma.\varepsilon : \neg < \pi_1 \cup \pi_2 > \varphi \in S$,
and $\sigma.\varepsilon : \neg < \pi_1 > \varphi \notin S$ or $\sigma.\varepsilon : \neg < \pi_2 > \varphi \notin S$,
then $S_1 := \{\sigma.\varepsilon : \neg < \pi_1 > \varphi, \sigma.\varepsilon : \neg < \pi_2 > \varphi\} \cup S$.

Figure 2: Rules for formulas

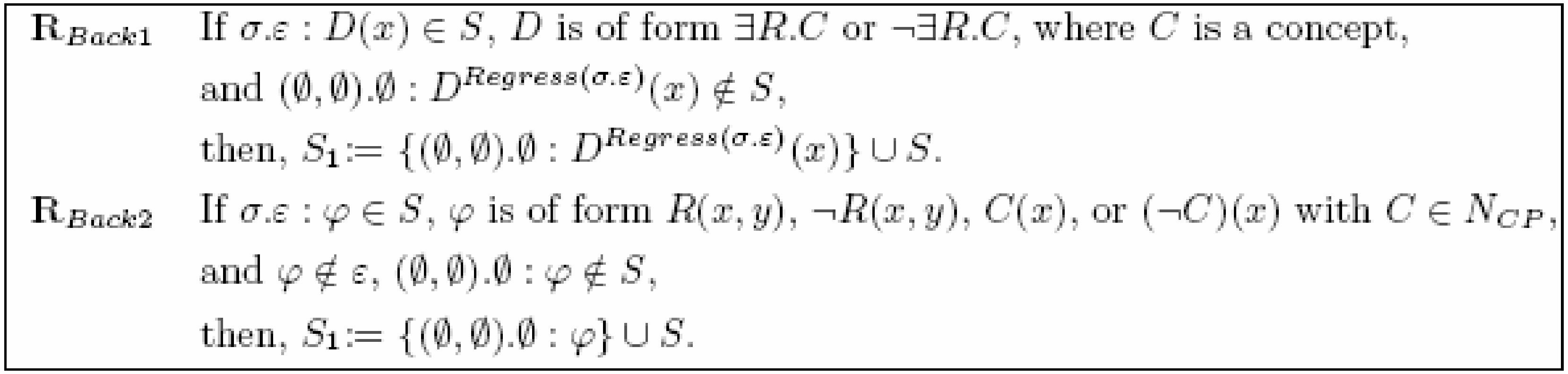
prefixed tableau calculus



$\mathbf{R}_{<atom>f}$ If $\sigma.\varepsilon :< (P,E) > \varphi \in S$, and $\{\sigma.\varepsilon : \phi \mid \phi \in P\} \nsubseteq S$ 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 $S_1 := \{\sigma.\varepsilon : \phi \mid \phi \in P\} \cup \{\sigma'.\varepsilon' : \varphi\} \cup \{\sigma'.\varepsilon' : \psi \mid \psi \in \varepsilon'\} \cup S$.

$\mathbf{R}_{\neg<atom>f}$ If $\sigma.\varepsilon : \neg < (P,E) > \varphi \in S$, and $\{\sigma : \neg\phi \mid \phi \in P\} \cap S = \emptyset$, 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



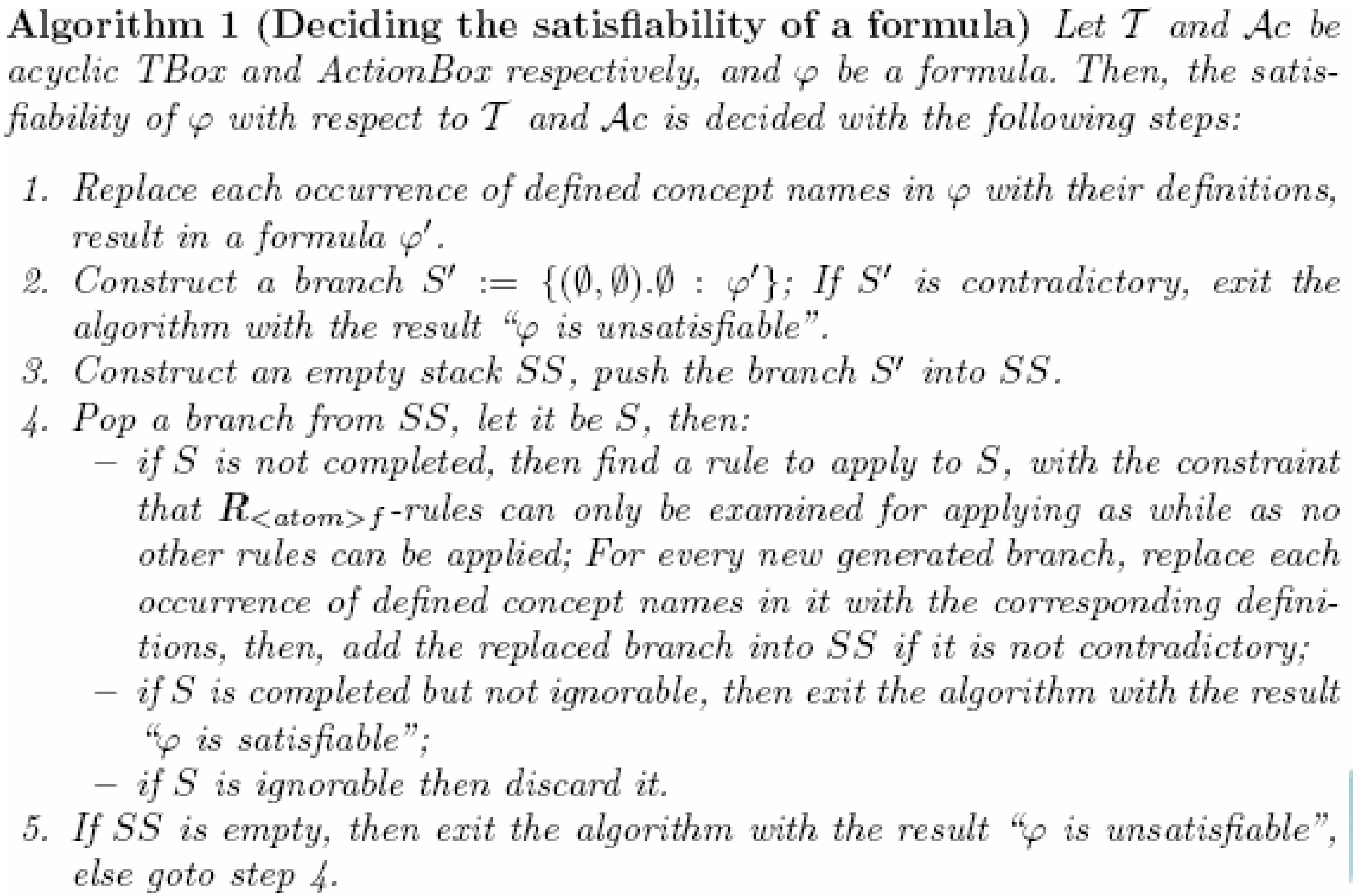
$\mathbf{R}_{Back1}$ If $\sigma.\varepsilon : D(x) \in S$, D is of form $\exists R.C$ or $\neg\exists R.C$, where C is a concept, and $(\emptyset,\emptyset).\emptyset : D^{Regress(\sigma.\varepsilon)}(x) \notin S$, then, $S_1 := \{(\emptyset,\emptyset).\emptyset : D^{Regress(\sigma.\varepsilon)}(x)\}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

$\mathbf{R}_{Back2}$ If $\sigma.\varepsilon : \varphi \in S$, φ 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

satisfiability problem

- A formula ϕ 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 $\mathcal{T}$ and $\mathcal{A}c$ be acyclic TBox and ActionBox respectively, and φ be a formula. Then, the satisfiability of φ with respect to $\mathcal{T}$ and $\mathcal{A}c$ is decided with the following steps:*

1. *Replace each occurrence of defined concept names in φ with their definitions, result in a formula φ'.*
2. *Construct a branch $S' := \{(\emptyset,\emptyset).\emptyset : \varphi'\}$; If S' is contradictory, exit the algorithm with the result "φ 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 $\mathbf{R}_{<atom>^f}$-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 "φ is satisfiable";*
   - *if S is ignorable then discard it.*
5. *If SS is empty, then exit the algorithm with the result "φ is unsatisfiable", else goto step 4.*

satisfiability problem

- *Theorem* 1: The satisfiability-checking algorithm is terminable, sound, and complete.

- *Theorem* 2: Let ϕ 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 size(M) $\leqslant 2^{p(\text{size}(\phi))} \times$ count_exist(ϕ)$^{\text{count\_exist}(\phi)}$, where p is a polynomial, count_exist(ϕ) is the number of "$\exists$" occurring in ϕ.

other reasoning problems on formulas

- **Entailment**： A formula ϕ_2 is entailed by a formula ϕ_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 ϕ_2 is equivalent to a formula ϕ_1, if and only if both ϕ_1 entails ϕ_2 and ϕ_2 entails ϕ_1.
- **Evaluation**： A formula ϕ 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 $C^{I(w)}\neq \varnothing$.
- **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 $C_2^{I(w)} \cap C_1^{I(w)} = \varnothing$ 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 π 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 π 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 ϕ really holds after executing an action π under certain situation specified by an ABox *A*.

reasoning problems on actions

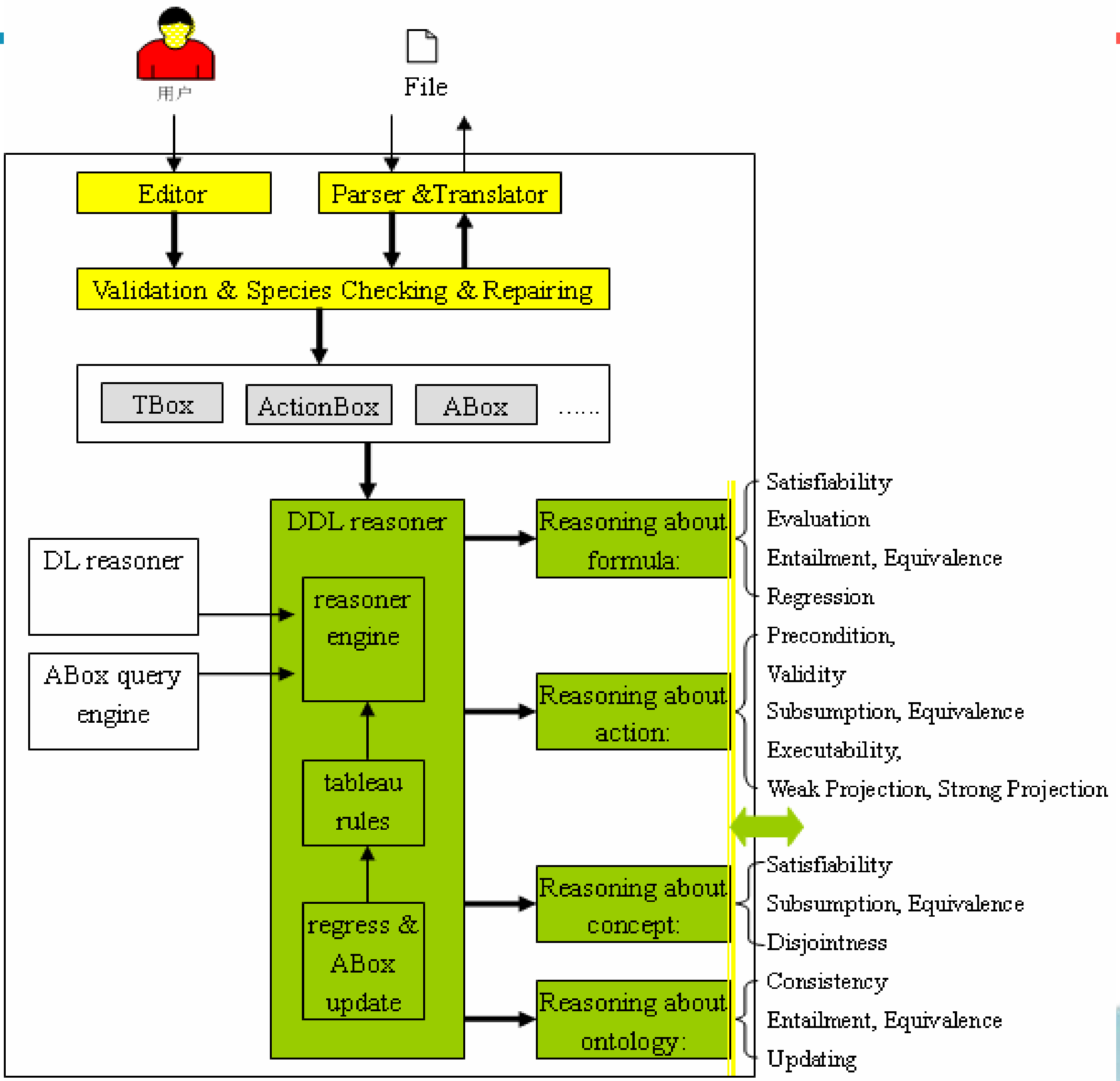
- **Subsumption**: An action π_2 is subsumed by an action π_1, if and only if $\pi_2^I \subseteq \pi_1^I$ for every model M=(W, I).
- **Equivalence:** An action π_2 is subsumed by an action π_1, if and only if $\pi_2^I = \pi_1^I$ for every model M=(W, I).

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

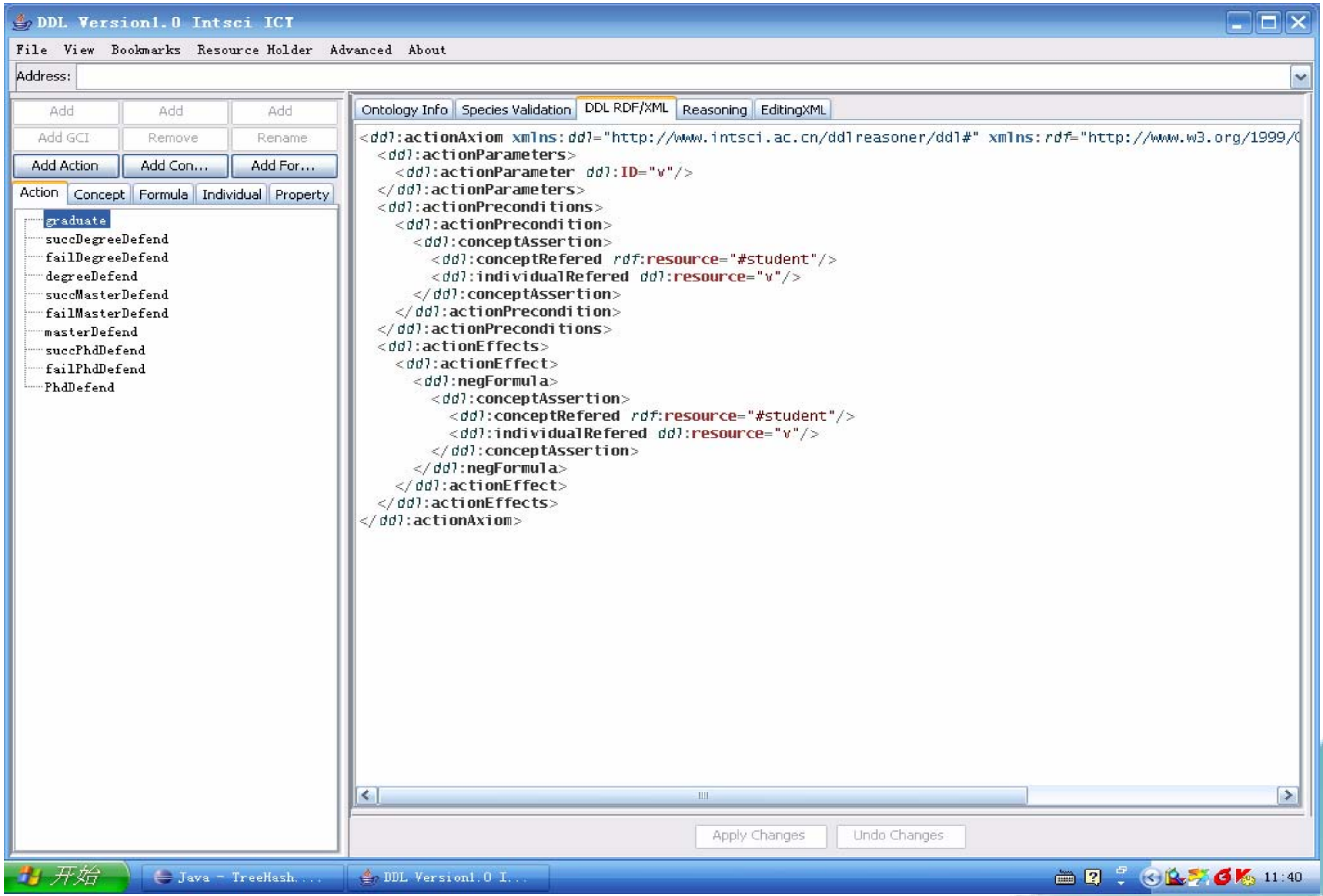
reasoning problems on knowledge base

- **Consistency**: A knowledge base K = <T, Ac, A> 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 π on a knowledge base K_1.

DDL Reasoner

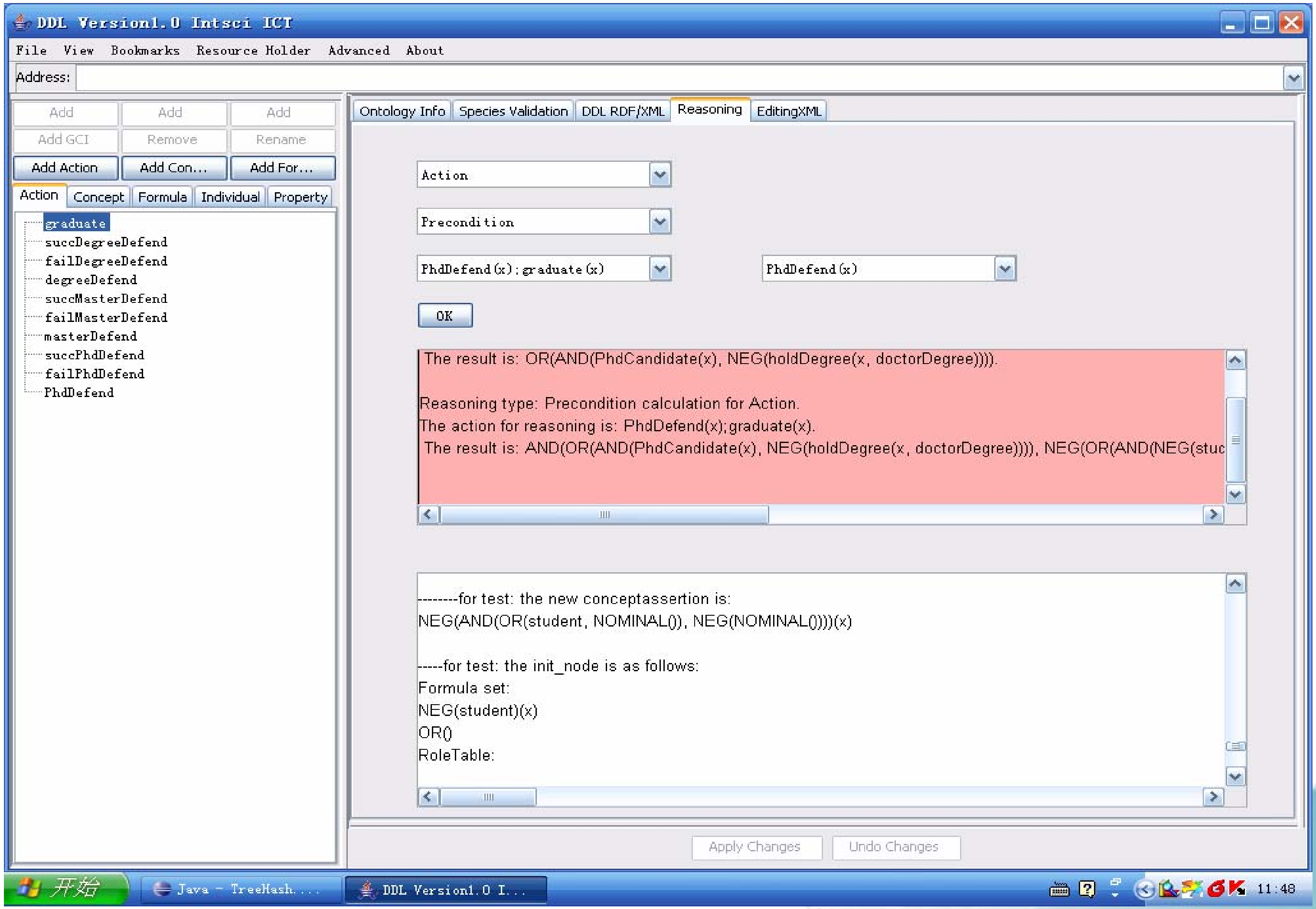


DDL Reasoner


DDL Version1.0 Intsci ICT
File View Bookmarks Resource Holder Advanced About
Address:
Add Add Add
Add GCI Remove Rename
Add Action Add Con... Add For...
Action Concept Formula Individual Property
graduate
succDegreeDefend
failDegreeDefend
degreeDefend
succMasterDefend
failMasterDefend
masterDefend
succPhdDefend
failPhdDefend
PhdDefend
Ontology Info Species Validation DDL RDF/XML Reasoning EditingXML
Apply Changes Undo Changes

```
<ddl:actionAxiom xmlns:ddl="http://www.intsci.ac.cn/ddlreasoner/ddl#" xmlns:rdf="http://www.w3.org/1999/(
  <ddl:actionParameters>
    <ddl:actionParameter ddl:ID="v"/>
  </ddl:actionParameters>
  <ddl:actionPreconditions>
    <ddl:actionPrecondition>
      <ddl:conceptAssertion>
        <ddl:conceptRefered rdf:resource="#student"/>
        <ddl:individualRefered ddl:resource="v"/>
      </ddl:conceptAssertion>
    </ddl:actionPrecondition>
  </ddl:actionPreconditions>
  <ddl:actionEffects>
    <ddl:actionEffect>
      <ddl:negFormula>
        <ddl:conceptAssertion>
          <ddl:conceptRefered rdf:resource="#student"/>
          <ddl:individualRefered ddl:resource="v"/>
        </ddl:conceptAssertion>
      </ddl:negFormula>
    </ddl:actionEffect>
  </ddl:actionEffects>
</ddl:actionAxiom>
```

DDL Reasoner


DDL Version1.0 Intsci ICT
File View Bookmarks Resource Holder Advanced About
Address:
Add Add Add
Add GCI Remove Rename
Add Action Add Con... Add For...
Action Concept Formula Individual Property
graduate
succDegreeDefend
failDegreeDefend
degreeDefend
succMasterDefend
failMasterDefend
masterDefend
succPhdDefend
failPhdDefend
PhdDefend
Ontology Info Species Validation DDL RDF/XML Reasoning EditingXML
Action
Precondition
PhdDefend(x);graduate(x)
PhdDefend(x)
OK
Apply Changes Undo Changes
开始 Java - TreeHash.... DDL Version1.0 I... 11:48

```
The result is: OR(AND(PhdCandidate(x), NEG(holdDegree(x, doctorDegree)))).

Reasoning type: Precondition calculation for Action.
The action for reasoning is: PhdDefend(x);graduate(x).
 The result is: AND(OR(AND(PhdCandidate(x), NEG(holdDegree(x, doctorDegree)))), NEG(OR(AND(NEG(stuc
```

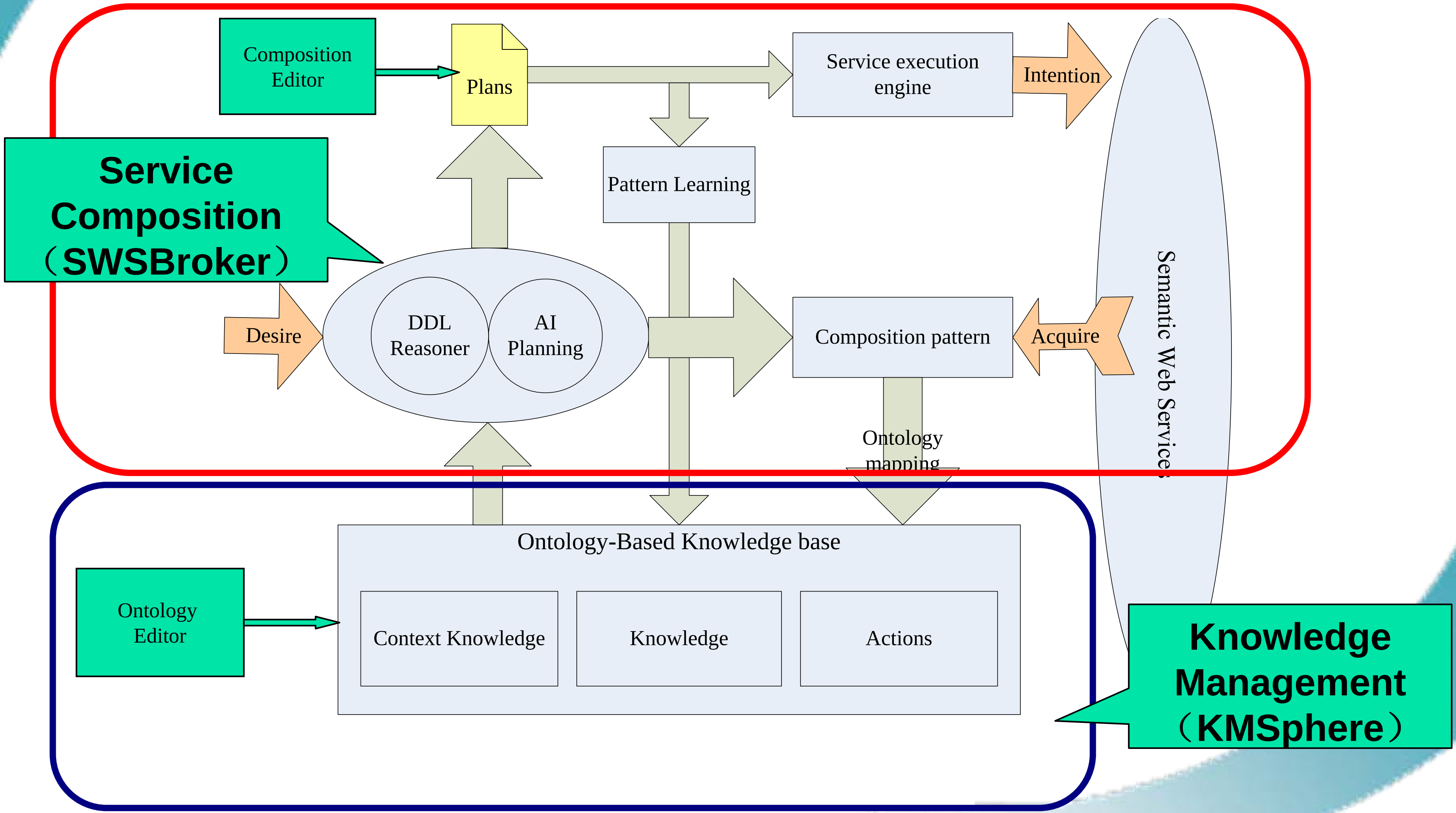
```
--------for test: the new conceptassertion is:
NEG(AND(OR(student, NOMINAL()), NEG(NOMINAL())))(x)

-----for test: the init_node is as follows:
Formula set:
NEG(student)(x)
OR()
RoleTable:
```

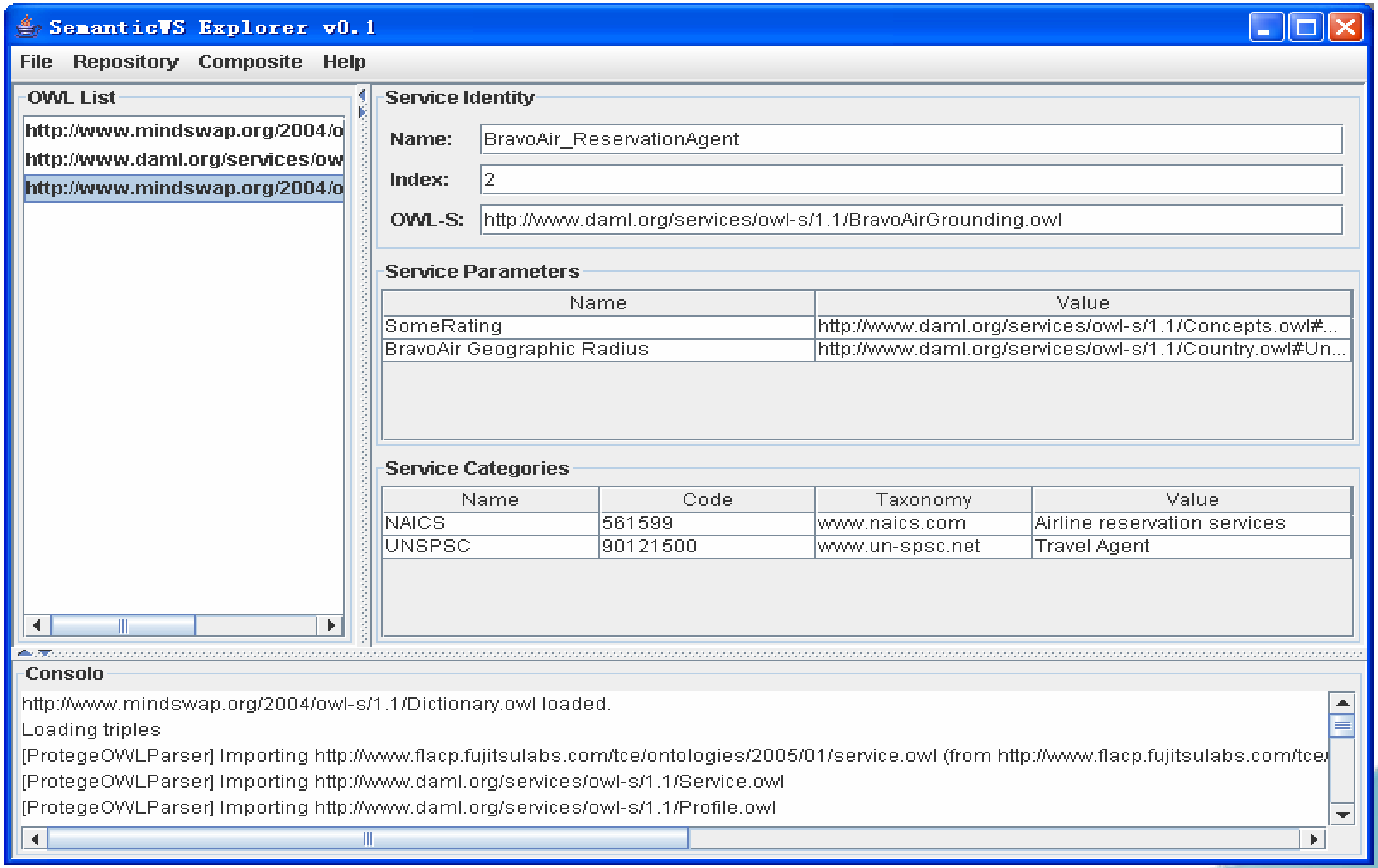
Outline

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

SWSBroker: Semantic Web Services Platform:



SWSBroker Interface



OWL List:
http://www.mindswap.org/2004/o
http://www.daml.org/services/ow
http://www.mindswap.org/2004/o

Service Identity

Name:	BravoAir_ReservationAgent
Index:	2
OWL-S:	http://www.daml.org/services/owl-s/1.1/BravoAirGrounding.owl

Service Parameters

Name	Value
SomeRating	http://www.daml.org/services/owl-s/1.1/Concepts.owl#...
BravoAir Geographic Radius	http://www.daml.org/services/owl-s/1.1/Country.owl#Un...

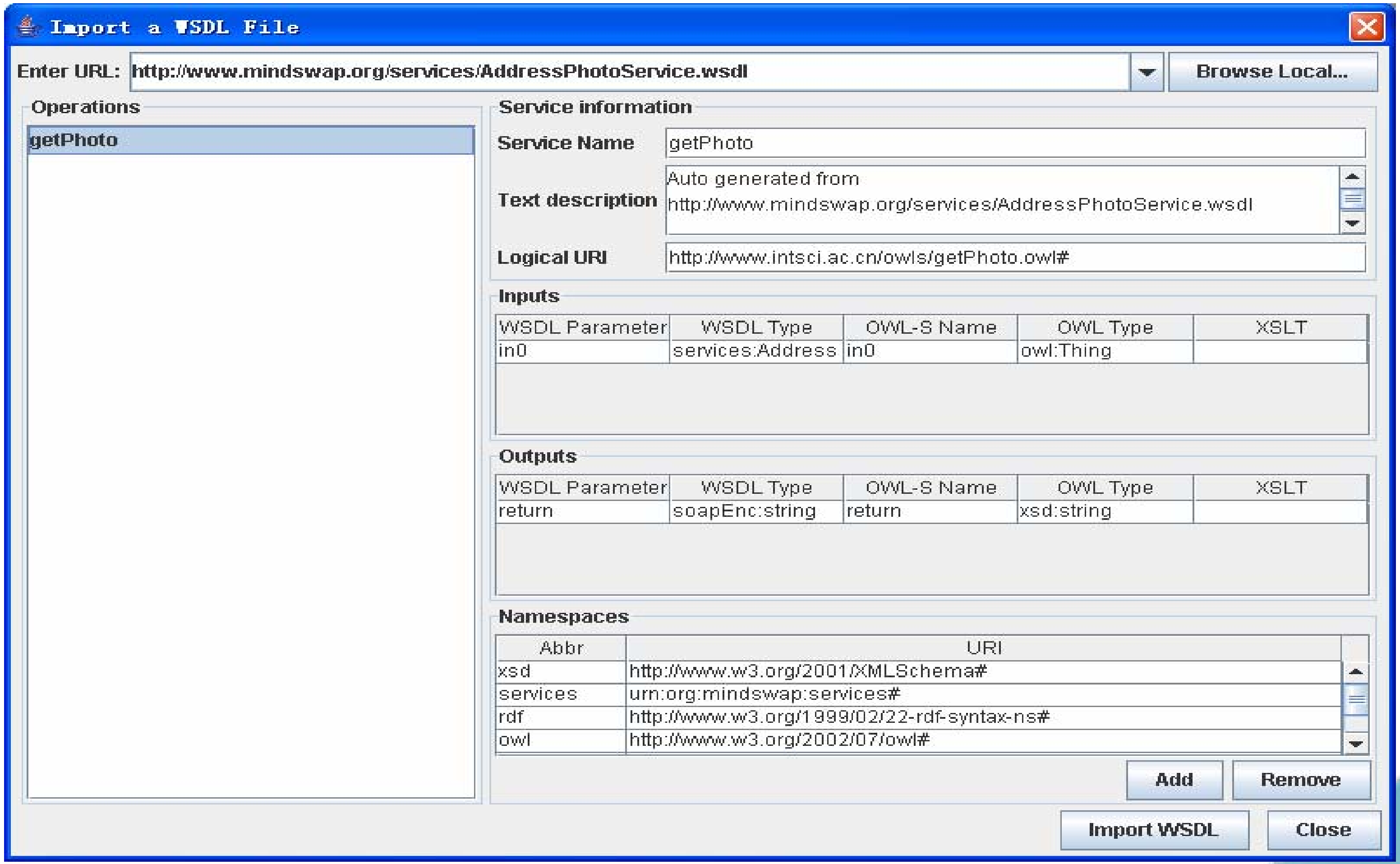
Service Categories

Name	Code	Taxonomy	Value
NAICS	561599	www.naics.com	Airline reservation services
UNSPSC	90121500	www.un-spsc.net	Travel Agent

Consolo

```
http://www.mindswap.org/2004/owl-s/1.1/Dictionary.owl loaded.
Loading triples
[ProtegeOWLParser] Importing http://www.flacp.fujitsulabs.com/tce/ontologies/2005/01/service.owl (from http://www.flacp.fujitsulabs.com/tce
[ProtegeOWLParser] Importing http://www.daml.org/services/owl-s/1.1/Service.owl
[ProtegeOWLParser] Importing http://www.daml.org/services/owl-s/1.1/Profile.owl
```

SWDL To OWL-S



Import a WSDL File

Enter URL: http://www.mindswap.org/services/AddressPhotoService.wsdl

Operations

getPhoto

Service information

Service Name: getPhoto

Text description: Auto generated from http://www.mindswap.org/services/AddressPhotoService.wsdl

Logical URI: http://www.intsci.ac.cn/owls/getPhoto.owl#

Inputs

WSDL Parameter	WSDL Type	OWL-S Name	OWL Type	XSLT
in0	services:Address	in0	owl:Thing	

Outputs

WSDL Parameter	WSDL Type	OWL-S Name	OWL Type	XSLT
return	soapEnc:string	return	xsd:string	

Namespaces

Abbr	URI
xsd	http://www.w3.org/2001/XMLSchema#
services	urn:org:mindswap:services#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
owl	http://www.w3.org/2002/07/owl#

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

Enjoy Xi’an

Intelligence Science

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

Question!