単一化について

情報工学科　篠埜 功

この資料では、論理型プログラミングの資料の補足として単一化アルゴリズムについて述べる。

単一化 (unify) とは、2 つの term s と t が与えられたとき、ある置換 σ をそれらに適用して同じ term を得ることである。2 つの term s, t が与えられたとき、このような置換が存在するときにそれを求め、存在しない時には失敗するようなアルゴリズムが 1965 年に Robinson によって提案された [2]。

これの改良版が 1982 年に Martelli と Montanari によって提案された [1]。以下では Martelli と Montanari によって提案されたアルゴリズムについて紹介する。Robinson のアルゴリズム [2] は与えられる term は 2 つであるが、Martelli と Montanari のアルゴリズム [1] は term が 2 つ以上の場合に対応している。このアルゴリズムは、term 間の等式の集合

$$E = \{s_1 = t_1, \ldots, s_n = t_n\}$$

が与えられたとき、E の単一化ができないとき失敗し、E の単一化ができるとき、等式

$$\{X_1 = r_1, \ldots, X_m = r_m\}$$

を返す。ここで、$X_1, \ldots, X_m$ は term $r_1, \ldots, r_m$ 中には現れない、互いに異なる変数である。単一化できる場合に得られる等式

$$\{X_1 = r_1, \ldots, X_m = r_m\}$$

を、置換

$$[r_1/X_1, \ldots, r_m/X_m]$$

とみなすと、これが E の最も一般的な単一化子 (most general unifier) になっている。

アルゴリズム 1 (Martelli と Montanari によって提案されたアルゴリズム)

1. 等式集合 E から一つの等式を任意に選択する。$X = t$ の形の等式で、他の等式中に X が含まれていない場合はそれは選択しない。もし E 中の等式が

$$\{X_1 = r_1, \ldots, X_m = r_m\}$$

の形で、$X_1, \ldots, X_m$ が $r_1, \ldots, r_m$ の中に含まれていない場合、成功して終了する。

2. 選択された等式の形で以下のように場合分けする。

   (a) $f(l_1, \ldots, l_k) = f(m_1, \ldots, m_k)$ の場合
   この等式を集合 E から取り除き、等式 $l_1 = m_1, \ldots, l_k = m_k$ を集合 E に追加する。

   (b) $f(l_1, \ldots, l_k) = g(m_1, \ldots, m_k)$ の場合
   f と g が異なるとき、失敗して終了する。

   (c) $X = X$ の場合
   この等式を集合 E から削除する。

   (d) $X = t$ の場合
   term t が変数 X を含まず、この変数 X が集合 E の中の別の等式に含まれているなら、置換 $[t/X]$ を集合 E の中の他のすべての等式に適用する。

   (e) $X = t$ の場合
   term t が変数 X を含む場合、失敗して終了する。(これを occurs check という。)

   (f) $t = X$ の場合
   t が変数でない場合、この等式を集合 E から取り除き、等式 $X = t$ を集合 E に追加する(つまり、左辺と右辺を入れ替えるということ)。

3. 1 へ戻る。

上記アルゴリズムが成功して終了したとき、集合 E は

$$\{X_1 = r_1, \ldots, X_m = r_m\}$$

の形をしており、上記で述べたようにこれは

$$[r_1/X_1, \ldots, r_m/X_m]$$

という置換であると見なす。これが E の最も一般的な単一化子 (most general unifier) になっている。

補足 1 上記アルゴリズムにおいて、term を $f(l_1, \ldots, l_k)$ の形で書いているが、これは $k = 0$ の場合に、引数のない定数 f を表しているとする。(つまり、term の中間ノードは $k \leq 1$、leaf ノードは $k = 0$ で表している。)

補足 2 置換 (substitution) については、第 3 回の補足資料にある、論理式に対する置換と同様に定義すればよい。第 3 回で用いた論理式は、ある特定の形をしてい

る term であり、例えば、$1 + 2 > 2$ のような論理式は、$>(+(1, 2), 2)$ のような形に書けばよく、f や g を $>$ や $+$ にしたものである。つまり、今回提示した単一化アルゴリズムは、論理式に限らず、さまざまな term に対応できる汎用的なアルゴリズムである。

補足 3 第 3 回の補足資料では、変数は小文字で書いているが、今回は、Prolog が変数は大文字から始まるため、それに合わせて変数を大文字で書いている。

参考文献

[1] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, Vol. 4, No. 2, pp. 258–282, April 1982.

[2] John A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, Vol. 12, No. 1, pp. 23–41, 1965.