$\mathcal{ALC}$-LTL Formula Generator

M.Fareed Arif

TU Dresden

May 18, 2011

Outline

Outline

Description Logic $\mathcal{ALC}$

$\mathcal{ALC}$ [Schmidt-Schau 1991] is a DL with conjunction ($\sqcap$), disjunction($\sqcup$), negation($\neg$), existential restriction ($\exists$) and **value restriction ($\forall$)**.

General Concept Inclusion (GCI):

A general concept inclusion (GCI) is of form $C \sqsubseteq D$ where C, D are $\mathcal{ALC}$-concept descriptions.

Assertion:

An assertion is of the form $a : C$ or $(a, b) : r$ where C is an $\mathcal{ALC}$-concept description, $r \in N_C$ and $a, b \in N_I$.

Example:

$$Person \sqcap \exists hasChild.Person$$

$$GermanCitizen \sqsubseteq \exists insured\_by.HealthInsurance$$

$$Germany : \exists winner.FIFA\_WORLD\_CUP$$

Temporalized DL $\mathcal{ALC}$-LTL

$\mathcal{ALC}$-LTL ([Franz Baader 2008]) is a temporalized extension of a logic-based knowledge representation formalism $\mathcal{ALC}$.

- If α is an $\mathcal{ALC}$-axiom (i.e., Both GCIs and Assertions are called $\mathcal{ALC}$-axioms), then α is an $\mathcal{ALC}$-LTL formula;
- If ϕ, ψ are $\mathcal{ALC}$-LTL formulas, then $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \mathsf{U} \psi$ and $\mathsf{X}\psi$ are $\mathcal{ALC}$-LTL formulas.

Example:

$$\Diamond\Box(PKCitizen \sqsubseteq \exists insured\_by.HealthInsurance)$$

Abbreviations: $true \equiv \phi \vee \neg\phi$, $\Diamond\phi \equiv true\mathsf{U}\phi$ and $\Box\phi \equiv \neg(true\mathsf{U}\phi)$.

Temporalized DL $\mathcal{ALC}$-LTL

The semantics of $\mathcal{ALC}$-LTL are described in [Franz Baader 2008] by using an $\mathcal{ALC}$-LTL structure.

$\mathcal{ALC}$-LTL Structure:

An $\mathcal{ALC}$-LTL structure is a sequence $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ of $\mathcal{ALC}$ interpretations $\mathcal{I}_i = (\Delta, .^{\mathcal{I}_i})$.

$$\mathcal{I}_0 \Rightarrow \mathcal{I}_1 \Rightarrow \mathcal{I}_2 \Rightarrow \ldots$$

Example:

$$\Diamond\Box(PKCitizen \sqsubseteq \exists insured\_by.HealthInsurance)$$

$$\mathcal{I}_0 \Rightarrow \mathcal{I}_1 \Rightarrow \mathcal{I}_2 \Rightarrow \mathcal{I}_3 \Rightarrow \ldots$$

Temporalized DL $\mathcal{ALC}$-LTL

Example:

$$\exists has\_father.Human \sqcap \Diamond(\forall has\_father.\neg Human)$$

A concept is called a *rigid concept* if its interpretation does not change over time.

Rigid Concept: [Franz Baader 2008]

For any given $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\dots}$, a concept $A \in N_C$ is a rigid concept iff for all $i, j \in \{0, 1, 2, \dots\}$ it holds that $A^{\mathcal{I}_i} = A^{\mathcal{I}_j}$.

A role is called *rigid role* if its interpretation does not change over time.

Rigid Role: [Franz Baader 2008]

For any given $\mathcal{ALC}$-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\dots}$, a role $r \in N_R$ is a rigid role iff $r^{\mathcal{I}_i} = r^{\mathcal{I}_j}$ holds for all $i, j \in \{0, 1, 2, \dots\}$.

Web Ontology Language (OWL)

OWL: ([Michael K. Smith 2003])

OWL provides a family of languages (i.e., OWL Lite, OWL DL and OWL Full) which are used to author OWL Ontologies. OWL provides a common standard for representing, exchanging and deriving logical consequences from different domains. Thus, providing machine-processable descriptions of domains including World Wide Web.

The OWL API: ([Sean Bechhofer 2003])

The OWL API is a programming interface to access and manupulate OWL Ontologies.

The OWL API & Descripton Logic $\mathcal{ALC}$

The OWL API provides appropriate data structure to deal with the description logic $\mathcal{ALC}$.

$\mathcal{ALC}$	OWL API
$a \in N_I$	OWLIndividual
$A \in N_C$	OWLClass
$r \in N_R$	OWLObjectProperty
$\sqcap$	OWLObjectIntersectionOf
$\sqcup$	OWLObjectUnionOf
$\neg$	OWLObjectComplementOf
$\exists$	OWLObjectSomeValuesFrom
$\forall$	OWLObjectAllValuesFrom

Outline

Motivation

1. Generated formulae help to test and measure performance of $\mathcal{ALC}$-LTL related systems.
   - Generate Random Sets (N_C, N_R and N_I),
   - Generate $\mathcal{ALC}$-axioms,
   - Generate $\mathcal{ALC}$-LTL formulae,
   - Load generated formulae.
2. The OWL API is insufficient to deal with $\mathcal{ALC}$-LTL.
3. Appropirate data structures are required to represent $\mathcal{ALC}$-LTL formulae.

Outline

Generating Algorithms

1. $\mathcal{ALC}$-LTL formulae:

Generating Algorithms

1. $\mathcal{ALC}$-LTL formulae:
   - $\mathcal{ALC}$-LTL formulae Algorithm (#3).
2. $\mathcal{ALC}$-axioms:

Generating Algorithms

1. $\mathcal{ALC}$-LTL formulae:
   - $\mathcal{ALC}$-LTL formulae Algorithm (#3).
2. $\mathcal{ALC}$-axioms:
   - $\mathcal{ALC}$-axioms Algorithm (#2)
3. $\mathcal{ALC}$-concept descriptions:

Generating Algorithms

1. $\mathcal{ALC}$-LTL formulae:
   - $\mathcal{ALC}$-LTL formulae Algorithm (#3).
2. $\mathcal{ALC}$-axioms:
   - $\mathcal{ALC}$-axioms Algorithm (#2)
3. $\mathcal{ALC}$-concept descriptions:
   - $\mathcal{ALC}$-concept descriptions Algorithm (#1).

$\mathcal{ALC}$-concept descriptions Algorithm (#1)

Algorithm Signature:

```
function alc_cd(n:Integer):ALC-concept description
begin
  ...
end function
```

Termination:

The procedure `alc_cd`(n) always terminates where $n \geq 1$.

Soundness:

The generated $\mathcal{ALC}$-concept description is always a well-formed $\mathcal{ALC}$-concept description.

Completeness:

The procedure *alc_cd(l)* returns all $\mathcal{ALC}$-concept descriptions of length l.

$\mathcal{ALC}$-axioms Algorithm (#2)

Algorithm Signature:

```
function alc_axiom(m:Integer):ALC-axiom
begin
  ...
end function
```

Termination:

The procedure `alc_axiom(m)` always terminates where $m \geq 1$.

Soundness:

The generated $\mathcal{ALC}$-axiom is always a well-formed $\mathcal{ALC}$-assertion axiom.

Completeness:

The procedure `alc_axiom(m)` returns all $\mathcal{ALC}$-axioms of length m.

$\mathcal{ALC}$-LTL formulae Algorithm (#3)

Algorithm Signature:

```
function alc_ltl_fm(m:Integer, n:Integer):ALC-LTL formula
begin
  ...
end function
```

Termination:

The procedure `alc_ltl_fm(m,n)` always terminates where $m, n \geq 1$.

Soundness:

The generated formula is always a well-formed $\mathcal{ALC}$-LTL formula.

Completeness:

The procedure `alc_ltl_fm(m, l)` returns all $\mathcal{ALC}$-LTL formulae of length l.

Optimization

In order to generate more compact and meaningful data following are some proposed optimizations

1. Eliminate multiple negation case.
2. User defined random sampling procedure.

Probability Sampling

Type	Operators	Probability Value
$\mathcal{ALC}$-concept description	$\sqcap$	0.3f
	$\sqcup$	0.5f
	$\exists$	0.2f
	$\forall$	0.6f
	$\neg$	0.1f
$\mathcal{ALC}$-axiom	CLASS_ASSERTION	0.4f
	ROLE_ASSERTION	0.1f
	NEG_ROLE_ASSERTION	0.1f
	$\sqsubseteq$	0.4f
$\mathcal{ALC}$-LTL Formula	$\neg$	0.5f
	X	0.5f
	$\wedge$	0.5f
	$\vee$	0.5f
	U	0.5f

Outline

Structure of $\mathcal{ALC}$-LTL formulae XML File

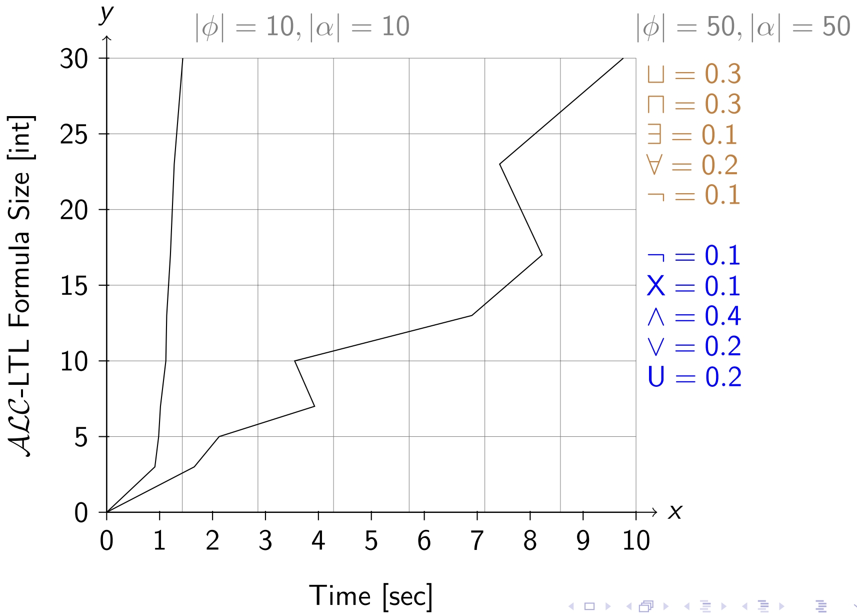
$\mathcal{ALC}$-axioms are stored in an OWL ontology file. We store $\mathcal{ALC}$-LTL formulae in an XML file using the following mapping.

$\mathcal{ALC}$-LTL XML mapping:

Operators	XML Mapping
$\neg$	$\langle$NegationOf$\rangle \ldots \langle$/NegationOf$\rangle$
$\sqcap$	$\langle$ConjunctionOf$\rangle \ldots \langle$/ConjunctionOf$\rangle$
$\sqcup$	$\langle$DisjunctionOf$\rangle \ldots \langle$/DisjunctionOf$\rangle$
X	$\langle$NextOf$\rangle \ldots \langle$/NextOf$\rangle$
U	$\langle$UntilOf$\rangle$ $\langle$LeftOf$\rangle \ldots \langle$/LeftOf$\rangle$
	$\langle$RightOf$\rangle \ldots \langle$/RightOf$\rangle$ $\langle$/UntilOf$\rangle$

Program Demo

Performance Measures for $\mathcal{ALC}$-LTL Formulae Generator



Outline

For Further Reading

- Manfred Schmidt-Schauß and Gert Smolka. 1991
  Attributive Concept Descriptions with Complements.
- Franz Baader, Silvio Ghilardi and Carsten Lutz. 2008
  LTL over Description Logic Axioms.
- Franz Baader, Ian Horrocks and Ulrike Sattler. 2008
  Handbook of Knowledge Representation, "Description Logics".
- Michael K. Smith, Chris Welty and Deborah L. McGuinness. 2003
  OWL Web Ontology Language Guide
- Sean Bechhofer, Rapheal Volz and Phillip Lord. 2003
  Cooking the Semantic Web with the OWL API

Thank You

Appendix I

Length of $\mathcal{ALC}$-concept description:

For an arbitrary $\mathcal{ALC}$-concept description C, its length $|C|$ is computed inductively:

- If $C \in N_C$, then $|C| := 1$,
- If C is of form $\neg D$ for some $\mathcal{ALC}$-concept description D, then $|C| := |D| + 1$,
- If C is of form $D \sqcap E$ for $\mathcal{ALC}$-concept descriptions D and E, then $|C| := |D| + |E| + 1$,
- If C is of form $D \sqcup E$ for $\mathcal{ALC}$-concept descriptions D and E, then $|C| := |D| + |E| + 1$,
- If C is of form $\exists r.D$ for some $\mathcal{ALC}$-concept description D and role $r \in N_R$, then $|C| := |D| + 1$,
- If C is of form $\forall r.D$ for some $\mathcal{ALC}$-concept description D and role $r \in N_R$, then $|C| := |D| + 1$,

Appendix II

Length of $\mathcal{ALC}$-LTL formulae:

For an arbitrary $\mathcal{ALC}$-LTL ϕ, its length $|\phi|$ is inductively computed in the following way.

- If ϕ is an $\mathcal{ALC}$-assertion axiom, then $|\varphi| := 1$,
- If ϕ is of form $\neg\psi$ for some $\mathcal{ALC}$-LTL formula ψ, then $|\phi| := |\psi| + 1$,
- If ϕ is of form $\varphi \wedge \psi$ for $\mathcal{ALC}$-LTL formulae φ and ψ, then $|\phi| := |\varphi| + |\psi| + 1$,
- If ϕ is of form $\varphi \vee \psi$ for $\mathcal{ALC}$-LTL formulae φ and ψ, then $|\phi| := |\varphi| + |\psi| + 1$,
- If ϕ is of form $\varphi \mathsf{U} \psi$ for $\mathcal{ALC}$-LTL formulae φ and ψ, then $|\phi| := |\varphi| + |\psi| + 1$.