知識継承における例外の扱いとその応用について

德永 健伸, 田中 碩積
東京工業大学

1. はじめに
筆者らはこれまで、知識表現形式 DCKR (Definite Clause Knowledge Representation) を提案し、自然言語の意味処理などに応用してきた [田中 86]。

DCKR では、概念の各スロットを基本述語 `sem` をヘッドとするホーン節で表し、知識の継承は `isa` という述語を介しておこなう。DCKR の形式で知識を表現すると推論のためのインタプリタを特別に用意しなくても、Prolog に組み込む機能で推論機構を代用できるという利点がある。しかし、そのままでは推論過程を Prolog のインタプリタに委ねるため、ユーザが明示的に推論過程を制御できない。知識表現において下位概念は基本的に上位概念のもつ性質を継承することになるが、例外的に継承されてならない性質が存在する。このような例外を含む意味ネットワークに関する研究として [Fahlman 81], [Etherington 83], [Tourretzky 86] などがある。特に、Etherington は Default Logic [Reiter 80] を用いてこのような例外を許す意味ネットワークを定式化した。本稿ではこの Etherington の定式化を参考にして継承の例外が取り扱えるように DCKR を拡張する。

2. DCKR による知識表現
DCKR では図 1 のように 3 引数の基本述語 `sem` を用いて知識を表現する。`sem` 述語の第 1 引数は概念名、第 2 引数はスロット名とスロット値がオペレータで結合されたもの、第 3 引数は、継承の履歴を保存するためのスタックである。

?- `sem(tom, Property, _)`. を実行すればよい。Prolog のユニフィケーションとバックトラックにより、その概念に関する知識が次々に得られる。

\[
\begin{align*}
&\text{isa(Upper, P, S):=} \\
&P = \text{isa:Upper;} \\
&\text{sem(Upper, P, S)}.
\end{align*}
\]

図 2 `isa` 述語の定義

3. 継承の例外の扱い
Etherington は [Etherington 83] の中で 5 種類のリンクを使った継承の例外を含む意味ネットワークを Default Logic を用いて定式化した。本稿では、この Etherington の定式化を参考にして、継承の例外を扱えるように DCKR を拡張する。ただし、否定の知識に関しては従来の DCKR と同様に閉世界仮説を採用することにする。すなわち、[Etherington 83] における。

\[
\begin{align*}
&\text{Strict IS-A (---\rightarrow)} \\
&\text{Default IS-A (\rightarrow\rightarrow)} \\
&\text{Exception (---\rightarrow)}
\end{align*}
\]

の 3 つのリンクだけでネットワークを記述する。

4. 記述形式
継承の例外を表現するために DCKR に exception 述語を新たに導入する。まず例をあげて exception 述語の意味を説明する。

[図 1]

知識の継承は述語 `isa` (定義を図 2 に示す) を介しておこなう。データベースからある概念（例えば `tom`）に関する知識を引き出すには Prolog のゴール

The handling of exceptions in knowledge inheritance and its applications
Takenobu TOKUNAGA, Hozumi TANAKA
Tokyo Institute of Technology
図3において(4-1)と(4-2)はStrict IS-Aリンクに相当し、この継承は常に起こってよいことを示している。これは従来のDCKRの表現と同じである。(4-3)では「dがcの性質を継承してよいのは、これまでにやってきた経路の概念がcの性質を継承しない場合である」ということを表現している。exception語法の第1引数は現在注目している概念、第2引数は継承を阻害する概念（これを例外概念と呼ぶことにする）のリスト、第3引数は継承経路の履歴が保存されているスタックである。exception語法はこれまでにやってきた経路の概念のいずれかが例外概念の性質を継承するなら失敗し、そうでなければ成功する語法である。

図3の概念aが成功できる性質について考えてみよう。まずaからbとcの性質はStrict IS-Aリンクにより継承できる。次にbを経由してdの性質を継承しようとするがスタックの中にはaが存在し、aからcの性質であるcの性質が継承できるのでaはdの性質を継承することができない。したがって、aが継承できるのはbとcの性質だけということになる。

一方、bは例外概念であるcの性質を継承できないので(4-3)より、dの性質を継承することができる。

5. 実現

DCKRにおけるexception語法の実現は容易である。基本となるアルゴリズムは以下の通りである。

```plaintext
def exception(node, ExList, Stack):
    for all element e of ExList do
        if e == s then fail
    for all element s but bottom of Stack do
        if sem(s, isa:e, node) success then fail
    od
```

図4 exception語法のアルゴリズム

外側のループではexception語法の第2引数に渡される例外概念をひとつずつ取り出し、内側のループではスタックの概念をひとつずつ取り出して、(1)継承経路中に例外概念が含まれるか、(2)継承経路中に含まれる概念中に例外概念の性質を継承するものがあるか、チェックをしている。途中failすることなくプロンプットを抜けて出るとexception語法は成功する。したがって、exception語法の効率はスタックの深さと例外概念の数が増えるにつれて低下する。常識的に「例外」というのは数少であるからexception語法はほとんどの場合成功するはずである。効率という面からはexception語法によって例外の記述をおこなうことには問題がある。

6. 応用

現在、筆者らは自然言語の意味処理に使用することを前提として概念の上位／下位関係を表すシーソーラスを作成中である。そのなかでどうしても継承の例外を表現する必要性を感じている。例えば、図5において「ガソリン」も「食用油」も「油」の下位概念であるが、「食用油」からは「油」の上位概念である「燃料」の性質を継承させたくない、といった場合である。例外として扱わなくても、知識の構造を変えることによって、このような問題を解消できる場合もあるが、現在、作成中のシーソーラスの模様（最下位の概念数で約1000）になると、知識の構造を変えることは、他の部分との関連もあり、なかなか困難であると思、例外として扱った方が自然な表現となる場合が多い。

図5

燃料

汽油

食用油

7. まとめ

本稿では知識表現形式DCKRを継承の例外が扱えるように機能を拡張した。exception語法を用いたと知識の記述が複雑になるのでより記述しやすい形式を考案することが必要である。現在、DCKRの上位水準の言語であるSRL/O【奥村86】に例外の記述でできるように拡張することを考えている。また、実際の応用でも使用し、効率などの評価をおこなう必要がある。

参考文献

Etherington, D.W. and Reiter, R.,
"On inheritance hierarchies with exceptions",
In Proc. AAAI-83, Washington, DC,

Fahlman, S.E., Tourletzky, D.S., Roggen, W.,
"Cancellation in a Parallel Semantic Network",
In Proc. IJCAI-81, pp257-263,
August, 1981.

奥村, 小池康晴, 田中徳積,
“意味記述用言語SRL/Oの設計とDCKR”

Reiter, R.,
"A Logic for Default Reasoning",
Artificial Intelligence 13, pp81-132,
April, 1980.

田中徳積, 小山晴生, 奥村,
“知識表現形式DCKRとその応用”.

Tourletzky, D.S.,
The Mathematics of Inheritance Systems,