paramodulation と superposition

服部桃子

June 17, 2019

Contents

前回までのあらすじ

- 第 1 回: 導出原理と Egison での実装
- 第 2 回: 項書き換え系の話
  - 停止性、合流性
  - Lexicographic Path Order
  - Knuth-Bendix 完備化

前回までのあらすじ / 導出原理

定義: resolvent

- resolvent の定義: 節 c, c_1, c_2 に対し以下を満たすリテラル L_1, L_2 が存在するとき、c を c_1, c_2 の resolvent という
  - $L_1 \in c_1$ かつ $L_2^* \in c_2$
  - L_1, L_2 の最汎単一化子 θ が存在し、$c = (c_1\theta \setminus \{L_1\theta\}) \cup (c_2\theta \setminus \{L_2^*\theta\})$

$$\frac{\{P(a,b)\} \qquad \{\neg P(a,x), Q(x)\}}{\{Q(b)\}}\ \theta = [b/x]$$

```
(define $resolution ; resolvent を求める
  (match-all-lambda (multiset (multiset literal))
    {[<cons <cons <lit ,1 $t> $xs>
       ; t との最汎単一化子が σ であるようなリテラル ¬s
       <cons <cons <lit ,-1 (& <unify ,t $sigma> $s)> $ys>
         _>>
      {@(subst-clause sigma xs) @(subst-clause sigma ys)}]}))
```

前回までのあらすじ / 群の性質を示す

- 群の公理から群の性質を示す
  - 等式は単なる 2 項述語なので 等式の公理 も必要

```
(define $p3
  {@axioms-of-groups
   @axioms-of-equality
   @congruence-groups
   ; (negation of) the goal: e * x = x
   {<Lit -1 <Compound "=" <Compound "*" {e e1}> e1>>}})

> (solve p3) ; => 現実的な時間では終わらない
```

目標

Egison で実装した導出原理のプログラムを改良し、探索空間を減らす

- paramodulation
- superposition

Contents

$$\frac{C \lor s \doteq t \qquad D \lor P[s']}{\sigma(C \lor D \lor P[t])}$$

- 導出原理で等式を直接扱う手法
- 上のような規則を追加
  - $s \doteq t$ は「$s = t$ または $t = s$」
  - σ は s と s' の最汎単一化子 ($\sigma = \mathrm{mgu}(s, s')$)
- 等式の公理は reflexivity(反射性) 以外は不要になる
  - functional reflexivity も必要とされる
    - $f(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$

定理: Refutational completeness

S が normal model[a] を持たないならば、S に reflexivity と functional reflexivity を加えたものから paramodulation と導出原理によって空節を導くことができる

[a] "="が等号として解釈されるモデル

証明 (概略):

普通の導出原理の完全性を利用して、「導出原理 + 等式の公理で導出可能」ならば「導出原理 + paramodulation + reflexivity で導出可能」を示す

- 例えば導出原理+(1-ary)functional congruence を用いている場合...

$$\frac{\neg(x = x') \vee f(\ldots, x, \ldots) = f(\ldots, x', \ldots) \qquad C \vee s = t}{C \vee f(\ldots, s, \ldots) = f(\ldots, t, \ldots)}$$

paramodulation + functional reflexivity で次のように示せる

$$\frac{f(\ldots, x, \ldots) = f(\ldots, [x], \ldots) \vee C \qquad s = t}{f(\ldots, s, \ldots) = f(\ldots, t, \ldots) \vee C} \ \sigma = [s/x]$$

- 他、(1-ary) predicate congruence を用いている場合や、対称性・推移性を用いている場合も同様の議論

paramodulation / 実装

prop

```
; predicate とは別に equation を用意する
(define $prop
  (matcher
    {[<pred $ $> [string (list term)]
       {[<Pred $p $a> {[p a]}]
        [_ {}]}]
     [ <equation $ $>  [term term]
       ; unordered-pair のようにマッチ (l ≐ r を表現)
      {[<Equation $l $r> { [l r] [r l] }]
       [_ {}]}]
     [<unify ,$t $> something
      {[$s (punify t s)]}]
     [<subterm $ $> [term something]
      {[$s (psubterm s)]}]
     [$ something
       {[$tgt {tgt}]}]}))
```

paramodulation / 実装

subterm

```
(define $prop
  (matcher
    {(...)
     ; subterm とその「環境」(term -> prop の関数) のペアにマッチ
     [<subterm $ $> [term something]
      [$s (psubterm s)]]
     [$ something
       {[$tgt {tgt}]}]}))
```

```
;; term -> {term, term -> term}
(define $subterm
  (match-all-lambda term
    {[<compound $f <join $xs <cons <subterm $x $context> $xs'>>>
       (let {[$context'
                 (lambda [$x] <Compound f (append xs (cons x xs'))>)]}
         [x (compose context' context)])]
     [$t [t id]]}))
```

paramodulation / 実装

resolution

```
(define $resolution
  (match-all-lambda (multiset (multiset literal))
     ; paramodulation
    {[<cons <cons <lit ,1 <equation $s $t>> $xs>
       <cons <cons <lit $sign
           ; context[s'], s'は変数でない, sigma = mgu(s, s')
           <subterm (& !<var _> <unify ,s $sigma> $s') $context>> $ys>
         _>>
      (csubst sigma {@xs @ys <Lit sign (context t)>})
     ]
     ; 普通の resolution
     [<cons <cons <lit ,1 $t> $xs>
       <cons <cons <lit ,-1 (& <unify ,t $sigma> $s)> $ys>
         _>>
      (csubst sigma {@xs @ys})
     ]}))
```

Contents

superposition

- 項、等式、リテラル、節すべてに順序 (重み) をつけ、一番重いものを書き換える
  - 書き換えの結果軽くなるように
- 参考文献

1. Bachmair, Leo, and Harald Ganzinger. *Rewrite-based equational theorem proving with selection and simplification.* Journal of Logic and Computation 4.3 (1994): 217-247.
2. `http://resources.mpi-inf.mpg.de/departments/rg1/teaching/autrea-ss12/script/script23.pdf`

order の multiset 拡張

$>$ を集合 S 上の ordering とすると、S の要素の multiset 上への拡張 $>_{mul}$ は次のように定義される

$$M >_{mul} N \overset{\text{def}}{\iff} \begin{cases} (i)\, M \neq N \\ (ii)\, \underbrace{N(x)}_{\#x \in N} > M(x) \Rightarrow \exists y > x.\, M(y) > N(y) \end{cases}$$

例

$$\{1, 2, 2, 3\} >_{mul} \{1, 1, 2, 3\}$$
$$\{1, 2, 3\} <_{mul} \{1, 1, 2, 3\}$$

superposition / orderings

等式の順序

$$s = t \succ^{e} u = v \overset{\text{def}}{\iff} \{s, t\} \succ_{mul} \{u, v\}$$

リテラルの順序

$$L(s = t) \succ^{o} L(u = v) \overset{\text{def}}{\iff} M(L(s = t))\,(\succ_{mul})_{mul}\, M(L(u = v))$$

$$L(s, t) = (s = t) \text{ または } (s \neq t)$$

$$M(s = t) = \{\{s\}, \{t\}\} \quad M(s \neq t) = \{\{s, \bot\}, \{t, \bot\}\}$$

例

$$S(S(O)) = S(O) \succ^{e} S(S(O)) = O$$

$$S(S(O)) = S(O) \prec^{o} S(S(O)) \neq O$$

reductive

$>$ を reduction ordering とする。
節 $C \vee s = t$ が $s = t$ について reductive であるとは、

1. $t \not\succeq s$
2. リテラル $s = t$ はリテラルの順序について $C \vee s = t$ で strict に極大 ($\neg \exists x \in C. \, x \succeq^o (s = t)$)

Superposition Left:

$$\frac{C \lor s = t \qquad u[s'] \neq v \lor D}{(u[t] \neq v \lor C \lor D)\sigma} \; \sigma = \text{mgu}(s, s')$$

Superposition Right:

$$\frac{C \lor s = t \qquad u[s'] = v \lor D}{(u[t] = v \lor C \lor D)\sigma} \; \sigma = \text{mgu}(s, s')$$

Equality Resolution:

$$\frac{u \neq v \lor C}{C\sigma} \; \sigma = \text{mgu}(u, v)$$

Equality Factoring:

$$\frac{C \lor s = t \lor s' = t'}{(C \lor t \neq t' \lor s' = t')\sigma} \; \sigma = \text{mgu}(s, s')$$

superposition / 推論規則

Superposition Left

$$\frac{C \lor s = t \qquad u[s'] \neq v \lor D}{(u[t] \neq v \lor C \lor D)\sigma}\ \sigma = \mathrm{mgu}(s, s')$$

- 節 $(C \lor s = t)\sigma$ は $s\sigma = t\sigma$ について reductive
- $v\sigma \not\succeq u\sigma$, $u\sigma \neq v\sigma$ は節 $(u \neq v \lor D)\sigma$ に含まれるリテラルの中で極大
- s' は変数でない

Superposition Right

$$\frac{C \lor s = t \qquad u[s'] = v \lor D}{(u[t] = v \lor C \lor D)\sigma}\ \sigma = \mathrm{mgu}(s, s')$$

- 節 $(C \lor s = t)\sigma$ は $s\sigma = t\sigma$ について reductive
- 節 $(D \lor u = v)\sigma$ は $u\sigma = v\sigma$ について reductive
- s' は変数でない

superposition / 推論規則

Equality Resolution

$$\frac{u \neq v \lor C}{C\sigma} \quad \sigma = \mathrm{mgu}(u, v)$$

- $u\sigma \neq v\sigma$ は節 $(u \neq v \lor C)\sigma$ に含まれるリテラルの中で極大

Equality Factoring

$$\frac{C \lor s = t \lor s' = t'}{(C \lor t \neq t' \lor s' = t')\sigma} \quad \sigma = \mathrm{mgu}(s, s')$$

- $t\sigma \not\succ s\sigma$ かつ $t'\sigma \not\succ s'\sigma$
- $s\sigma = t\sigma$ は節 $(C \lor s = t \lor s' = t')\sigma$ に含まれるリテラルの中で極大

superposition / 実装

- 実装の詳細は面白いところがないので割愛します
- 動いているところ: demo