Description Logic の Prolog 化

鎌田達哉

Japan Advanced Institute of Science and Technology
tatuya@jaist.ac.jp

2014 年 10 月 2 日

1 目的

本稿では、Description Logic (記述論理) によって記述したある対象領域の知識表現を論理プログラミング言語である Prolog で翻訳を行う。これにより Description Logic の Prolog から見た姿、また Description Logic から見た Prolog の姿を明らかにし、それぞれの理解を深める。

2 Description Logic の成り立ち

Description Logic は、*semantic network* [1] や *frame systems* [2] に代表される *knowledge representation* (知識表現) と、その *reasoning* (推論) の形式化、 特に曖昧であったそれらの *semantics* (意味論) を与えることをモチベーションとして開発された。

知識表現 の semantics (意味論) はまず first-order logic によって与えられ [3]、さらにその semantics は first-order logic の部分で良いことが示された [4]。また、その reasoning の計算複雑性は first-order logic のどの部分を利用するかによって異なることも明らかになった。異なる知識表現の言語は異なる first-order logic の部分となる。これらの文脈のもとで KL-ONE [5] や KRYPTON [6] といった言語そして、Description Logic が開発された。

2.1 Description Language: $\mathcal{AL}, \mathcal{ALUEN}$

ここでは利用する Description Logic の言語を定義する。Description Logic の言語は concept の constructor の種類によって、それぞれ別の言語、AL-languages [7] として定義されている。

AL(=attributive language)-languages のなかでもっとも基礎的な言語は、以下の $\mathcal{AL}$ である。

記法として、A, B を atmoic concepts, R を atomic role, C, D を concept discriptions とする。

$\mathcal{AL}$ Syntax:

$$
\begin{aligned}
C, D \longrightarrow A &\mid \text{(atomic concept)} \\
\top &\mid \text{(universal concept)} \\
\bot &\mid \text{(bottom concept)} \\
\neg A &\mid \text{(atomic negation)} \\
C \sqcap D &\mid \text{(inter section)} \\
\forall R.C &\mid \text{(value restriction)} \\
\exists R.T & \ \text{(limited exisitential quantification)}
\end{aligned}
$$

本稿では、$\mathcal{AL}$ に以下の $C \sqcup D, \exists R.C, n \geq R$ (それぞれ $\mathcal{U}, \mathcal{E}, \mathcal{N}$) の 3 つの constructors を加えた $\mathcal{ALUEN}$ を知識表現の言語として採用する。すなわち、$\mathcal{ALEUN} = \mathcal{AL} + \mathcal{UEN}$ である。

$\mathcal{UEN}$ Syntax:

$$
\begin{aligned}
C, D \longrightarrow C \sqcup D &\mid \text{(union)} \\
\exists R.C &\mid \text{(full exisitential quantification)} \\
n \geq R & \ \text{(number restrictions)}
\end{aligned}
$$

同時に $\mathcal{ALUEN}$ の Semantics として、Interpretation $\mathcal{I}$ を以下に与える。Interpretation の Domain は $\Delta^{\mathcal{I}}$ = non empty set である。

$\mathcal{ALUEN}$ Semantics:

$$
\begin{aligned}
A^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \\
R^{\mathcal{I}} &\subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset^{\mathcal{I}} \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(n \geq R)^{\mathcal{I}} &= \left\{a \in \Delta^{\mathcal{I}} \middle| \ |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\right\}
\end{aligned}
$$

2.2 Terminological axioms: TBox

前節で concepts の作り方を定義した。Description Language において concept と他の concept との関係を定義するのが、*terminological axioms* である。

terminological axioms Syntax:

$$
\begin{aligned}
C \equiv D & \ \text{(equalities)} \\
C \sqsubseteq D & \ \text{(inclusion)}
\end{aligned}
$$

terminological axioms Semantics ($\mathcal{I}$: Interpretation):

$$\begin{aligned}\mathcal{I} &\models C \equiv D \ \text{ iff } C^{\mathcal{I}} = D^{\mathcal{I}} \\ \mathcal{I} &\models C \sqsubseteq D \ \text{ iff } C^{\mathcal{I}} \subseteq D^{\mathcal{I}} \\ \mathcal{I} &\models \mathcal{T} \ \text{ iff } \ \forall \Phi.(\mathcal{I} \models \Phi \in \mathcal{T})\end{aligned}$$

上で定義した terminological axioms の有限集合 $\mathcal{T}$ を、*a terminology* または *TBox* という。

2.3 Definitions

TBox の equality(≡) の左辺が Atomic concept である等式は *Definitions* として用いられる。Definitions の使途はより複雑な記述 (Description) のための *symbolic names* を導入することである。以下に例を挙げる。

Example

$Woman \equiv Person \sqcap Female$
$Man \equiv Person \sqcap \neg Woman$
$Mother \equiv Woman \sqcap \exists hasChild.Person$
$Parent \equiv Mother \sqcup Father$
$Wife \equiv Woman \sqcap \exists hasHusband.Man$

また、Symbolic names の分類として、Definition の左辺を *Name symbols*, 右辺を *Base Symbols* という。

2.4 World desciptions: ABox

Description Logic において、Concept や Role 内の個々の name の構成を定義するのが、*world desciptions* (*ABox*) である。

以下の Syntax では、*a,b,c* を name, concepts を C, role を R とする。

world desciption Syntax:

$$\begin{aligned}C(a) &\ \text{ (concept assertion)} \\ R(b,c) &\ \text{ (role assertion)}\end{aligned}$$

world desciption Semantics:

$$\begin{aligned}&\mathcal{I} \models C(a) \text{ iff } a^{\mathcal{I}} \in C^{\mathcal{I}} \\ &\mathcal{I} \models R(b,c) \text{ iff } (b^{\mathcal{I}}, c^{\mathcal{I}}) \in R^{\mathcal{I}} \\ &\mathcal{I} \models \mathcal{A} \text{ iff } \forall \phi(\mathcal{I} \models \phi \in \mathcal{A})\end{aligned}$$

上で定義した $\mathcal{A}$、すなわち assertion の有限集合を *world desciption* または *ABox* という。

2.5 First-order Logic への埋め込み (embedding)

Description Logic は First-order Logic の subset であり、以下のように First-order Logic へ埋め込み (embedding) [10] を行うことができる。

$$\phi \exists R.C(y) = \exists x.R(y,x) \wedge \phi C(x)$$
$$\phi \forall R.C(y) = \forall x.R(y,x) \rightarrow \phi C(x)$$
$$\phi \geqslant nR(x) = \exists y_1, ...y_n.R(x,y_1) \wedge ... \wedge R(x,y_n) \wedge \bigwedge_{i<j} y_i \neq y_j$$
$$\phi \leqslant nR(x) = \forall y_1, ...y_n.R(x,y_1) \wedge ... \wedge R(x,y_{n+1}) \rightarrow \bigvee_{i<j} y_i = y_j$$

2.6 Description Logic による人事情報の知識表現

前節までで定義した Description Logic の言語を用いて、ある組織の人事情報を表現することを考える。

注意が必要な点は、$\mathcal{ALUEN}$ における $\forall, \exists$ の Interpretation では Role の *domain* しか集められないことである。そのため Description Logic で、ある Role の *codomain* が必要な場合は、あらかじめ逆向きの Role を記述する必要がある。

```
# A_0 は ABox
A_0 = {
  employ(FooCompany, John),
  employ(BarCompany, Mary),
  employ(BarCompany, Jeremy),
  employ(BarCompany, Camilla),
  employ(BazCompany, Percival),
  employ(BazCompany, Lucinda),

  # belong は employ の 逆向きの Role
  belong(John, FooCompany),
  belong(Mary, BarCompany),
  belong(Jeremy, BarCompany),
  belong(Camilla, BarCompany),
  belong(Percival, BazCompany),
  belong(Lucinda, BazCompany),

  Development(Percival),
  Development(Lucinda),
  Development(Camilla),
  Development(John),
  Development(Mary),
  Finance(Jeremy),
  Finance(John),
  Finance(Mike),
  Finance(Ann),

  qualify(John, Programming),
  qualify(John, Accounting),
  qualify(Percival, Programming),
  qualify(Percival, Accounting),

  Qualification(Programming),
  Qualification(Accounting)
}.

# TBox
BarCompany ⊔ FooCompany ⊑ Construction
```

$BazCompany \sqsubseteq Pharmaceutical$
$Generalist \equiv 2 \geq Qualify$

Code 1　ある組織の人事情報の知識表現 (Description Logic)

上の例の Concept と Role が表現している知識の説明を行う。

employ(**組織**, **人**)
: ある組織が雇用している人

belong(**人**, **組織**)
: ある人が所属している組織

Development(**人**)
: 開発部門に所属してる人

Finance(**人**)
: 経理部門に所属してる人

Qualify(**人**, **資格**)
: ある人が持っている資格

Qualification(**資格**)
: 資格の名前

$Construction$
: 建設会社

$Pharmaceutical$
: 製薬会社

$Generalist \equiv 2 \geq Qualify$
: 資格を 2 つ以上持っている人は $Generalist$, 万能型 (ジェネラリスト)

3　Prolog への翻訳

ここでは、Description Logic のそれぞれの要素、$\mathcal{ALEUN}$, TBox, ABox を Prolog に翻訳する方法を検討する。

3.1　$\mathcal{ALEUN}$ の翻訳

3.1.1　⊤

```
true.
```

Code 2　⊤ の翻訳 (Prolog)

3.1.2　⊥

```
fail.
```

Code 3　⊥ の翻訳 (Prolog)

3.1.3 $\neg A$

```
not(c(a)).
```

Code 4 $\neg A$ の翻訳 (Prolog)

3.1.4 $C \sqcap D$

```
c(X), d(X).
```

Code 5 $C \sqcup D$ の翻訳 (Prolog)

3.1.5 $\forall R.C$

```
setof(Y, r(X, Y), L), findall(Z, c(Z), L2), subset(L, L2).
```

Code 6 $\forall R.C$ の翻訳 (Prolog)

3.1.6 $\exists R.T$

```
findall(Y, r(X, Y), L).
```

Code 7 $\exists R.T$ の翻訳 (Prolog)

3.1.7 $C \sqcup D$

```
c(X); d(X).
```

Code 8 $C \sqcup D$ の翻訳 (Prolog)

3.1.8 $\exists R.C$

```
setof(Y, X^(r(X, Y), c(Y)), L.
```

Code 9 $\exists R.C$ の翻訳 (Prolog)

3.1.9 $n \geq R$

```
setof(Y, r(X, Y), L), length(L, LL), LL >= n.
```

Code 10 $n \geq R$ の翻訳 (Prolog)

3.2 TBox の翻訳

3.2.1 $C \sqsubseteq D$ (inclusion)

```
d(X) :- c(X).
```

Code 11 $C \sqsubseteq D$ の翻訳 (Prolog)

3.2.2 $C \equiv D$ (equalities)

```
c(X) :- d(X).
d(X) :- c(X).
```

Code 12 $C \equiv D$ の翻訳 (Prolog)

3.3 ABox の翻訳

3.3.1 $C(a)$ (concept assertion)

```
c(a).
```

Code 13 $C(a)$ の翻訳 (Prolog)

3.3.2 $R(b, c)$ (role assertion)

```
r(b, c).
```

Code 14 R(b,c) の翻訳 (Prolog)

3.4 Prolog への翻訳と実行例

翻訳の具体例として、2.4 節で定義したある組織の人事情報を Prolog に翻訳することを考える。前節で見たように、ABox については Description Logic から Prolog への翻訳は直接的に行われる。

```
# ABox の翻訳
employ(foo_company, john).
employ(bar_company, mary).
employ(bar_company, jeremy).
employ(bar_company, camilla).
employ(baz_company, percival).
employ(baz_company, lucinda).

belong(john, foo_company).
belong(mary,bar_company).
belong(jeremy, bar_company).
belong(camilla, bar_company).
belong(percival, baz_company).
belong(lucinda, baz_company).

development(percival).
development(lucinda).
development(camilla).
development(john).
development(mary).
finance(jeremy).
finance(john).
finance(mike).
finance(ann).

qualify(john,programming).
```

```
qualify(john,accounting).
qualify(percival, programming).
qualify(percival, accounting).

qualification(programming).
qualification(accounting).

# TBox の翻訳
construction(X) :- X = foo_campany; X = bar_company.
pharmaceutical(X) :- X = baz_company.

doublequal(L3) :-
  findall(X, (setof(Y, qualify(X, Y), L2), length(L2, L2L), L2L >=2), L3),!.

doublequal(X) :-
  generalist(X).

generalist(X) :-
  doublequal(X).
```

Code 15　人事情報の知識表現 (Prolog)

3.4.1　Example Interpretations

いま翻訳した人事情報の知識表現に対して、いくつかの Interpretation (解釈) を行う。

$(\exists Belong.Construction)^{\mathcal{I}}$

　　建設会社に所属する人

$(\exists Belong.Pharmaceutical)^{\mathcal{I}}$

　　製薬会社に所属する人

$(\exists Employ.Development)^{\mathcal{I}}$

　　開発部署社員を持つ会社

$(\forall Employ.(2 \geq Qualify))^{\mathcal{I}}$

　　全ての社員が資格を 2 個以上持つ会社

$(Generalist)^{\mathcal{I}}$

　　万能型 (ジェネラリスト) の人

3.4.2　$(\exists Belong.Construction)^{\mathcal{I}}$

```
?- setof(P, C^(belong(P, C), construction(C)), I).
I = [camilla, jeremy, john, mary].
```

Code 16　$(\exists Belong.Construction)^{\mathcal{I}}$ の翻訳と実行 (Prolog)

3.4.3　$(\exists Belong.Pharmaceutical)^{\mathcal{I}}$

```
?- setof(P, C^(belong(P, C), pharmaceutical(C)), I).
I = [lucinda, percival].
```

Code 17　$(\exists Belong.Construction)^{\mathcal{I}}$ の翻訳と実行 (Prolog)

3.4.4 $(\exists Employ.Development)^{\mathcal{I}}$

```
?- setof(C, P^employ(C, P), development(P)), I).
I = [bar_company, baz_company, foo_company].
```

Code 18 $(\exists Employ.Development)^{\mathcal{I}}$ の翻訳と実行 (Prolog)

3.4.5 $(\forall Employ.(2 \geq Qualify))^{\mathcal{I}}$

```
?- setof(C, (setof(P, employ(C, P), L), findall(X, (setof(Y, qualify(X, Y), L2), length(
    L2, L2L), L2L >=2), L3), subset(L, L3)), CL).
L = [john],
L3 = [john, percival],
CL = [foo_company].
```

Code 19 $(\forall Employ.(2 \geq Qualify)^{\mathcal{I}}$ の翻訳と実行 (Prolog)

3.4.6 $(Generalist)^{\mathcal{I}}$

```
?- generalist(X).
X = [john, percival].
```

Code 20 $(Generalist)^{\mathcal{I}}$

4 まとめ

本稿では、Description Logic によって記述したある対象領域の知識表現を Prolog に翻訳した。以下に翻訳の過程で学んだ事柄を述べる。

まず、ABox の翻訳において Description Logic の ABox と、Prolog の Fact [11] はその Syntax、Semantics においてほぼ等価であるということを見た。
次に、Description Language $\mathcal{ALUEN}$ の $\forall, \exists$ の翻訳では、Prolog の findall や setof といった metapredicate (メタ述語) を使用した。この翻訳は ABox の翻訳に比べると不自然であり、Prolog の metapredicate は強力であるが万能ではないことを学んだ。

$\mathcal{ALUEN}$ の翻訳時には同時に、$\forall$、$\exists$ の Interpretation は Role の domain 側の収集しか行わないということを見た。Role の codomain を集めるよう拡張した言語を考えることは容易であるが、$\mathcal{ALUEN}$ はそのように制限された言語であるということがわかった。
最後に、一連の翻訳のなかで、ある知識記述言語の Syntax と Semantics (Interpretation) を、集合の記法と一階述語論理を利用して定義する方法を学んだ。

参考文献

[1] M, Ross Quillian. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410-430, 1967.

[2] Marvin Minsky. A framework for representing knowledge. In J. Haugeland, editor, *Mind Design*. The MIT Press, 1981.

[3] Patrick J.Hayes. The logic of frames. In D. Metzing, editor, *Frame Conceptions and Text Understanding*, pages 46-61. Walter de Gruyter and Co., 1979.

[4] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation*, Morgan Kaufmann, Los, Altos, 1985.

[5] Ronald J. Brachman and James G.Schmolze. An overview of the KL-ONE kowledge representation system. *Cognitive Science*, 9(2):171-216, 1985.

[6] Ronald J. Brachman, Richard E.Eikes, and Hector J.Levesque. KRYPTON: A functional approach to knowledge representation. *IEEE Computer*, October:67-73, 1983.

[7] Schmidt-Schauß, Gert Smolka. Attributive concept descriptions with complements. Artificial Intelligence, 48(1):1-26,1991.

[8] The Discription Logic Handbook: theory, implementation, and applications, Cambridge University Press New York, NY, USA, 2003

[9] `http://en.wikipedia.org/wiki/Description_logic`

[10] `http://en.wikipedia.org/wiki/Embedding`

[11] Logic programming and knowledge representation – The A-prolog perspective. Artificial Intelligence, 138:3-38, 2002

[12] Description Logic Programs: Combining Logic Programs with Description Logic, Benjamin N. Grosof, Ian Horrocks, Raphael Volz, Stefan Decker, 2003
`http://papers.ssrn.com/sol2/papers.cfm?abstract_id=460986`