平成 19 年度「コンパイラ」定期試験の解説

国島丈生

2007-07-19

1 正則表現と有限オートマトン

1. (a) $(01\,|\,10)(0\,|\,1)^*$

   目立った間違いはありませんでした。

   (b) $-(a|\cdots|z|A|\cdots|Z|0|\cdots|9)\,|\,--(a|\cdots|z)(a|\cdots|z)^+$

   もしくは $-[a-zA-Z0-9]\,|\,--[a-z][a-z]^+$

   後半は「複数個」が 1 個以上なのか 2 個以上なのか曖昧だったので、1 個以上としたもの、つまり $--[a-z]^+$ でも正解としました。この問題で目立ったのは、正則表現の略記法 [] の誤用です。$[a-z]$ は $(a|\cdots|z)$ の略記法ですが、$a,\cdots,z$ は記号でなければなりません。[] 内に正則表現を入れることも、[] を入れ子にすることも、$[a-z|A-Z|0-9]$ のように | 演算子を用いることもできません。今回は明らかな誤用を除いて大幅な減点はしませんでしたが、まず略記法を使わない書き方ができるようにしておいて下さい。略記法を使う場合は、定義を充分に理解してからにして下さい。

2. 図 1 参照。この問題の誤答で目立ったのは、オートマトンの枝のラベルに正則表現を書いているものです（01, $[a-z]$ など）。元々の（決定性）有限オートマトンの定義では、遷移関数は状態 1 つと入力記号 1 つから状態 1 つを返します。つまり、枝のラベルは入力記号 1 つです（$a,\cdots,z$ は a から z までの記号という意味で、枝を複数本書くのをサボるために用いています）。枝に正則表現を書けるように拡張しても計算能力は変わらない（受理される言語クラスはやはり正則言語）のですが、どのような言語を受理するかは定義が必要です。計算能力が変わらないこと、有限オートマトンの厳密な定義を講義では述べていないことから、減点はしませんでしたが、本来は減点すべきものです。

3. $0^*\,|\,0^*1^+$, もしくは 0^*1^*

   状態 0 で受理される言語は 0^*、状態 1 で受理される言語は 0^*1^+ であることから、上のような解答になります。この問題で目立った誤答は、状態 3 に到達する言語（$0^*1^+0(0|1)^*$ など）を解答としたものです。有限オートマトンは、入力を読み切ったとき受理状態（状態遷移図の二重丸の状態）にいる場合、その入力を受理します。最後の状態と受理状態は異なりますので、注意して下さい。

2 文脈自由文法

1. 図 2 参照。目立った誤答はありませんでした。

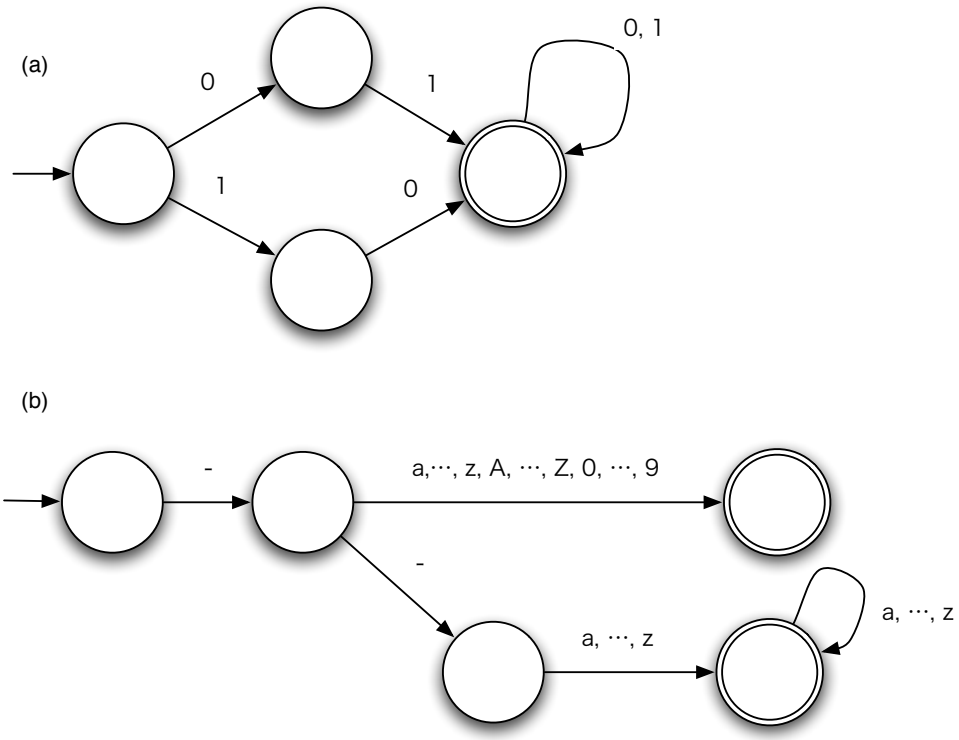

図 1　問 1-2 の解答例

2.

$$S \Rightarrow SbS \Rightarrow abS \Rightarrow abScS \Rightarrow abacS \Rightarrow abacb$$
$$S \Rightarrow ScS \Rightarrow SbScS \Rightarrow abScS \Rightarrow abacS \Rightarrow abaca$$

導出の複数ステップをまとめてしまう（$abScS \Rightarrow abaca$ など）と、最左導出でない導出も含まれてしまうので、解答としてはよくありません。面倒でも、上のようにすべてのステップを書くべきです。また、導出は（生成規則と区別するという意味で）二重矢印を使うべきです。

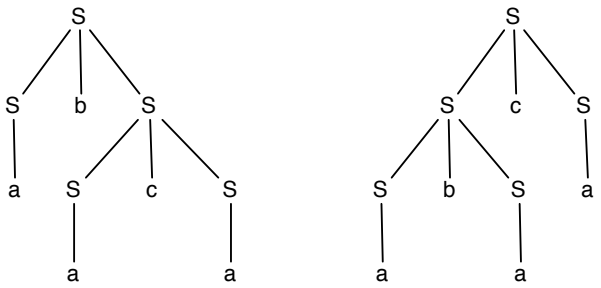

図 2　問 2-1 の解答例

3 LL(1) 文法

1.

$$
\begin{aligned}
FIRST(S) &= \{a, b, c, d\} \\
FIRST(A) &= \{a, \epsilon\} \\
FIRST(B) &= \{a, b, c\} \\
FIRST(C) &= \{a, b, c\} \\
FIRST(D) &= \{a, b, c, d\} \\
FOLLOW(S) &= \{\$\} \\
FOLLOW(A) &= \{a, b, c, \$\} \\
FOLLOW(B) &= \{a, \$\} \\
FOLLOW(C) &= \{a, \$\} \\
FOLLOW(D) &= \{a, \$\}
\end{aligned}
$$

2. 前問より

$$
\begin{aligned}
DIRECTOR(A, aA) &= FIRST(aA) = \{a\} \\
DIRECTOR(A, \epsilon) &= (FIRST(\epsilon) - \{\epsilon\}) \cup FOLLOW(A) \\
&= \{a, b, c, \$\}
\end{aligned}
$$

となるので、LL(1) 文法ではない。

LL(1) 文法はやはり分かりにくいようで、完全な正答はごくわずかでした。

$FIRST()$ に関する誤答で最も多かったのは、ϵ の扱いを間違ったものでした。例えば $FIRST(B) = \{a, b, \epsilon\}$ となってしまっているものです。$B \to AcD$ について、$FIRST(A)$ に加えるのは以下の 2 つです。

- 右辺の先頭の記号 A から導出される記号列で先頭に出現する終端記号。すなわち $FIRST(A) - \{\epsilon\} = \{a\}$。
- 右辺の先頭の記号 A から ϵ が導出される場合、この部分は消えてしまうことを意味する。したがって、2 文字目の記号 c（正確には $FIRST(c)$）が $FIRST(A)$ に追加される。

$FOLLOW()$ に関しては誤答パターンが様々で、はっきりした間違いの傾向が見つかりませんでした。おそらく計算方法が分かりづらく、整理し切れていないのだと思われます。以下に、この問題を考えているときに思いついた $FOLLOW()$ 計算の整理方法を示します。まだ十分に検討し切れていないので、バグがあるかもしれない点を留意して、読んで下さい[*1]。

$FOLLOW()$ の値が変化するのは、次の 3 通りの場合しかありません。

- 開始記号 S について、無条件に \$ が加えられる。
- $A \to \alpha B \beta$ の形の規則について、$FIRST(\beta)$ が $FOLLOW(B)$ に加えられる。(β から導出される記号列の先頭の文字は B の直後に現れるから)

[*1] この部分については、分かりやすさその他、意見や疑問などあればぜひ t.kunishi@gmail.com までご連絡下さい。

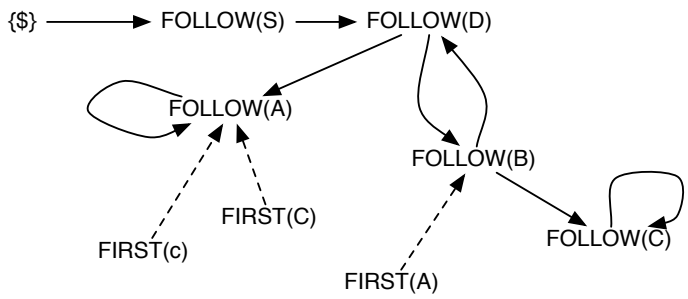

図 3　FOLLOW の値の伝搬を表すグラフ

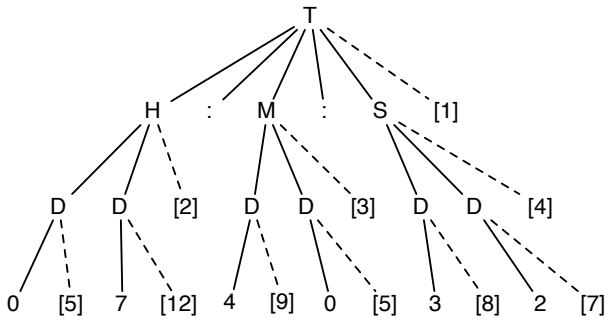

図 4　問 4-1 の答

- $A \to \alpha B$、もしくは $A \to \alpha B\beta$ で β から ϵ が導出される（β が消えてしまう）場合について、$FOLLOW(A)$ が $FOLLOW(B)$ に加えられる。（A の直後に現れる文字は B の直後にも現れるから）

そこで、生成規則を見ながら、値が伝搬していく様子を有向グラフにしてみます。例えば $B \to bAC$ から、$FIRST(C) - \{\epsilon\}$ が $FOLLOW(A)$ に、$FOLLOW(B)$ が $FOLLOW(C)$ にそれぞれ伝搬することが分かりますので、枝 $FIRST(C) \to FOLLOW(A)$、$FOLLOW(B) \to FOLLOW(C)$ をグラフに加えます。$\{\epsilon\}$ を引かなければならない場合は枝を破線にしておきます。すると図 3 のようなグラフが得られます。

このグラフにおいて、{$}、$FIRST()$ を、到達可能な節点（$FOLLOW$）に加えていきます。この結果得られた記号集合が $FOLLOW$ になるわけです。

このグラフは $FOLLOW$ の検算にも用いることができます。例えば $FOLLOW(D) \to FOLLOW(A)$ という枝があるということは、$FOLLOW(D)$ の要素はすべて $FOLLOW(A)$ の要素でもある、という意味ですから、結局 $FOLLOW(D) \subseteq FOLLOW(A)$ という関係が成り立つことが分かります。今回の問題の答が、このグラフから得られる集合の包含関係を満たしていることを確かめてみて下さい。

4　翻訳スキーム

1. 図 4。$T.val$ の値は 27632。
2. 文字列を時刻とみなしたとき、00:00:00 からの秒数を求める翻訳スキーム。

目立った誤答はありませんでした。

5 実行時環境

3
2
3

b() および main() で参照される変数 x は大域変数、c() で参照される変数 x は局所変数です。したがって、大域変数 x の値は b() 中で 3 と変更された後 printf() で参照されるので、1 番目と 3 番目は 3 です。一方 c() 中の局所変数 x は c() の外部には影響を及ぼしません。

誤答で多かったのは、最後が 1 や 2 になっているものでした。局所変数と大域変数の区別、変数がメモリ上のどこに確保されいつ解放されるのか、など、変数や関数とメモリ管理との関連はプログラミング言語の動作を理解する上で極めて重要です。誤解していた人は改めて整理をした上で、一度実際にプログラムを動かしてみて下さい。