2H-5

推論システムのためのプログラミング環境としての知識表現システム

松方 純・田中 英彦・元岡 達
(東京大学 工学部)

0. はじめに、

知識へのアクセス法もまた知識としてもつ知識表現システムの構想について述べる。このシステムのねらいは、エキスパート・システムなどの推論機構をもつシステムの作成のために、高級なプログラミング環境を提供することである。

1. システムの特徴

(1) 知識へのアクセス法に関する知識をメタ知識としてもつ。
(2) 知識へのアクセス法としては、いろいろな程度の推論を伴なうものを設定できる。
(3) メタ知識は、本来の知識、すなわち、システムの適用分野の知識と同じやり方で、知識として、記憶される。
(4) 知識表現の枠組は、述語論理にもとづく。
(5) 知識の外部表現、すなわち、このシステムへの問合わせにおける知識の表現は、述語論理にもとづいた表現が使われる。(この外部表現は、Prolog/KRにおける知識表現との適合が考慮されている。)
(6) いろいろな推論手続を組み込むことができる。
(7) 知識の内部表現として、いろいろな知識表現が使用できる。
(8) 知識の分画化、階層化が考慮されている。
(9) システムの実装に使用する言語は、論理プログラミング言語 (Prolog/KR) を使用する。

2. メタ知識

このシステムでは、メタ知識を、本来の知識、すなわち、システムの適用分野の知識 (たとえば、電子工学の知識、医学の知識) と同じように、知識として表現される。

3. 知識へのアクセス法

このシステムでは、メタ知識として、知識へのアクセス法に関する知識をもつ。この知識には、知識の内部表現の選択、推論手続の選択等に関する記述が含まれる。知識へのアクセス法は、いろいろな程度の推論を行なうものを設定できる。

知識へのアクセス法の規定は、原則として、Assert、Lookup (あるいは、Prove-to-be-true)、Unassert の三つのモードのアクセスを定義することによる。

4. 知識へのアクセス法の記述の例

以下は、Assert の定義の例である。

```
( assert ( Assert . *q ) ( access MyAssert *q ))          Assert の定義
( assert ( access *p *q ) ( howto *p *q *r ) (*r . *q ))   access の定義
```

このとき、howto として、次のような知識が記憶されているとすると、Assert の定義は、システムの assert ということになる。

```
( assert ( howto MyAssert *q assert ))
```

5. 知識へのアクセス法の例

次の例は、対象領域における推論を行なわないようなアクセス法の例である。

```
> (Record (color-of Clyde grey))
recorded
> (Lookup (color-of Clyde *x))
(color-of Clyde grey)
> (Unrecord (color-of Clyde grey))
unrecorded
> (Lookup (color-of Clyde *x))
nil
```

次の例は、推論を伴なうアクセス法の例である。

```
> (Assert (fallible *x) (human *x))     (A human being is fallible.)
asserted
> (Assert (human Socrates))
asserted
> (Prove-to-be-true (fallible *x))
(fallible Socrates)
> (Lookup (fallible *x))
nil
```

Prove-to-be-true が推論を行なうのに対し、Lookup は、推論を行なわない。

6. 知識の内部表現

知識の内部表現としては、いろいろな表現を使うことができる。知識表現の形態として、Prolog/KR の知識ベース、性質リスト、機械語のルーチンなど、様々なものが考えられる。これらの内部表現は、知識へのアクセス法に関する知識によって、使いわけることができる。

7. 推論の制御

推論の制御の記述も知識へのアクセス法として記述される。手続付加は、アクセス法として、手続を呼び出すという形で実現できる。Forward chaining は、Assert において、backward chaining は、Lookup において、推論を行なうという形で実現できる。

推論手続は、いろいろなものを組み込んで使用できる。特定の推論手続が起動されるための条件は、知識へのアクセス法の一部として記述される。

8. 実装

本システムは、論理プログラミング言語 Prolog/KR 上で実装される。Prolog/KR の推論機能は、本システムの主要な推論機構として、使用される。

9. おわりに

本システムが上に実装される Prolog/KR は、知識表現を目的としている言語である。本システムは、内部表現、推論手続に関して、Prolog/KR より柔軟なものになっており、論理プログラミング言語の発展の一つの方向を示すものと考える。

[参考文献]

(i) M.R. Genesereth, R. Greiner, and D. Smith, "MRS Manual," Stanford Heuristic Programming Project Memo, HPP-80-24, 1980.

(ii) H. Nakashima, "Prolog/KR User's Manual," 1982.