置換の自動証明タクティク

坂口 和彦 (筑波大学)

置換は列に関する重要な概念の 1 つであり、プログラムの検証においてもソートアルゴリズムの正当性の証明等で利用される。本発表では、対話的定理証明器 Coq [3] 上で、検証済みのマージソートとリフレクションを用いてリスト上の置換同値関係に関する自動証明を行う手法 [2] を提案する。また、本手法の応用として部分的に末尾再帰化したマージソートの検証例を示す。

置換と置換同値関係

リスト $[x_1; \ldots; x_n]$ の**置換**とは、その要素を並び換えて得られるリストである。例えば、リスト $[1; 2; 3]$ の置換は以下の 6 通りである。

$$[1;2;3], [1;3;2], [2;1;3], [2;3;1], [3;1;2], [3;2;1]$$

リスト X, Y が互いの置換であるとき、X, Y は**置換同値** ($X =_{\mathsf{perm}} Y$) である。$=_{\mathsf{perm}}$ は同値関係であり、以下の補題が成り立つ。

$$\forall X, Y.\ X \mathbin{+\!\!+} Y =_{\mathsf{perm}} Y \mathbin{+\!\!+} X \tag{1}$$

$$\forall X, X', Y.\ X =_{\mathsf{perm}} X' \Leftrightarrow X \mathbin{+\!\!+} Y =_{\mathsf{perm}} X' \mathbin{+\!\!+} Y \tag{2}$$

$$\forall X.\ X =_{\mathsf{perm}} X^R \tag{3}$$

$$\text{where}\quad [x_1; \ldots; x_n] \mathbin{+\!\!+} [y_1; \ldots; y_m] = [x_1; \ldots; x_n; y_1; \ldots; y_m]$$
$$[x_1; \ldots; x_n]^R = [x_n; \ldots; x_1]$$

置換同値関係に対する自動証明

上に示した補題群を用いることで、置換同値関係の両辺の $\_ \mathbin{+\!\!+} \_, \_^R$ の入れ子*の中に共通して現れる項を取り除く同値変形ができる。本研究では、この操作を Coq のゴール中に現れる置換同値関係に対して行うタクティク `autoperm` を開発した。以下に、その同値変形の例を示す。

$$(A \mathbin{+\!\!+} B) \mathbin{+\!\!+} (C \mathbin{+\!\!+} D) =_{\mathsf{perm}} (C^R \mathbin{+\!\!+} (B' \mathbin{+\!\!+} A^R)) \mathbin{+\!\!+} D$$

$\Updownarrow$ 両辺の $\_ \mathbin{+\!\!+} \_, \_^R$ の入れ子を flatten$[\ldots]$ の形に変形する

$$\mathsf{flatten}[A; B; C; D] =_{\mathsf{perm}} \mathsf{flatten}[C; B'; A; D]$$

$\Updownarrow$ 両辺の flatten$[\ldots]$ の各要素に番号を付ける

$$\mathsf{flatten}(\mathsf{map}_f [4; 3; 2; 1]) =_{\mathsf{perm}} \mathsf{flatten}(\mathsf{map}_f [2; 0; 4; 1])$$

where $f(0) = B'$, $f(1) = D$, $f(2) = C$, $f(3) = B$, $f(4) = A$, $f(n) = []$ $(5 \leqslant n)$

$\Updownarrow$ 両辺の番号の列を降順にソートする

$$\mathsf{flatten}(\mathsf{map}_f (\mathit{sort}_{\geqslant} [4; 3; 2; 1])) =_{\mathsf{perm}} \mathsf{flatten}(\mathsf{map}_f (\mathit{sort}_{\geqslant} [2; 0; 4; 1]))$$

$\Updownarrow$ `simpl`

$$\mathsf{flatten}(\mathsf{map}_f [4; 3; 2; 1]) =_{\mathsf{perm}} \mathsf{flatten}(\mathsf{map}_f [4; 2; 1; 0])$$

$\Updownarrow$ 両辺の番号の列から重複を取り除く

$$\mathsf{flatten}(\mathsf{map}_f [3]) =_{\mathsf{perm}} \mathsf{flatten}(\mathsf{map}_f [0])$$

$\Updownarrow$ `simpl`

$$B =_{\mathsf{perm}} B'$$

* 実際には $[], \_ :: \_, \mathsf{rcons}(\_, \_), \mathsf{catrev}(\_, \_)$ も扱う。

マージソートの末尾再帰化

(* マージはアキュムレータ引数を追加して末尾再帰化する *)
Fixpoint merge $((\leqslant) : \alpha \to \alpha \to \mathsf{bool})$ $(X\ Y\ A : \alpha\ \mathsf{list}) : \alpha\ \mathsf{list} :=$
  match X, Y with
    | $[], \_ \Rightarrow \mathsf{catrev}(Y, A)$ | $\_, [] \Rightarrow \mathsf{catrev}(X, A)$
    (* catrev は $\mathsf{catrev}(X, Y) = X^R \mathbin{+\!\!+} Y$ を満たす末尾再帰関数 *)
    | $x :: X', y :: Y' \Rightarrow$
      if $x \leqslant y$ then $\mathit{merge}\ (\leqslant)\ X'\ Y\ (x :: A)$ else $\mathit{merge}\ (\leqslant)\ X\ Y'\ (y :: A)$
  end.

(* 以下の $\mathit{push}, \mathit{pop}$ を用いてマージソートを構造的再帰の形で記述する [1] *)
Fixpoint push $((\leqslant) : \alpha \to \alpha \to \mathsf{bool})$
    $(r : \mathsf{bool})\ (X : \alpha\ \mathsf{list})\ (\mathcal{Y} : \alpha\ \mathsf{list}\ \mathsf{list}) : \alpha\ \mathsf{list}\ \mathsf{list} :=$
  match $\mathcal{Y}$ with
    | $[] :: \mathcal{Y}'$ | $[]$ as $\mathcal{Y}' \Rightarrow X :: \mathcal{Y}'$
    | $Y :: \mathcal{Y}' \Rightarrow$
      let $Z :=$ if r then $\mathit{merge}\ (\geqslant)\ X\ Y\ []$ else $\mathit{merge}\ (\leqslant)\ Y\ X\ []$ in
        (* $(\geqslant)$ は $(\lambda x\ y.\ y \leqslant x)$ の略記 *)
      $[] :: \mathit{push}\ (\leqslant)\ (\neg r)\ Z\ \mathcal{Y}'$
      (* この部分だけ tail call ではないがスタックの消費量は $\mathcal{O}(\log n)$ *)
  end.

Fixpoint pop $((\leqslant) : \alpha \to \alpha \to \mathsf{bool})$
    $(r : \mathsf{bool})\ (X : \alpha\ \mathsf{list})\ (\mathcal{Y} : \alpha\ \mathsf{list}\ \mathsf{list}) : \alpha\ \mathsf{list} :=$
  if $\mathcal{Y}$ is $Y :: \mathcal{Y}'$
  then let $Z :=$ if r then $\mathit{merge}\ (\geqslant)\ X\ Y\ []$ else $\mathit{merge}\ (\leqslant)\ Y\ X\ []$ in
    $\mathit{pop}\ (\leqslant)\ (\neg r)\ Z\ \mathcal{Y}'$
  else if r then X^R else X.

Fixpoint sort' $((\leqslant) : \alpha \to \alpha \to \mathsf{bool})$ $(X : \alpha\ \mathsf{list})\ (\mathcal{Y} : \alpha\ \mathsf{list}\ \mathsf{list}) : \alpha\ \mathsf{list} :=$
  if X is $x :: x' :: X'$
  then $\mathit{sort}'\ (\leqslant)\ X'\ (\mathit{push}\ (\leqslant)\ \mathsf{false}$ (if $x \leqslant x'$ then $[x; x']$ else $[x'; x]$) $\mathcal{Y})$
  else $\mathit{pop}\ (\leqslant)\ \mathsf{false}\ X\ \mathcal{Y}$.

Definition sort $((\leqslant) : \alpha \to \alpha \to \mathsf{bool})$ $(X : \alpha\ \mathsf{list}) : \alpha\ \mathsf{list} := \mathit{sort}'\ (\leqslant)\ X\ []$.

マージソートの正当性の証明

定理 1 (sort_sorted)**.** $(\leqslant_1), (\leqslant_2) : \alpha \to \alpha \to \mathsf{bool}$ がそれぞれ完全、推移的であり、リスト $X : \alpha\ \mathsf{list}$ が $\leqslant_2$ でソート済みであるとき、リスト $\mathit{sort}\ (\leqslant_1)\ X$ は以下の二項関係 $\leqslant$ でソートされた列となる。

$$x \leqslant y \overset{\text{def}}{\iff} x \leqslant_1 y \land (\neg y \leqslant_1 x \lor x \leqslant_2 y)$$

即ち、sort は安定ソートである。

定理 2 (perm_sort)**.** 任意の $(\leqslant) : \alpha \to \alpha \to \mathsf{bool}$ と $X : \alpha\ \mathsf{list}$ について、以下が成り立つ。

$$\mathit{sort}\ (\leqslant)\ X =_{\mathsf{perm}} X$$

置換同値関係に対する自動証明を用いて、定理 2 とその補題を 33 行で証明した。

今後の予定

1. 置換同値関係以外（multiset 等）への一般化
2. 末尾再帰化したマージソートを用いて自動証明自体を高速化

参考文献

[1] Georges Gonthier. *RE: [Coq-Club] Sorting*. URL: https://sympa.inria.fr/sympa/arc/coq-club/2009-04/msg00040.html.

[2] Kazuhiko Sakaguchi. *autoperm.v*. URL: https://gist.github.com/pi8027/35997c5dfb4cacd277ca53592bec917e.

[3] The Coq Development Team. *The Coq Proof Assistant Reference Manual*. Version 8.7.0. 2017. URL: https://coq.inria.fr/distrib/V8.7.0/refman/.