プレスバーガー算術の決定可能性

有限オートマトンを使う証明の形式化

坂口和彦

筑波大学 コンピュータサイエンス専攻

2015/9/16 TPP2015

概要

- 自然数とその加法に関する一階の理論**プレスバーガー算術**は決定可能
- 量化子除去 (QE) を使う決定手続きが良く知られている
- 本発表では有限オートマトンを使うプレスバーガー算術の決定手続き [Boudet and Comon, 1996] を紹介し、その Coq での形式化について述べる

概要

- 自然数とその加法に関する一階の理論**プレスバーガー算術**は決定可能
- 量化子除去 (QE) を使う決定手続きが良く知られている
- 本発表では有限オートマトンを使うプレスバーガー算術の決定手続き [Boudet and Comon, 1996] を紹介し、その Coq での形式化について述べる

仮定する知識:

- 正規言語に関する学部相当の知識
  - 決定性有限オートマトン (DFA)
    非決定性有限オートマトン (NFA) の定義
  - 有限オートマトンの直積構成、サブセット構成

プレスバーガー算術

自然数とその加法に関する一階の理論:

- 定数記号 $0, 1$
- 関数記号 $+$
- 述語記号 $\leq$

プレスバーガー算術

自然数とその加法に関する一階の理論:

- 定数記号 $0, 1$
- 関数記号 $+$
- 述語記号 $\leq$

表現できるもの:

- 任意の自然数
- 定数倍
  - 偶奇

表現できないもの:

- 乗算
- 素数

プレスバーガー算術は決定可能

Theorem 1.

与えられたプレスバーガー算術の文の真偽を判定するアルゴリズムが存在する

証明の方針

- プレスバーガー算術の論理式を有限オートマトンに変換する
  - オートマトンが入力として取る語は自由変数への自然数の割り当てに対応する
- 論理式の真偽と変換後の有限オートマトンの受理非受理が一致することを示す

$\mathbb{N}^n$ を 2^n の語にエンコードする

自然数を一番下の桁が最初の文字になるように二進数で表す
(least significant bit first)

Example 2.

変数 (x_1, x_2, x_3) への自然数の割当て $(5, 10, 6)$ を以下の 2^3 の語で表す:

$$\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$\mathbb{N}^n$ を 2^n の語にエンコードする

自然数を一番下の桁が最初の文字になるように二進数で表す
(least significant bit first)

Example 2.

変数 (x_1, x_2, x_3) への自然数の割当て $(5, 10, 6)$ を以下の 2^3 の語で表す:

$$\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \rightarrow 1 + 4$$

$\mathbb{N}^n$ を 2^n の語にエンコードする

自然数を一番下の桁が最初の文字になるように二進数で表す
(least significant bit first)

Example 2.

変数 (x_1, x_2, x_3) への自然数の割当て $(5, 10, 6)$ を以下の 2^3 の語で表す:

$$\begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{array}{l} \\ \to 1 + 4 \\ \to 2 + 8 \end{array}$$

$\mathbb{N}^n$ を 2^n の語にエンコードする

自然数を一番下の桁が最初の文字になるように二進数で表す
(least significant bit first)

Example 2.

変数 (x_1, x_2, x_3) への自然数の割当て $(5, 10, 6)$ を以下の 2^3 の語で表す:

$$\begin{array}{cccccc} x_1 & \begin{bmatrix}1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\1\end{bmatrix} & \begin{bmatrix}1\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \to 1+4 \\ x_2 & & & & & \to 2+8 \\ x_3 & & & & & \to 2+4 \end{array}$$

$\mathbb{N}^n$ を 2^n の語にエンコードする

自然数を一番下の桁が最初の文字になるように二進数で表す
(least significant bit first)

Example 2.

変数 (x_1, x_2, x_3) への自然数の割当て $(5, 10, 6)$ を以下の 2^3 の語で表す:

$$\begin{array}{cccccc} x_1 & \begin{bmatrix}1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}1\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\1\end{bmatrix} & \begin{bmatrix}0\\0\\0\end{bmatrix} & \to 1+4 \\ x_2 & & & & & & \to 2+8 \\ x_3 & & & & & & \to 2+4 \end{array}$$

$\mathbb{N}^n$ を 2^n の語にエンコードする

自然数を一番下の桁が最初の文字になるように二進数で表す
(least significant bit first)

Example 2.

変数 (x_1, x_2, x_3) への自然数の割当て $(5, 10, 6)$ を以下の 2^3 の語で表す:

$$\begin{array}{cccccccc} x_1 & \begin{bmatrix}1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\end{bmatrix} & \begin{bmatrix}1\\0\\1\end{bmatrix} & \begin{bmatrix}0\\1\\1\end{bmatrix} & \begin{bmatrix}0\\0\\0\end{bmatrix} & \begin{bmatrix}0\\0\\0\end{bmatrix} & \to 1+4 \\ x_2 & & & & & & & \to 2+8 \\ x_3 & & & & & & & \to 2+4 \end{array}$$

原始命題に対応するDFAの構成 - 1

- 自由変数 $x_1, \ldots, x_n$ を含む命題 $t_1 \leq t_2$ は
  $a_1 x_1 + \cdots + a_n x_n \leq b$ という形の同値な命題に変形できる

原始命題に対応するDFAの構成 - 1

- 自由変数 $x_1, \ldots, x_n$ を含む命題 $t_1 \leq t_2$ は
  $a_1 x_1 + \cdots + a_n x_n \leq b$ という形の同値な命題に変形できる
  - ただしここでの $a_1, \ldots, a_n, b$ は整数

原始命題に対応するDFAの構成 - 1

- 自由変数 $x_1, \ldots, x_n$ を含む命題 $t_1 \leq t_2$ は
  $a_1x_1 + \cdots + a_nx_n \leq b$ という形の同値な命題に変形できる
  - ただしここでの $a_1, \ldots, a_n, b$ は整数
- 任意の i について $x_i = 2x'_i + c_i$ $(c_i \in 2)$ と仮定すると

$$
\begin{aligned}
& a_1x_1 + \cdots + a_nx_n \leq b \\
\Leftrightarrow\ & a_1(2x'_1 + c_1) + \cdots + a_n(2x'_n + c_n) \leq b \\
\Leftrightarrow\ & a_1x'_1 + \cdots + a_nx'_n \leq \left\lfloor \frac{b - a_1c_1 - \cdots - a_nc_n}{2} \right\rfloor
\end{aligned}
$$

原始命題に対応するDFAの構成 - 1

- 自由変数 $x_1, \ldots, x_n$ を含む命題 $t_1 \leq t_2$ は $a_1x_1 + \cdots + a_nx_n \leq b$ という形の同値な命題に変形できる
  - ただしここでの $a_1, \ldots, a_n, b$ は整数
- 任意の i について $x_i = 2x'_i + c_i$ $(c_i \in 2)$ と仮定すると

$$
\begin{aligned}
& a_1x_1 + \cdots + a_nx_n \leq b \\
\Leftrightarrow\ & a_1(2x'_1 + c_1) + \cdots + a_n(2x'_n + c_n) \leq b \\
\Leftrightarrow\ & a_1x'_1 + \cdots + a_nx'_n \leq \left\lfloor \frac{b - a_1c_1 - \cdots - a_nc_n}{2} \right\rfloor
\end{aligned}
$$

- $x_1, \ldots, x_n$ と $x'_1, \ldots, x'_n$ に対応する語を w, w' $(w \neq \varepsilon)$ とすると $w = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$ w' が成り立つ

原始命題に対応するDFAの構成 - 1

- 自由変数 $x_1, \ldots, x_n$ を含む命題 $t_1 \leq t_2$ は
  $a_1x_1 + \cdots + a_nx_n \leq b$ という形の同値な命題に変形できる
  - ただしここでの $a_1, \ldots, a_n, b$ は整数
- 任意の i について $x_i = 2x'_i + c_i$ $(c_i \in 2)$ と仮定すると

$$a_1x_1 + \cdots + a_nx_n \leq b$$
$$\Leftrightarrow a_1(2x'_1 + c_1) + \cdots + a_n(2x'_n + c_n) \leq b$$
$$\Leftrightarrow a_1x'_1 + \cdots + a_nx'_n \leq \left\lfloor \frac{b - a_1c_1 - \cdots - a_nc_n}{2} \right\rfloor$$

状態が遷移

入力を 1 文字消費

- $x_1, \ldots, x_n$ と $x'_1, \ldots, x'_n$ に対応する語を w, w' $(w \neq \varepsilon)$ と
  すると $w = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} w'$ が成り立つ

原始命題に対応するDFAの構成 - 2

プレスバーガー算術の原始命題 $a_1x_1 + \cdots + a_nx_n \leq b$ は以下のDFA $(Q, \Sigma, \delta, q_0, F)$ に変換される:

$$
\begin{aligned}
Q &\subset \mathbb{Z} \\
\Sigma &= 2^n \\
\delta\left(q, \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}\right) &= \left\lfloor \frac{q - \sum_{i=1}^{n} a_i c_i}{2} \right\rfloor \\
q_0 &= b \\
F &= \{q \in Q \mid 0 \leq q\}
\end{aligned}
$$

原始命題に対応するDFAの構成 - 2

プレスバーガー算術の原始命題 $a_1x_1 + \cdots + a_nx_n \leq b$ は以下のDFA $(Q, \Sigma, \delta, q_0, F)$ に変換される:

$$Q \subset \mathbb{Z}$$

$$\Sigma = 2^n$$

$$\delta\left(q, \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}\right) = \left\lfloor \frac{q - \sum_{i=1}^{n} a_i c_i}{2} \right\rfloor$$

$$q_0 = b$$

$$F = \{q \in Q \mid 0 \leq q\}$$

残りの入力が空の場合は不等式の左辺が 0 の場合に対応する

原始命題に対応するDFAの構成 - 2

プレスバーガー算術の原始命題 $a_1x_1 + \cdots + a_nx_n \leq b$ は以下のDFA $(Q, \Sigma, \delta, q_0, F)$ に変換される:

$$Q \subset \mathbb{Z}$$

$$\Sigma = 2^n$$

$$\delta\left(q, \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}\right) = \left\lfloor \frac{q - \sum_{i=1}^{n} a_i c_i}{2} \right\rfloor$$

$$q_0 = b$$

$$F = \{q \in Q \mid 0 \leq q\}$$

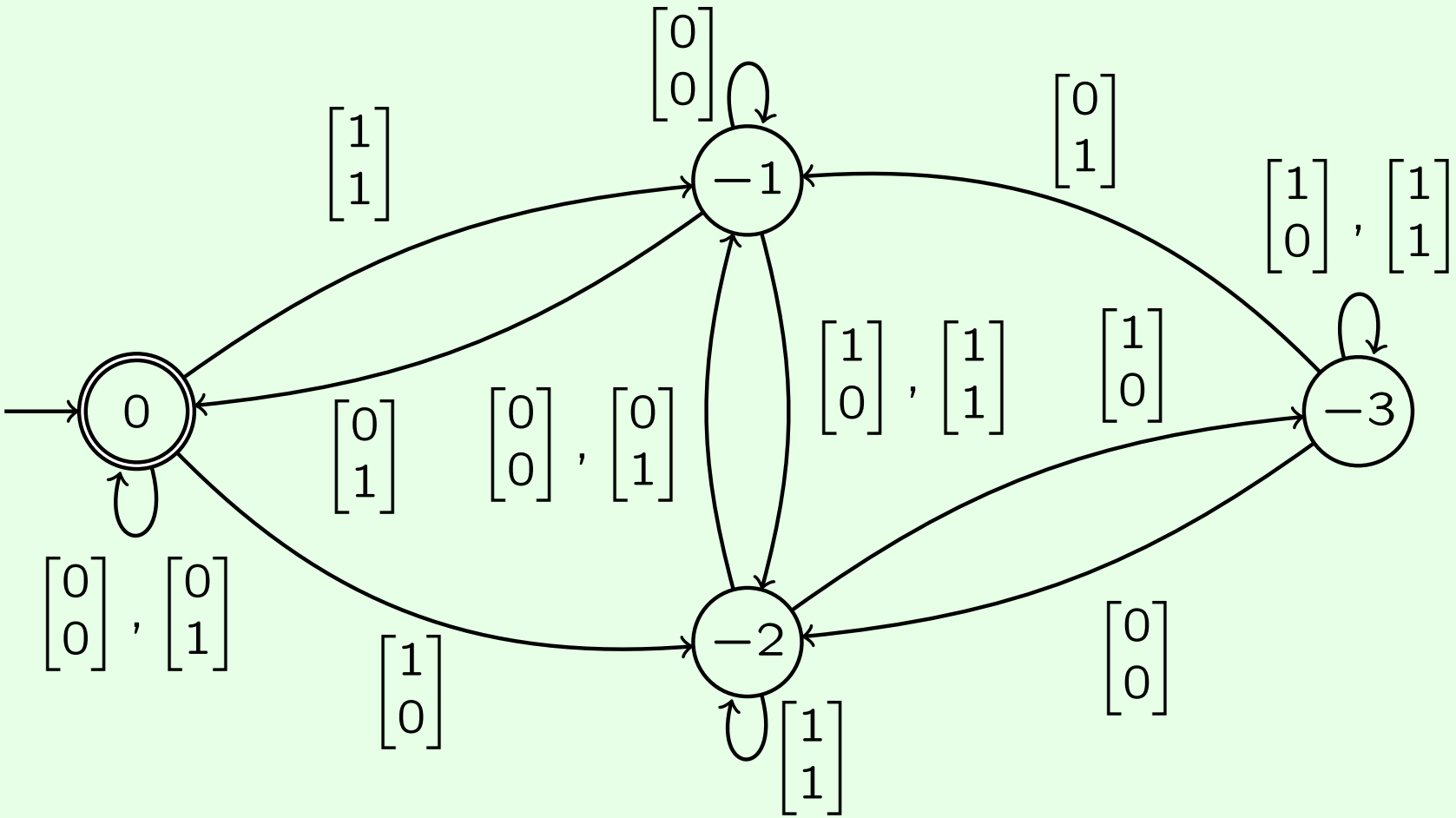
残りの入力が空の場合は不等式の左辺が0の場合に対応する

- Q は $q_0 \in Q$ かつ δ に関して閉じていれば何でも良い
  - これが必ず有限の範囲に収まることは後で説明する

原始命題に対応するDFAの構成 - 例

Example 3.

原始命題 $3x_1 - x_2 \leq 0$ を変換して得られるDFA:



論理和、論理積、否定

- それぞれ言語としては和、共通部分、補集合に対応する
- 正規言語は和、共通部分、補集合に関して閉じていることが知られている
  - 直積構成
  - DFA の受理状態、非受理状態の交換

存在量化 - 1

- 論理式 $\varphi(x_1, \ldots, x_n)$ のオートマトンが

$$\begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix} \begin{bmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,n} \end{bmatrix} \cdots \begin{bmatrix} b_{r,1} \\ b_{r,2} \\ \vdots \\ b_{r,n} \end{bmatrix}$$

を受理するとき、$\exists x_1.\, \varphi(x_1, \ldots, x_n)$ のオートマトンは

$$\begin{matrix} x_2 \\ \vdots \\ x_n \end{matrix} \begin{bmatrix} b_{1,2} \\ \vdots \\ b_{1,n} \end{bmatrix} \cdots \begin{bmatrix} b_{r,2} \\ \vdots \\ b_{r,n} \end{bmatrix}$$

を受理する

存在量化 - 1

- 論理式 $\varphi(x_1, \ldots, x_n)$ のオートマトンが

$$\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{bmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,n} \end{bmatrix} \cdots \begin{bmatrix} b_{r,1} \\ b_{r,2} \\ \vdots \\ b_{r,n} \end{bmatrix}$$

を受理するとき、$\exists x_1.\, \varphi(x_1, \ldots, x_n)$ のオートマトンは

$$\begin{array}{c} x_2 \\ \vdots \\ x_n \end{array} \begin{bmatrix} b_{1,2} \\ \vdots \\ b_{1,n} \end{bmatrix} \cdots \begin{bmatrix} b_{r,2} \\ \vdots \\ b_{r,n} \end{bmatrix}$$

を受理する

- φ に対応する DFA の状態遷移から入力の x_1 相当のビットを落とした NFA が $\exists x_1.\, \varphi$ に対応する

存在量化 - 1

- 論理式 $\varphi(x_1, \ldots, x_n)$ のオートマトンが

$$\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{bmatrix} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,n} \end{bmatrix} \cdots \begin{bmatrix} b_{r,1} \\ b_{r,2} \\ \vdots \\ b_{r,n} \end{bmatrix} \begin{bmatrix} b_{r+1,1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdots \begin{bmatrix} b_{r+s,1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

を受理するとき、$\exists x_1.\, \varphi(x_1, \ldots, x_n)$ のオートマトンは

$$\begin{array}{c} x_2 \\ \vdots \\ x_n \end{array} \begin{bmatrix} b_{1,2} \\ \vdots \\ b_{1,n} \end{bmatrix} \cdots \begin{bmatrix} b_{r,2} \\ \vdots \\ b_{r,n} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

を受理する

- φ に対応する DFA の状態遷移から入力の x_1 相当のビットを落とした NFA が $\exists x_1.\, \varphi$ に対応する
- $\mathbf{0}^*$ で受理状態に到達可能な状態は受理状態にする

存在量化 - 2

- 命題 $\varphi(x_1, \ldots, x_n)$ に対応する DFA を $(Q, 2^n, \delta, q_0, F)$ とすると、命題 $\exists x_1.\, \varphi(x_1, \ldots, x_n)$ に対応する NFA $(Q, 2^{n-1}, \delta', q_0, F')$ は以下のように定義できる:

$$\delta'\left(q, \begin{bmatrix} c_2 \\ \vdots \\ c_n \end{bmatrix}\right) = \left\{ \delta\left(q, \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}\right) \;\middle|\; c_1 \in \{0, 1\} \right\}$$

$$F' = \{q \in Q \mid \exists q' \in F.\, q\ R_0^*\ q'\}$$

$$\text{where } q\ R_0\ q' \Leftrightarrow q' \in \delta'(q, \mathbf{0})$$

存在量化 - 2

- 命題 $\varphi(x_1, \ldots, x_n)$ に対応する DFA を $(Q, 2^n, \delta, q_0, F)$ とすると、命題 $\exists x_1.\, \varphi(x_1, \ldots, x_n)$ に対応する NFA $(Q, 2^{n-1}, \delta', q_0, F')$ は以下のように定義できる:

$$\delta'\left(q, \begin{bmatrix} c_2 \\ \vdots \\ c_n \end{bmatrix}\right) = \left\{ \delta\left(q, \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}\right) \;\middle|\; c_1 \in \{0, 1\} \right\}$$

$$F' = \{q \in Q \mid \exists q' \in F.\, q\ R_0^*\ q'\}$$

$$\text{where } q\ R_0\ q' \Leftrightarrow q' \in \delta'(q, \mathbf{0})$$

- 必要に応じてサブセット構成で DFA にする

全称量化

- $\forall x_1.\, \varphi(x_1, \ldots, x_n)$ は $\neg\exists x_1.\, \neg\varphi(x_1, \ldots, x_n)$ と同値
- 全称量化に対応するオートマトンの構成はここまでに説明した方法の組み合わせでできる

形式化

以下の証明器とライブラリを用いる

証明器 Coq 8.5 beta 2

ライブラリ
- SSReflect 1.5
- MathComp 1.5
- A Constructive Theory of Regular Languages in Coq [Doczkal et al., 2013]

Coqの正規言語ライブラリ [Doczkal et al., 2013]

- SSReflect, MathComp ベース
  - SSReflect には元々有限集合のライブラリがある
  - 有限オートマトンの定義を書くのに便利
- 正規表現と有限オートマトンに関する基本的な道具が一通り揃っている
  - 直積構成
  - サブセット構成
  - 反復補題
  - Kleene の定理
  - Myhill-Nerode の定理

Coqの正規言語ライブラリ [Doczkal et al., 2013]

- SSReflect, MathComp ベース
  - SSReflect には元々有限集合のライブラリがある
  - 有限オートマトンの定義を書くのに便利
- 正規表現と有限オートマトンに関する基本的な道具が一通り揃っている
  - 直積構成
  - サブセット構成
  - 反復補題
  - Kleene の定理
  - Myhill-Nerode の定理

fintype: SSReflectの有限集合ライブラリ

- 取り得る値の数が有限個であるような型を一般化したfinTypeを提供
- 定義: 型Tの要素を列挙したリスト$e \in T^*$について$\forall x \in T. |e|_x = 1$ならば$T$はfinType
- finTypeの例:
  - 'I_n $= \{0, \ldots, n-1\}$
  - 'I_n * bool
  - option 'I_n

Coq 上での n 組の表現

自由変数への自然数の割当てを表す $\mathbb{N}^n$ やオートマトンの入力 2^n のような n 組は `finfun` で表現:

- `finfun` は定義域が `finType` の関数を表す型

Coq 上での n 組の表現

自由変数への自然数の割当てを表す $\mathbb{N}^n$ やオートマトンの入力 2^n のような n 組は `finfun` で表現:

- `finfun` は定義域が `finType` の関数を表す型
- 実体はリストなので Leibniz equality がそのまま使える

Coq 上での n 組の表現

自由変数への自然数の割当てを表す $\mathbb{N}^n$ やオートマトンの入力 2^n のような n 組は `finfun` で表現:

- `finfun` は定義域が `finType` の関数を表す型
- 実体はリストなので Leibniz equality がそのまま使える
- `A, B : finType` について `{ffun A -> B}` は `finType`

Coq 上での n 組の表現

自由変数への自然数の割当てを表す $\mathbb{N}^n$ やオートマトンの入力 2^n のような n 組は `finfun` で表現:

- `finfun` は定義域が `finType` の関数を表す型
- 実体はリストなので Leibniz equality がそのまま使える
- `A, B : finType` について `{ffun A -> B}` は `finType`
  - 2^n の語を入力として取るオートマトンも簡単に書ける

Coq 上での n 組の表現

自由変数への自然数の割当てを表す $\mathbb{N}^n$ やオートマトンの入力 2^n のような n 組は `finfun` で表現:

- `finfun` は定義域が `finType` の関数を表す型
- 実体はリストなので Leibniz equality がそのまま使える
- `A, B : finType` について `{ffun A -> B}` は `finType`
  - 2^n の語を入力として取るオートマトンも簡単に書ける
- 例: $(0, 1, 0, 1, \ldots) \in 2^n$ は
  `[ffun i : 'I_n => odd i] : bool ^ n` と書ける

自然数の組と 2^n の語の相互変換

- w←a : $\mathbb{N}^n \to 2^{n*}$
- a←w : $2^{n*} \to \mathbb{N}^n$

自然数の組と 2^n の語の相互変換

- $\mathsf{w}{\leftarrow}\mathsf{a} : \mathbb{N}^n \to 2^{n*}$
- $\mathsf{a}{\leftarrow}\mathsf{w} : 2^{n*} \to \mathbb{N}^n$

以下が成り立つ:

- $\forall a \in \mathbb{N}^n.\ \mathsf{a}{\leftarrow}\mathsf{w}(\mathsf{w}{\leftarrow}\mathsf{a}(a)) = a$
- $\forall w_1\, w_2 \in 2^{n*}.$
  $\mathsf{a}{\leftarrow}\mathsf{w}(w_1 w_2) = \mathsf{a}{\leftarrow}\mathsf{w}(w_1) + 2^{|w_1|}\, \mathsf{a}{\leftarrow}\mathsf{w}(w_2)$
- $\forall n \in \mathbb{N}.\ \mathsf{a}{\leftarrow}\mathsf{w}(\mathbf{0}^n) = \mathbf{0}$
- $\forall w \in 2^{n*}, n \in \mathbb{N}.\ \mathsf{a}{\leftarrow}\mathsf{w}(w\mathbf{0}^n) = \mathsf{a}{\leftarrow}\mathsf{w}(w)$

原始命題に対応するDFAの構成 - 1

原始命題 $a_0x_0 + \cdots + a_{n-1}x_{n-1} \leq b$ を変換して得られるDFAの状態の上限、下限はそれぞれ:

$$u = \max\left(-\sum_{i=0}^{n-1}\{a_i \mid a_i \leq 0\},\ b\right)$$

$$l = \min\left(-\sum_{i=0}^{n-1}\{a_i \mid 0 \leq a_i\},\ b\right)$$

原始命題に対応するDFAの構成 - 1

原始命題 $a_0x_0 + \cdots + a_{n-1}x_{n-1} \leq b$ を変換して得られるDFAの状態の上限、下限はそれぞれ:

$$u = \max\left(-\sum_{i=0}^{n-1}\{a_i \mid a_i \leq 0\},\ b\right)$$

$$l = \min\left(-\sum_{i=0}^{n-1}\{a_i \mid 0 \leq a_i\},\ b\right)$$

l 以上 u 以下の整数の型 `range l u` とその `finType` のインスタンスを作る

原始命題に対応するDFAの構成 - 2

目的のDFAを定義するために以下を証明する:

- 初期状態 q_0 が状態の集合に含まれている

$$l \leq b \leq u$$

- 状態の集合が δ について閉じている

$$\forall q : \mathtt{range}\, l\, u, \mathbf{c} : 2^n.\, l \leq \left\lfloor \frac{q - \sum_{i=0}^{n-1} a_i c_i}{2} \right\rfloor \leq u$$

Lemma 4.

$$w \in L(A) \Leftrightarrow \sum_{i=0}^{n-1} a_i\, \mathsf{a} \leftarrow \mathsf{w}(w)_i \leq b$$

Lemma 4の証明

$$w \in L(A) \Leftrightarrow \sum_{i=0}^{n-1} a_i \, \mathrm{a} \leftarrow \mathrm{w}(w)_i \leq b$$

Lemma 4の証明

$$0 \leq \hat{\delta}(b, w) \Leftrightarrow \sum_{i=0}^{n-1} a_i \, \mathrm{a} \leftarrow \mathrm{w}(w)_i \leq b$$

Lemma 4の証明

$$0 \leq \hat{\delta}(b, w) \Leftrightarrow \sum_{i=0}^{n-1} a_i \, \mathsf{a}{\leftarrow}\mathsf{w}(w)_i \leq b$$

$w = \varepsilon$ の場合:

$$\begin{aligned} 0 \leq \hat{\delta}(b, \varepsilon) &\Leftrightarrow 0 \leq b \\ &\Leftrightarrow \sum_{i=0}^{n-1} a_i \mathbf{0}_i \leq b \\ &\Leftrightarrow \sum_{i=0}^{n-1} a_i \mathsf{a}{\leftarrow}\mathsf{w}(\varepsilon)_i \leq b \end{aligned}$$

Lemma 4の証明

$$0 \leq \hat{\delta}(b, w) \Leftrightarrow \sum_{i=0}^{n-1} a_i \, \mathsf{a}{\leftarrow}\mathsf{w}(w)_i \leq b$$

$w = cw'$ の場合:

$$0 \leq \hat{\delta}(b, cw') \Leftrightarrow 0 \leq \hat{\delta}\left(\left\lfloor \frac{b - \sum_{i=0}^{n-1} a_i c_i}{2} \right\rfloor, w'\right)$$

$$\Leftrightarrow 2 \sum_{i=0}^{n-1} a_i \, \mathsf{a}{\leftarrow}\mathsf{w}(w')_i \leq b - \sum_{i=0}^{n-1} a_i c_i \quad \text{(IH)}$$

$$\Leftrightarrow \sum_{i=0}^{n-1} a_i (2 \, \mathsf{a}{\leftarrow}\mathsf{w}(w')_i + c_i) \leq b$$

$$\Leftrightarrow \sum_{i=0}^{n-1} a_i \mathsf{a}{\leftarrow}\mathsf{w}(cw')_i \leq b$$

Lemma 4 の証明

$$0 \leq \hat{\delta}(b, w) \Leftrightarrow \sum_{i=0}^{n-1} a_i \, \mathsf{a} \leftarrow \mathsf{w}(w)_i \leq b$$

$w = cw'$ の場合:

$$0 \leq \hat{\delta}(b, cw') \Leftrightarrow 0 \leq \hat{\delta}\left(\left\lfloor \frac{b - \sum_{i=0}^{n-1} a_i c_i}{2} \right\rfloor, w'\right)$$

$\sum$ に関する式変形
`big_morph`,
`big_split`

$$\Leftrightarrow 2 \sum_{i=0}^{n-1} a_i \, \mathsf{a} \leftarrow \mathsf{w}(w')_i \leq b - \sum_{i=0}^{n-1} a_i c_i \quad \text{(IH)}$$

$$\Leftrightarrow \sum_{i=0}^{n-1} a_i (2 \, \mathsf{a} \leftarrow \mathsf{w}(w')_i + c_i) \leq b$$

$$\Leftrightarrow \sum_{i=0}^{n-1} a_i \mathsf{a} \leftarrow \mathsf{w}(cw')_i \leq b$$

存在量化に対応するNFAの構成 - 1

F' の定義はMathCompの有限グラフライブラリを使うと簡単に書ける

$$q \in F' \Leftrightarrow \exists q' \in F.\, q\ R_0^*\ q'$$
$$\text{where } q\ R_0\ q' \Leftrightarrow q' \in \delta'(q, \mathbf{0})$$

```
nfa_fin q :=
  [exists (q' | q' \in dfa_fin A),
   connect (nfa_trans' ^~ [ffun => false]) q q']
```

存在量化に対応するNFAの構成 - 1

F' の定義はMathCompの有限グラフライブラリを使うと簡単に書ける

$$q \in F' \Leftrightarrow \exists q' \in F.\, q\ R_0^*\ q'$$
$$\text{where } q\ R_0\ q' \Leftrightarrow q' \in \delta'(q, \mathbf{0})$$

```
nfa_fin q :=
  [exists (q' | q' \in dfa_fin A),
   connect (nfa_trans' ^~ [ffun => false]) q q']
```

- `[exists ...]` は有限集合用の存在量化
- `connect` は有限集合上の関係の反射推移閉包を計算する関数
- どちらも返すのは `Prop` ではなく `bool`

存在量化に対応するNFAの構成 - 2

存在量化の外と中で自由変数の数が異なるので以下の関数を用意する:

- `cons_tuple` : $A \times A^n \to A^{n+1}$
- `cons_word` : $\mathbb{N} \times 2^{n*} \to 2^{n+1*}$

存在量化に対応するNFAの構成 - 2

存在量化の外と中で自由変数の数が異なるので以下の関数を用意する:

- $\texttt{cons\_tuple} : A \times A^n \to A^{n+1}$
- $\texttt{cons\_word} : \mathbb{N} \times 2^{n*} \to 2^{n+1*}$

以下が成り立つ:

- $\forall a \in \mathbb{N}, w \in 2^{n*}.$
  $\texttt{cons\_tuple}(a, \texttt{a←w}(w)) = \texttt{a←w}(\texttt{cons\_word}(a, w))$

存在量化に対応するNFAの構成 - 3

Lemma 5.

命題 $\varphi(x_1, \ldots, x_{n+1})$ と 2^{n+1} の語を受理する *DFA* A について

$$\forall w \in 2^{n+1*}.\, w \in L(A) \Leftrightarrow \varphi(\mathrm{a}{\leftarrow}\mathrm{w}(w))$$

と仮定する

このとき A に存在量化に対応する *NFA* の構成法を適用して得られる A' は

$$\forall w \in 2^{n*}.\, w \in L(A') \Leftrightarrow (\exists a \in \mathbb{N}.\, \varphi(\, a, \mathrm{a}{\leftarrow}\mathrm{w}(w)))$$

を満たす

存在量化に対応するNFAの構成 - 3

Lemma 5.

命題 $\varphi(x_1, \ldots, x_{n+1})$ と 2^{n+1} の語を受理する *DFA* A について

$$\forall w \in 2^{n+1*}.\, w \in L(A) \Leftrightarrow \varphi(\mathrm{a}{\leftarrow}\mathrm{w}(w))$$

と仮定する

このとき A に存在量化に対応する *NFA* の構成法を適用して得られる A' は

$$\forall w \in 2^{n*}.\, w \in L(A') \Leftrightarrow (\exists a \in \mathbb{N}.\, \varphi(\underbrace{a, \mathrm{a}{\leftarrow}\mathrm{w}(w)}_{\texttt{cons\_tuple}}))$$

を満たす

存在量化に対応するNFAの構成 - 3

Lemma 5.

命題 $\varphi(x_1, \ldots, x_{n+1})$ と 2^{n+1} の語を受理する *DFA* A について

$$\forall w \in 2^{n+1*}.\, w \in L(A) \Leftrightarrow \varphi(\mathrm{a}{\leftarrow}\mathrm{w}(w))$$

と仮定する

このとき A に存在量化に対応する *NFA* の構成法を適用して得られる A' は

$$\forall w \in 2^{n*}.\, w \in L(A') \Leftrightarrow (\exists a \in \mathbb{N}.\, \varphi(\underbrace{a, \mathrm{a}{\leftarrow}\mathrm{w}(w)}_{\texttt{cons\_tuple}}))$$

を満たす

証明は煩雑なので省略

まとめ

- プレスバーガー算術の命題を真偽が受理非受理に対応するように有限オートマトンに変換
- Coq での形式化
  - 600 行
  - SSReflect，MathComp，正規言語のライブラリに必要な道具がほとんど揃っている
- https://github.com/pi8027/sandpit/blob/master/coq/presburger.v

参考文献 I

Boudet, A. and Comon, H. (1996).

Diophantine equations, presburger arithmetic and finite automata.

In *Trees in Algebra and Programming - CAAP '96*, volume 1059 of *Lecture Notes in Computer Science*, pages 30–43. Springer Berlin Heidelberg.

Doczkal, C., Kaiser, J.-O., and Smolka, G. (2013).

A constructive theory of regular languages in Coq.

In *Certified Programs and Proofs*, volume 8307 of *Lecture Notes in Computer Science*, pages 82–97. Springer International Publishing.