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

福田大我

February 7, 2020

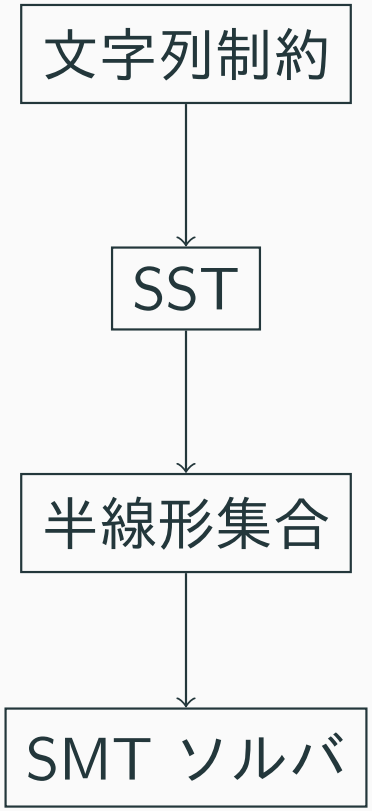
文字列制約

直線文字列制約

$$
\begin{aligned}
x_2 &= x_0 \cdot x_1 \\
x_3 &= x_2.replaceAll(aba, c) \\
x_0 &\in (a \cup b)^* \\
|x_2| &= 5
\end{aligned}
$$

$x_0 = aba, x1 = ba, x_2 = ababa, x3 = cba$ は解の一つとなる.

先行研究での充足可能性判定 [Zhu, Akama, Minamide, 2019]



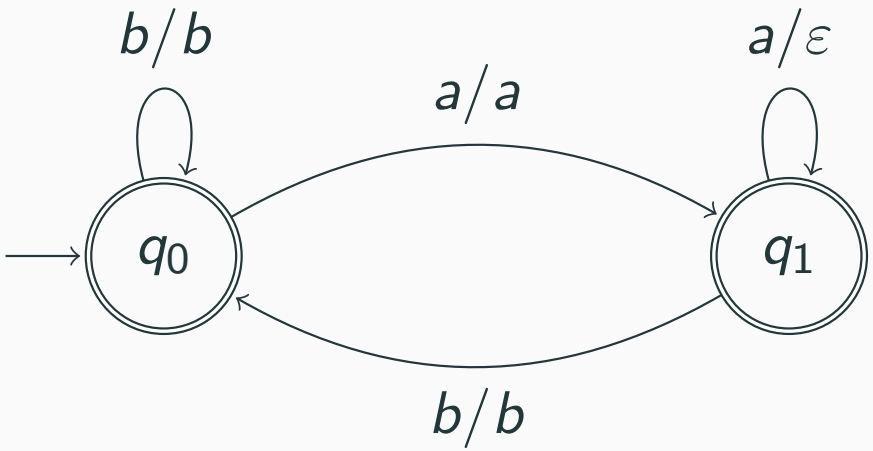
1. 文字列制約 φ から Streaming String Transducer S を構成.
2. S から $x_0, \ldots, x_n$ の長さを Parikh Image の半線形集合として与える.
3. 与えられた長さを用いて整数制約を解く.

本研究の成果

- Zhu による Streaming String Transducer を用いたソルバの等号否定 $x_i \neq x_j$ を含む文字列制約への拡張.
- SMT ソルバに与える論理式のサイズの大幅な改善.
  - トランスデューサの Parikh Image を Presburger Formula によって構成.
- 上記のソルバの Scala による実装.

トランスデューサ

受け取った文字列の連続した a を一つの a にまとめるトランスデューサの例



- 入力 w に対するトランスデューサ T の出力を $T(w)$ と書く.
- $$T(aaabb) = abb$$

直線文字列制約

$\{x_0, \ldots, x_n\}$: 文字列変数, T : 決定性トランスデューサ

基礎直線制約

$$\varphi_{sl} ::= (x_m = e_m) \wedge (x_{m+1} = e_{m+1}) \wedge \cdots \wedge (x_n = e_n)$$

$$e_i ::= w \mid x_j \mid x_j \cdot x_k \mid T(x_j) \quad (w \in \Sigma^*, j, k < i)$$

正規制約

$$\varphi_{reg} ::= \bigwedge (x_i \in R_i) \quad (R_i : \text{正規言語},\ 0 \leq i \leq n)$$

整数制約

$$\varphi_{int} ::= t = t \mid t < t \mid \varphi \wedge \varphi \mid \neg\varphi$$

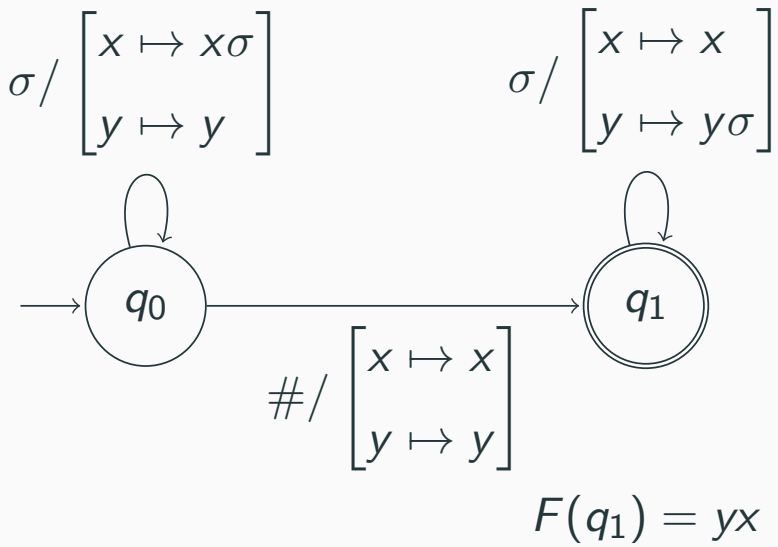
$$t ::= c \mid u \mid |x_i| \mid t + t \mid t - t \quad (c \in \mathbb{Z} : \text{定数}, u \in \mathbb{Z} : \text{変数})$$

制約

$$\varphi ::= \varphi_{sl} \wedge \varphi_{reg} \wedge \varphi_{int} \wedge \left(\bigwedge x_i \neq x_j\right)$$

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例

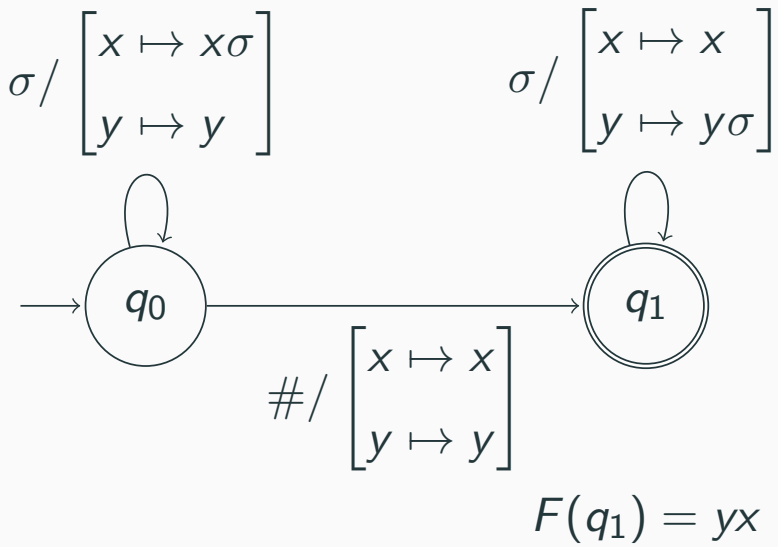


入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例

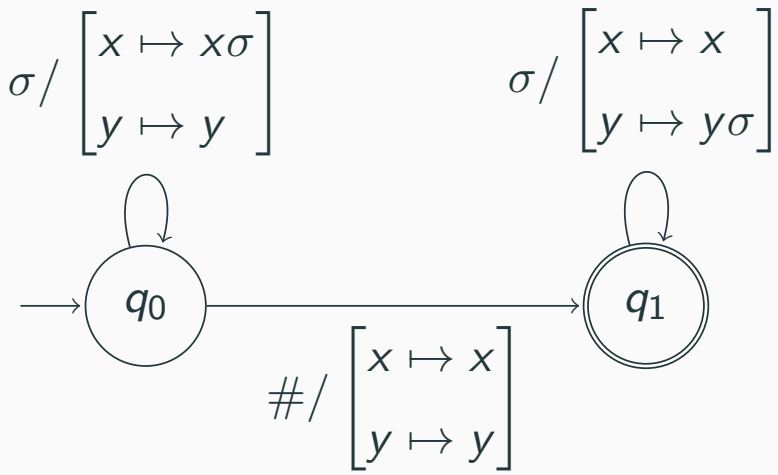


入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.
- $\langle q_0, [\varepsilon, \varepsilon]\rangle \xrightarrow{a} \langle q_0, [a, \varepsilon]\rangle$

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例

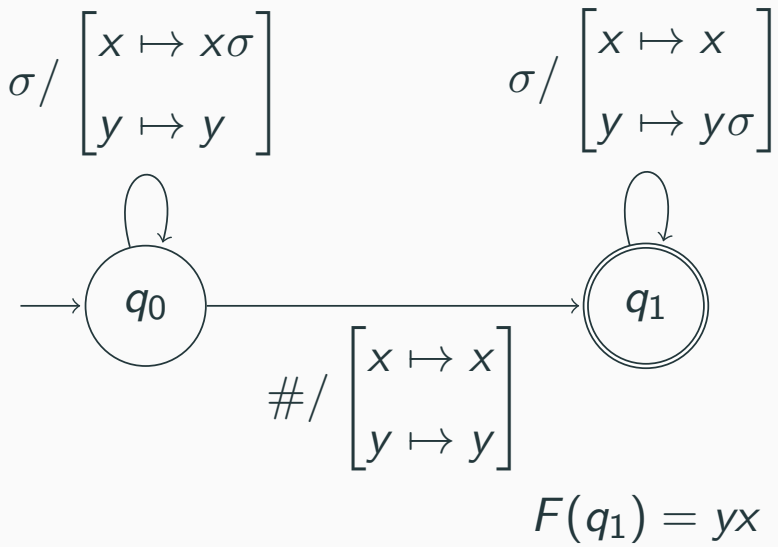


入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.
- $\langle q_0, [\varepsilon, \varepsilon]\rangle \xrightarrow{a} \langle q_0, [a, \varepsilon]\rangle$
- $\langle q_0, [a, \varepsilon]\rangle \xrightarrow{b} \langle q_0, [ab, \varepsilon]\rangle$

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例

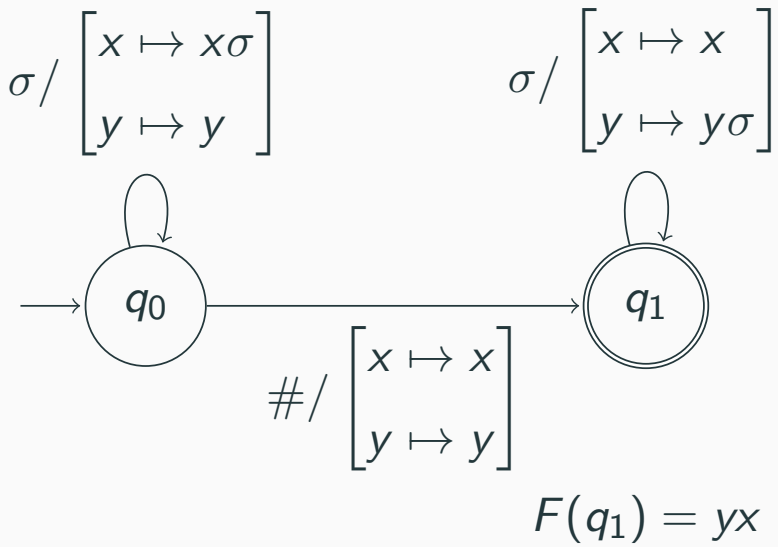


入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.
- $\langle q_0, [\varepsilon, \varepsilon]\rangle \xrightarrow{a} \langle q_0, [a, \varepsilon]\rangle$
- $\langle q_0, [a, \varepsilon]\rangle \xrightarrow{b} \langle q_0, [ab, \varepsilon]\rangle$
- $\langle q_0, [ab, \varepsilon]\rangle \xrightarrow{\#} \langle q_1, [ab, \varepsilon]\rangle$

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例

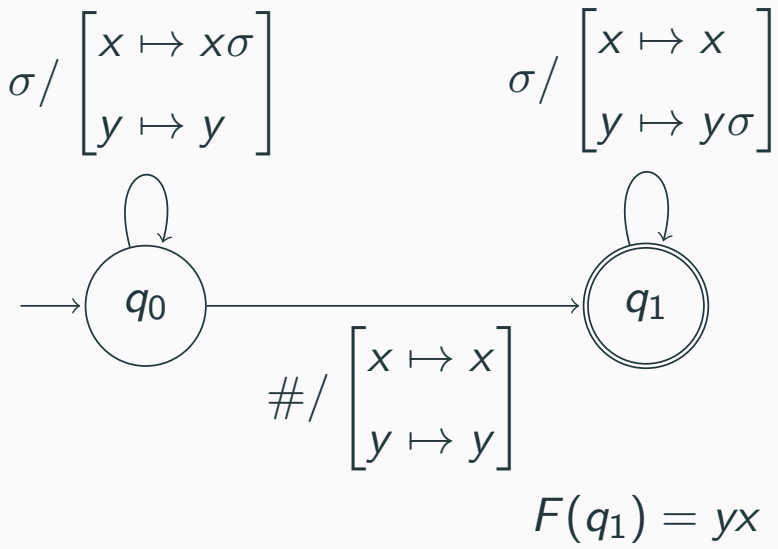


入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.
- $\langle q_0, [\varepsilon, \varepsilon]\rangle \xrightarrow{a} \langle q_0, [a, \varepsilon]\rangle$
- $\langle q_0, [a, \varepsilon]\rangle \xrightarrow{b} \langle q_0, [ab, \varepsilon]\rangle$
- $\langle q_0, [ab, \varepsilon]\rangle \xrightarrow{\#} \langle q_1, [ab, \varepsilon]\rangle$
- $\langle q_1, [ab, \varepsilon]\rangle \xrightarrow{b} \langle q_1, [ab, b]\rangle$

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例

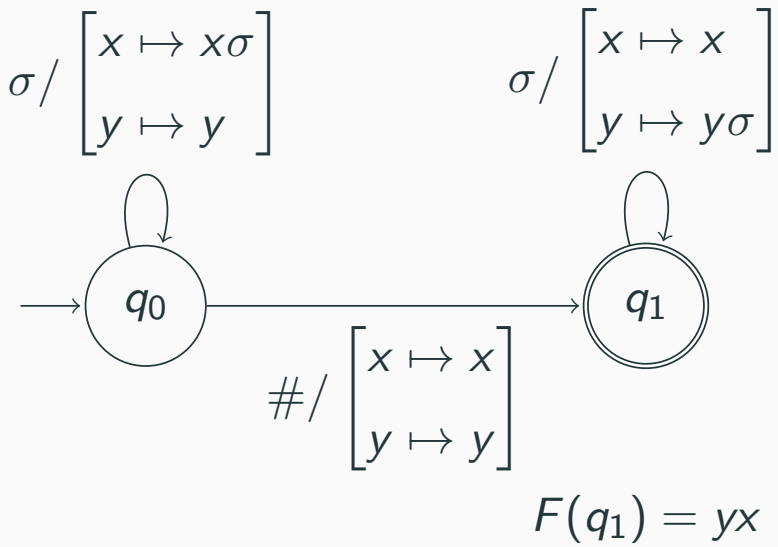


入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.
- $\langle q_0, [\varepsilon, \varepsilon]\rangle \xrightarrow{a} \langle q_0, [a, \varepsilon]\rangle$
- $\langle q_0, [a, \varepsilon]\rangle \xrightarrow{b} \langle q_0, [ab, \varepsilon]\rangle$
- $\langle q_0, [ab, \varepsilon]\rangle \xrightarrow{\#} \langle q_1, [ab, \varepsilon]\rangle$
- $\langle q_1, [ab, \varepsilon]\rangle \xrightarrow{b} \langle q_1, [ab, b]\rangle$
- $\langle q_1, [ab, b]\rangle \xrightarrow{a} \langle q_1, [ab, ba]\rangle$

Streaming String Transducer (SST)

入力 $w_0\#w_1$ で出力 w_1w_0 の **SST** の例



入力 $ab\#ba$ に対する動作

- $\langle q, [x, y]\rangle = \langle q_0, [\varepsilon, \varepsilon]\rangle$ から始める.
- $\langle q_0, [\varepsilon, \varepsilon]\rangle \xrightarrow{a} \langle q_0, [a, \varepsilon]\rangle$
- $\langle q_0, [a, \varepsilon]\rangle \xrightarrow{b} \langle q_0, [ab, \varepsilon]\rangle$
- $\langle q_0, [ab, \varepsilon]\rangle \xrightarrow{\#} \langle q_1, [ab, \varepsilon]\rangle$
- $\langle q_1, [ab, \varepsilon]\rangle \xrightarrow{b} \langle q_1, [ab, b]\rangle$
- $\langle q_1, [ab, b]\rangle \xrightarrow{a} \langle q_1, [ab, ba]\rangle$
- $F(q_1) = yx$ であるから出力は $baab$ となる.

Zhu によるソルバ

Zhu によるソルバでは
$\varphi = ((x_{m+1} = e_{m+1}) \wedge \cdots \wedge (x_n = e_n)) \wedge \varphi_{reg}$ に対し,
以下を満たすような決定性 SST S を構成する.

$$[x_0 \mapsto w_0, \ldots, x_n \mapsto w_n] \models \varphi$$
$$\Leftrightarrow S(w_0\# \cdots \# w_m\#) = w_0\# \cdots \# w_n\#$$

- S は制約の右辺にしか出てこない文字列変数の割り当て $w_0\# \cdots \# w_m\#$ を受け取り, 制約 φ を満たす変数の割り当て $w_0\# \cdots \# w_n\#$ を出力する.
- この SST の定義域が空でなければ制約を満たす変数割り当てが存在する.

等号否定 $x_i \neq x_j$ の充足可能性判定

- $\varphi_{sl} \wedge \varphi_{reg}$ から構成した SST の出力 $x_0\# \cdots \# x_n\#$ を x_i, x_j に修正した決定性 SST S_i, S_j が得られる.
- すなわち $\varphi = ((x_{m+1} = e_{m+1}) \wedge \cdots \wedge (x_n = e_n)) \wedge \varphi_{reg}$ に対し,

$$S_i(w_0\# \cdots \# w_m\#) = w_i$$
$$S_j(w_0\# \cdots \# w_m\#) = w_j$$

を満たす SST が得られる.

- $$x_i \neq x_j \Leftrightarrow \exists w \in (\Sigma^*\#)^m.\ S_i(w) \neq S_j(w)$$

であるから, 等号否定 $x_i \neq x_j$ の充足可能性判定を関数的 SST の非等価性判定に帰着した.

関数的 SST の非等価性判定 [Alur, 2011]

- 関数的 SST とは任意の入力に対し, 出力が一意に定まるような SST である.
- 関数的 SST S, S' が非等価であるとき, ある入力 w に対して以下の二条件のいずれかを満たす.
  - i $S(w)$ と $S'(w)$ の出力文字列の長さが異なる
  - ii $S(w)$ と $S'(w)$ の出力文字列の p 文字目がそれぞれ相異なる文字になる
- i については整数制約 ($|x_i| \neq |x_j|$) として解くことができる.
- ii については部分文字列長の一致可能性問題として解くことができる.

また, $\{w \mid S(w) \neq S'(w)\}$ が半線形集合であり, この方法で解が構成できることを示した.

実験

本研究では, 以下を Scala によって実装した.

- SST を用いたソルバの等号否定 $x_i \neq x_j$ のサポート
- SMT ソルバに与える論理式のサイズの大幅な改善.

実験

$$y = \mathit{reverse}(x)$$
$$z = y.\mathit{replaceAll}(aa, x)$$
$$z1 = \mathit{reverse}(z)$$
$$x1 = x.\mathit{replaceAll}(aa, x)$$
$$x1 \neq z1$$
$$|x| = 3$$

結果 sat

$$x = aaa, y = aaa, z = xa,$$
$$x1 = xa, z1 = ax$$

実行時間: 2.274s

実験

$$
\begin{aligned}
y_0 &= x_0.replaceAll(< sc >, \varepsilon) \\
y_1 &= x_1.replaceAll(< sc >, \varepsilon) \\
0 &< |x_0| \\
0 &< |x_1| \\
|y_0| &= |y_1| \\
|y_0| &= 10
\end{aligned}
$$

結果 sat

$x_0 = sc << sc >< sc < sc >< sc < sc > s, x_1 = s <<<<<<<<< s,$

$y_0 = sc << sc < scs, y_1 = s <<<<<<<<< s$

実行時間: 8.663s

まとめ

本研究の成果

- SST を用いたソルバの等号否定 $x_i \neq x_j$ を含むような直線文字列制約への拡張.
- トランスデューサの Parikh Image を表す Presburger Formula の構成.
- 上記のソルバの Scala による実装.

部分文字列長の一致可能性判定

SST の非等価性

i $S(w)$ と $S'(w)$ の出力文字列の長さが異なる

ii $S(w)$ と $S'(w)$ の出力文字列の p 文字目がそれぞれ相異なる文字になる

ii について

$a, b \in \Sigma, a \neq b$ とする.

- $S(w)$ の出力で a の prefix となる部分文字列を出力する非決定性 SST S^a を構成する. 同様にして S'^b を構成.
- S^a と S'^b の文字列長の一致可能性問題として解く.

トランスデューサの Parikh Image を表す Presburger Formula

トランスデューサ $T = \langle \Sigma, \Gamma, Q, \delta, Q_0, F \rangle$ に対して, T を実際に動作させた時の run $p \xrightarrow[t]{w/w'} r$ に対応する Presburger Formula を構成.

- (Euler Condition) 任意の状態 q について q に入る回数と出る回数の差が p では -1, r では 1, それ以外では 0 になる.
- (Connectivity) t で使用されている状態が $q_0 \in Q_0$ から t で使用されている遷移のみを用いて到達可能.
- Parikh Image を表す.
- $p \in Q_0, r \in F$ である.

Presburger Formula

$$\phi ::= t = t \mid t < t \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x.\ \phi$$

$$t ::= 0 \mid 1 \mid u \mid t + t \mid t - t$$