PCPの文字列制約を介した不可能性判定と, ParikhAutomaton の高速な最小限の計算

大森章裕

目次

Post's Correspondence Problem

- アルファベット集合 Σ

サイズ sのPCPインスタンスとは, s 個の Σ 上の word のペア (タイル)

$$((g_1, h_1), \ldots, (g_s, h_s))$$

このインスタンスの解とは, 添字の列 $(i_1, i_2, \ldots i_n) \in \{1, \ldots, s\}^*$

$$g_{i_1} g_{i_2} \ldots g_{i_n} = h_{i_1} h_{i_2} \ldots h_{i_n}$$

となるもの

Post's Correspondence Problem

ex. サイズ3のPCPインスタンス

$$\begin{bmatrix} g \\ h \end{bmatrix} = \begin{bmatrix} 100 & 0 & 1 \\ 1 & 100 & 00 \end{bmatrix}$$

解

$$”1311322” \in \{1, 2, 3\}^*$$

実際並べてみると...

$$\begin{bmatrix} 100 & 1 & 100 & 100 & 1 & 0 & 0 \\ 1 & 00 & 1 & 1 & 00 & 100 & 100 \end{bmatrix}$$

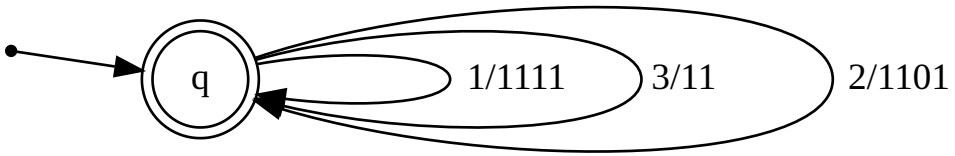
文字列制約問題としての定式化

- PCPのインスタンス $P = (T_1, T_2)$
- T_1, T_2 : 上段、下段に対応するトランスデューサ
- 制約: $T_1(x) = T_2(x)$

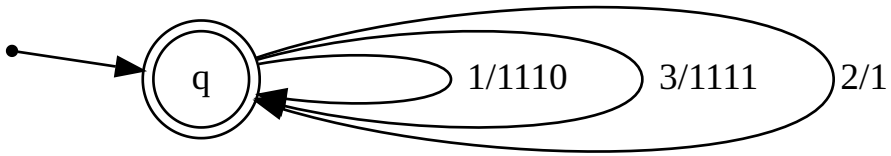
ex.

$$\begin{bmatrix} 1111 & 1101 & 11 \\ 1110 & 1 & 1111 \end{bmatrix}$$

T_1



T_2



Post's Correspondence Problem

- 決定不能問題
- 実用性はあんまりない
  - 文字列制約の難しい部分問題としての位置付けを重視
  - このあと提案する手法が、PCPに対して有効→文字列制約に一般化した後でも期待できるね、という話
- 肯定的に解決する
  - 実際に解を見つける
  - 探索的手法が強い
- 否定的に解決する
  - 今回はこちらがメイン

PCP を否定的に解決する

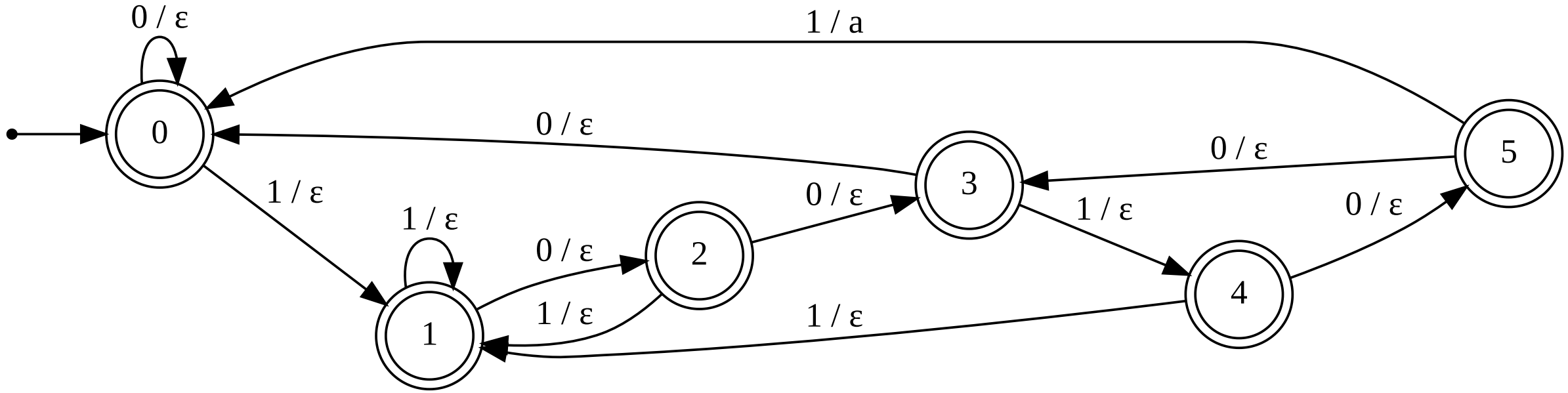
PCPの文字列制約 $\mathbf{T_1(x) = T_2(x)}$ を効率よく解ける形に緩和したい

案

- $Length(T_1(x)) = Length(T_2(x))$
- $Parikh(T_1(x)) = Parikh(T_2(x))$
- $Count_{100101}(T_1(x)) = Count_{100101}(T_2(x))$
  - $Count_{100101}(w)$ は, w 中の100101の出現回数
- $\mathbf{Parikh(W(T_1(x))) = Parikh(W(T_2(x)))}$ W: Transducer
  - Wをそれぞれ合成すれば、上から二番目の形

$Count_{100101}(T_1(x)) = Count_{100101}(T_2(x))$

下のトランスデューサーをWにすれば



$$\mathbf{Parikh(W(T_1(x))) = Parikh(W(T_2(x)))}$$

の形になる

$Parikh(T_1(x)) = Parikh(T_2(x))$

- ベクトル出力トランスデューサ $Parikh(T_1), Parikh(T_2)$

$S = (Q, \Sigma_{idx}, \Sigma_{pcp}, \Delta, q_0, q_f)$

- $Q = Q_1 \times Q_2$
- $\Delta \subset Q \times \Sigma_{idx} \times \mathbb{Z}^{\Sigma_{pcp}} \times Q$
- $\Delta = \{((p_1, p_2), a, \mathbf{v_1 - v_2}, (q_1, q_2)) | (p_1, a, v_1, q_1) \in \Delta_1, (p_2, a, v_2, q_2) \in \Delta_2\}$
  - 普通の直積構成で、出力するベクトルが差になっている

このSに,論理式 $\varphi = \forall a \in \Sigma_{pcp}, \text{'a'} = 0$ を課したParikh Automaton を考える

ParikhAutomaton S の解き方（これまで）

1. S の有向グラフとしての表現 G とする
2. 各辺 e に対して、その辺を通る回数を表す変数 y_e を用いて, G の開始頂点から受理頂点への路をちょうど全て捉える論理式ψを構成[1]
   - $\psi(y) \iff \exists t(G\text{の開始頂点から受理頂点への路}), \forall e, y_e = |t|_e$
3. $\Sigma_e y_e \cdot v_e = 0$ ($\text{辺}e\text{で出力するベクトル}v_e$) と, 2 の論理式の論理積をSMTソルバに解かせる

Z3が遅いので,改善したい. 混合整数計画問題として定式化して高速化

[1] 等号否定入り文字列制約のStreaming String Transducerを用いた充足可能性判定

オートマトンの受理する路を捉える論理式

各辺 e に対して、その辺を通る回数を表す変数 y_e を用いて, G の開始頂点から受理頂点への路をちょうど全て捉える論理式ψを構成

- euler condition (流量保存則)
  - これが満たされているyの, どの連結成分も一筆書き可能（対応する路が存在）
  - **最大流問題をLP定式化したときの制約部分**
- connectivity constraint
  - 連結成分が複数あっては困るので、それを制限する
  - **素直に定式化できない**
    - 線形計画は、論理式の**OR**が扱えないため
    - やるにしても、指数個の制約が必要

connectivity constraint を混合整数計画で扱う

案1 (辺cutの方法)

1. connectivity を課さずに解く
2. 指数個の制約のうち, 違反されたものだけ追加して incremental に解く

案2 (探索的方法)

1. connectivity を課さずに解く
2. 開始頂点と連結でない&流量が正の頂点 v があれば,
   - v への流量が正 (使うパターン)
   - v への流量が0 (使わないパターン)

   と場合分けして再帰的に解く

connectivity constraint を混合整数計画で扱う

案1 (辺cutの方法)

- **cons**: 線形緩和したときに弱い (混合整数計画ソルバは線形緩和した問題を利用するので,効率が悪くなる)

案2 (探索的方法)

- **cons**: 変数が増える

混合整数計画定式化

- 何らかの値の最小化,最大化もできる
  - Parikh Automaton の言語の最小元が取ってこれる等
  - 肯定的解決のための探索で枝狩りに下界として利用可能
- 線形緩和によって、速度と精度のトレードオフができる

SMT

- 扱える論理式の幅が広い
  - 混合整数計画はORが直接的には扱えない
  - 非線形な式を入れても扱えるソルバは多い(決定不能にはなるが)

実験

TBD

- 先行研究で未解決だったPCPインスタンスで結構な数のunsatを証明
  - トランスデューサWとして,色々なwについての$Count_w$の積を使用
- 小さいケースから大きいケースまでZ3より圧倒的に速い

今後

- equalityの緩和方法は, PCPの文字列制約に限らず一般の文字列制約問題に一般化することが可能なはず
  - unsat性に関してあまり研究が進んでいる様子はなさそう？
- 良いトランスデューサWとは何なのか？
- アルファベットが増えるとちょっとまずいかもしれない