新しい項システムについて

H2nI3sc

目 次

♠は疑問のあるセクション

1 作業中

1.1 代入操作

$\sigma_1 \cdot \sigma_2$ の定義はよくできている。
ただし、$\{\overleftarrow{x}\}$ がおかしくしていないか。
その性質をちゃんと理解しているかどうか。

1.1.1 σ と t の順番とは

$\sigma[t] = t \cdot \sigma$ は成り立つと思うのだが、それは項になる。

1.2 merge

2 何を書きたいか

2.1 意図

Term と substition(代入) の (たぶん) 新しいシステムを考えた。変数の binding の考え方を変える。
誰が得をするのかがよくわからないが、このほうがシンプルになるので、どうなるのかを知りたい。

3 Term, 項の定義

3.1 Symbol

Symbol は、基本の単語であり、かっこなどを含まない。
定義

$$< symbol >: symbol$$

例

$$x, y, a, b$$

3.2 Term

定義

$$< term > ::= \quad < symbol > | < symbol > (< term >, ...)$$

一階述語では、$\forall$ や $\exists$ によって bind されることで変数を指定するが、ここでの term ではその区別はない。

例

$$x, y, f(x), g(a, x)$$

3.3 項の関係

3.3.1 部分表現

項 s が t の、(t_1, s_1) が (t, s) の部分表現であることを次のように表記する。

$$t \sqsupset s$$
$$(t_1, s_1) \sqsupset (t, s) \quad : \quad t_1 \sqsupset t \wedge s_1 \sqsupset s \, , respectively$$

4 代入, Substitution

symbol x と項 t が与えられるとき、代入は $\{x \leftarrow t\}$ と書く。代入は σ で表すことが多い。

この x の場所に書かれる記号を変数と呼ぶ。

項と代入の演算は $t \cdot \sigma$ である。次のように定義される。

$$t = x \cdot \{x \leftarrow t\} \tag{1}$$
$$x = s \cdot \{x \leftarrow t\} \qquad if \, x \neq s \tag{2}$$

代入は、t に出現する v を t で置き換えた記号列を作る操作である。項自体は変数という概念を持たず、代入が変数を決定する。

表記上、$\bar{v}$ を変数のベクトル、$\bar{t}$ を term のベクトルとするとき、 $\bar{v} \leftarrow \bar{t}$ とも書く。このベクトルは記号の並びということ。

代入が変数を決定するのだが、特に、

$$\{x \leftarrow x\}$$

は変数の binding のみを定義する代入であり、$\{\overleftarrow{x}\}$ とも書く。

また、binding は ε でも表し、変数を明示する必要があれば

$$\varepsilon_{x,y}$$

などと書くつもりだが、たぶんそのような記法はこの後の部分では使わないだろう。

代入のもう一つの表現として

$$\sigma[t] = t \bullet \sigma$$

と定義する。

これは、代入が項を変換する関数であるという見方になる。

symbol に対する $\sigma[v]$ はあらかじめ定義されているので、$\sigma[t]$ はそれを項目に自然に拡張した機能になる。

代入操作の適用の別表現でもある。

$\sigma[t]$ の定義は次の通り。

$$\begin{aligned} \sigma[t] &:= \sigma[v] \qquad if\, t\, is\, a\, var\, v \\ \sigma[t] &:= \sigma[f](\sigma[t_1], ...) \qquad if\, t = f(t_1, ...) \end{aligned}$$

また、項の同値関係 $t_1 \approx t_2$ を、あとで定義する。♠

$$\sigma_1 \bullet t_1 \approx \sigma_2 \bullet t_2$$

変数を除いて同じというような意味にしたい。必要なければ定義しないが … どうかな。

4.0.1 インスタンス

項 s が項 t のインスタンスであるとは、ある代入 σ が存在し

$$t \geq s \iff \exists \sigma\, t = s \bullet s$$

となることと定義する。

4.1 代入間の演算

また、2 つの代入 σ_1, σ_2 の間の合成を次のように定義する。

$$\{x \leftarrow t\} \bullet \sigma_2 = \{x \leftarrow t \bullet \sigma_2\}$$

$\sigma = \sigma_1 \bullet \sigma_2$ は、代入を順番に適用した結果になる。つまり、任意の項 t について

$$t \bullet (\sigma_1 \bullet \sigma_2) \approx (t \bullet \sigma_1) \bullet \sigma_2$$

となる。

この $\bullet$ では、σ_1 と σ_2 に共通の変数がある場合、σ_2 のその変数への代入は無視される。

例)

$$\{x \leftarrow t\} \bullet \{x \leftarrow s, y \leftarrow u\} = \{x \leftarrow t, y \leftarrow u\}$$

Julia の Dict は代入に似ていて、$merge(d_1, d_2)$ はこの操作に似ているが、同じ変数があったときの処理が逆になる。

$$merge(\sigma_1, \sigma_2) == \sigma_2 \bullet \sigma_1$$

4.2　基本的な性質

定義から、

$$t \geq t \bullet \sigma$$

代入の合成は可換ではない。つまり、

$$\sigma_1 \bullet \sigma_2 = \sigma_2 \bullet \sigma_1$$

とは限らない

例 1

$$\{x \leftarrow a\} = \{x \leftarrow a\} \bullet \{x \leftarrow b\}$$

$$\{x \leftarrow b\} = \{x \leftarrow b\} \bullet \{x \leftarrow a\}$$

パラレルな合成の場合はどちらも fail する。

例 2

$$\{x \leftarrow f(b), y \leftarrow b\} = \{x \leftarrow f(y), y \leftarrow b\} \bullet \{y \leftarrow a\}$$

$$\{x \leftarrow f(a), y \leftarrow a\} = \{y \leftarrow a\} \bullet \{x \leftarrow f(y), y \leftarrow b\}$$

これも例 1 と同じか。

例 3

(作成中)

4.3　変数

代入 σ が与えられた時、その代入が決める変数の集合を定義する。

$$\mathbf{V}(\sigma) = \{v | \{v \leftarrow *\} \in \sigma\}$$

代入を実行すると、対象の項からはその変数が消えてしまうので、項に注目するとその変数は存在しないことになるが、操作としての代入にとっては意味がある。たとえば、resolution で resolvent に残るリテラルに対して代入を適用するとき、それらの変数こそが必要である。

4.3.1　全変数の集合

全変数の集合 ($\mathbf{V}$) を考えると、すべての代入の長さが同じになるので、話が簡単になりそうな気はする。同じ変数の index が同じになるので、変数は index で表せ、変数を探す処理が単純になる (今考えているように、Julia の Dict を使えば、もともと計算量はそれほど大きくないので、メリットはないだろう)

しかし、resolution のように、変数がどんどん増えていく場合、$\mathbf{V}$ は無限集合になる。

処理を考えている場合は、1 つの resolution の間は固定長さになるだろう。

5　項の unification

項 t, s と代入 σ があるとき、unification を $\sigma < t : s >$ と表記する。この代入は、t, s の変数の binding を定義するだけでなく、t, s に出現していない変数の binding を含みうる。

unification は代入を計算するが、代入が求められない場合は未定義 (ω) の値をとる。(Julia の実装では、値ではなく例外を返す)

代入 σ が ε の場合、これは通常の unification となる。

$$< t : s >= \varepsilon < t : s >$$

5.1　不変量 ♠

代入と項のペア (σ, t) は unification の処理の進行途中での不変量になるようなきがする。項 t,s の unification によって mgu σ が得られたとする。そして $r = t \bullet \sigma = s \bullet \sigma$ の場合、

$$(\sigma, r) =< t : s >$$

と書く。

$(t^{'}, s^{'}) \sqsubset (t, s)$ を 2 つの表現の間のパラレルな包含関係と定義する。

$(t^{'}, s^{'}) \sqsubset (t, s)$ であり、$t^{'} \neq s^{'}$ のときこの $(t^{'}, s^{'})$ を disagreement set と呼ぶ。

代入 σ のもとで、disagreement set $(t^{'}, s^{'})$ の片方が変数である場合 (ここでは t が変数だとする) に $\sigma^{'} = \{t \leftarrow s\}$ を定義すると

$$\sigma < t : s >= (\sigma \bullet \sigma^{'}) < t : s >$$

である。(本当か??)

5.2　disagreement set と代入 ♠

また、disagreement set を解消していくと、t と s の部分項のシーケンスができる。

$$t \supset t_1 \supset t_2 \supset \cdots \supset t_k$$

$$s \supset s_1 \supset s_2 \supset \cdots \supset s_k$$

それぞれの disagreement set について σ_i という代入を作ったとすると、

$$< \sigma_i : \sigma_{i+1} >< t_{i+1} : s_{i+1} >= \sigma_i \bullet < t_i : s_i > \bullet \sigma_{i+1}$$

みたいな関係になる。はず。

また、(t, s) のすべての disagreement set を Δ と書く。これらの disagreement set からそれぞれ求められた代入を σ_i と書くとき

$$\sigma =< \cdots << \sigma_1 : \sigma_2 >: \sigma_3 >: \cdots >: \sigma_k >$$

になるのではないか

5.3　代入は unification

代入 σ が 1 引数をとると、代入の適用になるが、2 引数をとると unification になる。

$$\sigma(t) = t \cdot \sigma$$
$$\sigma(t,s) = \sigma < t : s >$$

6　代入の unification

mgu は代入の一種だが、二つの term t_1, t_2 があったとき、それを同じにする代入の中で最も一般的なものである。この操作を term の間の unification と呼ぶ。ここでは、次のように表記する。

$$< t_1 : t_2 > \iff \sup\{\mu t_1 \cdot \mu = t_2 \cdot \mu\}$$

さらに、2 つの代入の間の unification 操作を次のように定義する。これは例であり、完全な定義は未。

$$< \{x \leftarrow t\} : \{x \leftarrow s\} > \iff \{x \leftarrow < t : s >\}$$

これによると

$$< \{x \leftarrow f(y), y \leftarrow y\} : \{x \leftarrow f(a)\} >= \{x \leftarrow f(a), y \leftarrow a\}$$

となる。

6.1　代入の unification の意味

代入の unification は、代入の間の・操作と型は同じだが、後者はシリアルな積であり、前者はパラレルな積になる。

その意味は、代入への代入と異なり、同じ変数への代入があったとき、片方の代入を無視しない。

6.2　変化する項

term の変化というものについて考える。

時間を t_1, t_2, t_3, t_4 とし、term のシーケンスがあるとする。

$$s_{t_1}, s_{t_2}, s_{t_3}, s_{t_4}$$

このとき、隣り合う term の mgu は、2 つの term の差を表している。

$$\sigma_{1,2} =< s_{t_1} : s_{t_2} >$$
$$\sigma_{2,3} =< s_{t_2} : s_{t_3} >$$
$$\sigma_{3,4} =< s_{t_3} : s_{t_4} >$$

そして、これらの mgu の間の mgu を求めると、それは差の差を表していると考えられる。

$$\mu_{1,2,3} =< \sigma_{1,2} : \sigma_{2,3} >$$
$$\mu_{2,3,4} =< \sigma_{2,3} : \sigma_{3,4} >$$

7 unification ♠

項 t, s の unification を $unification(t, s) = (\sigma, r)$ と書くとする。
ここで、$r = t \bullet \sigma = s \bullet \sigma$ である。
不変量にもっていきたいのだけれど、この定義だと、ちょっと話が展開しないような。

$$\begin{aligned} \sigma &= < t, s > \\ t \bullet \sigma &= s \bullet \sigma \end{aligned}$$

8 disagreement set

うまく定義できるだろうか。まだ必要かどうか不明。
項 t, s の dsagreement set $\Delta(t, s)$ の定義。
$\Delta(t, s) = \{(d_t, d_s) : t \sqsupset d_t \wedge s \sqsupset d_s \wedge \exists\delta(d_t \bullet \delta \wedge d_s \bullet \delta) \wedge ((d_1, d_2) \in \Delta(t, s) \rightarrow \nexists(d_1', d_2') \sqsupset (d_1, d_2)\})\}$
また、t, s から 1 つの disagreement set を取り出す関数 δ があるとする。

$$\delta(t, s) = (d_{t_1}, d_{s_1})$$

Δ 集合のどの要素を選ぶのかは決めない。選択する順番によって形が違ってくることにも注意。
(ここに mgu 構成の処理の流れの画像がはいる)