back track つき構文解析器問題の回答・考察

soundscope

提出年月日：2022 年 1 月 28 日

以下の問題の回答である.
`https://github.com/soundscope/enshud/blob/main/question.pdf`

1　原因とマッチする集合の考察

以下はバグが発生する原因で, ある種等価な言い換えである.

- 一度 true と判定されてしまうと false の可能性を一切考慮せず, そのまま back track してしまっている
- 葉の可能性をすべて評価 (展開) しなれけばならないができていない
- 各メソッド (関数) は true / false ではなく状態集合を返す必要がある

これを図示し, マッチ (認識) する集合を考えると,
図 1 の a の右肩の o はそこまで入力が消費できることを指し, x は入力が終端に到達し false が返ったことを表している.
よって図から a^{2^n} $(n > 0)$ にマッチすると考えられる.

2　参考

ほとんど参考にしていないが, 私以外の人間の考察を発見したので示しておく.
`https://compilers.iecc.com/comparch/article/06-11-099`

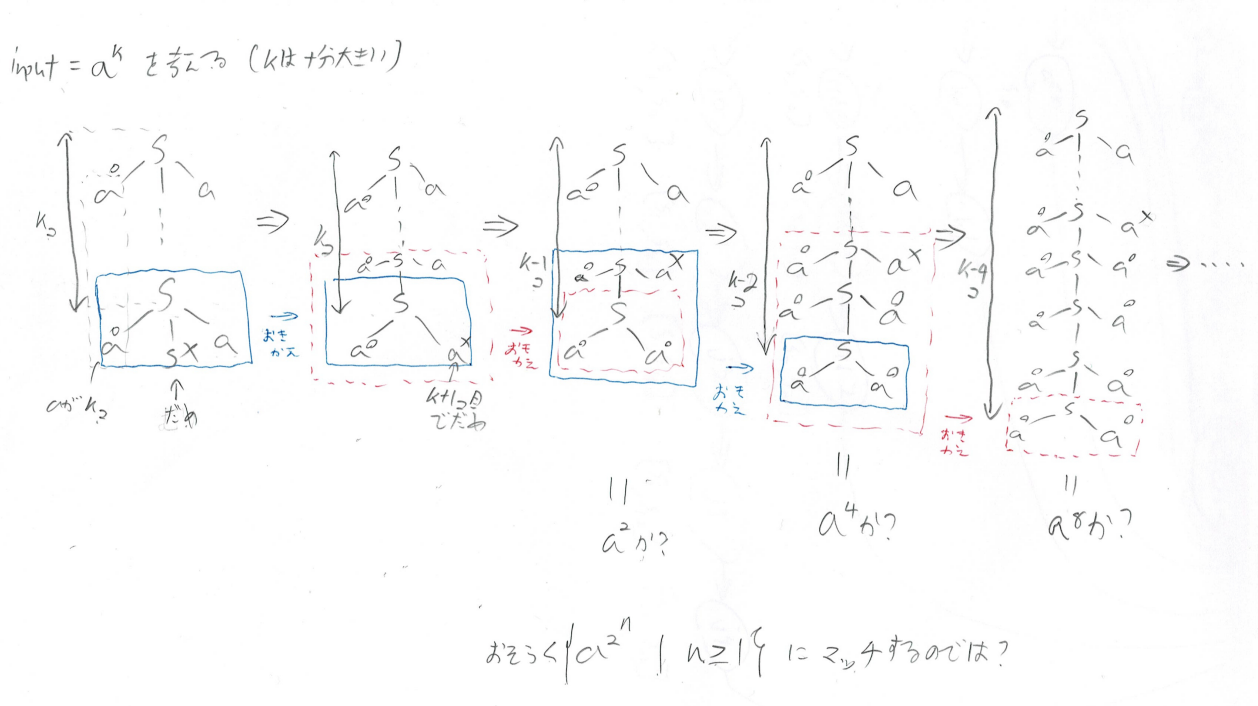

図 1: 関数の呼び出しを図示

3 解決方法

この問題には parser generator を作成する場合などにおいて致命的なバグを引き起こす潜在的な可能性がある.

以下に考えられる解決策を示す.

- parser をより強力なもの (考えられる導出をすべて展開するもの, 導出しうる”集合”を処理できるもの) にする.
- parser をより強力なもの (入力の終わりからチェックできるもの) にする.
- 文法が簡単になるように (first 集合が一致しない形に) 変換する.

しかし, 最も上の解決策での実装は (与えられた文法にもよるが,) べき乗の計算量となってしまうと考えている. これはバックトラック型の降下構文解析器の大きな欠点だと思う.