述語変換子意味論と余代数

こっとん (@CottonShampoo)

2025 年 12 月 20 日 土曜日

Mathematical Logic Advent Calendar 2025
の 20 日目の投稿です。

このドキュメントは，見やすく読みまちがえにくい
ユニバーサルデザインフォントを採用しています。

目次

0　前置き

0.1　この記事が想定する対象読者

0.2　この記事の４つの目標

この記事では、以下の４つのことを目標とします。

1.
2.
3.
4.

0.3　この記事では扱わないこと

1　ホーア論理と述語変換子

ホーア論理（Hoare logic）と**述語変換子**（predicate transformer）は、プログラムの正しさ（意図した挙動をするか、バグがないかどうか）を数学的に検証するための、異なる 2 つのアプローチです。プログラムの満たしていてほしい性質（仕様）を論理式で記述し、それが満たされているかを論理学的手法を用いて確かめます。

たとえば、以下のような「入力 N に対し、階乗 $N!$ を計算して出力するプログラム」を考えてみましょう。

```
n := N
k := 1
while (n>0) {
  k := k*n;
  n := n-1;
}
```

N に正の自然数を入力してこのプログラムを実行した後に、事後条件「$k = N!$」を満たしてほしいものとします。ループが何回実行されるかわからない中で、あらゆる入力 N に対してどうやって保証すればよいでしょうか。答えは、各回のループを通過する前後でいつも保存される条件（ループ不変量）「$k \cdot n! = N!$」に注目する、というものです。そうすることで「階乗を出力する」という仕様が必ず満たされることを保証できます。つまり、事後条件を保証するためには、ループ不変量を探せばよいのです。これが**ホーア論理**の考え方です。

では今度は、この事後条件「$k = N!$」を成り立たせるためには事前条件として何を課す必要があるか、という問題を考えてみましょう。答えは、「$N \geq 0$」です。これを求めるには、事後条件から出発してプログラムを後ろから前へたどり、各代入に応じて条件を置き換える、という操作をする必要があります。ただし、ループは何回実行されるかわからないので、この操作について不動点をとる必要があります。これが**述語変換子**の考え方です。

1.1　ホーア論理

ホーア論理では $\{P\}\,C\,\{Q\}$ という三つ組を使います。これは「事前条件 P から始めて、プログラム C を実行した結果、事後条件 Q が成り立つ」という関係です。

定義 1　ホーア論理の導出規則は以下のものからなる。

$$\frac{}{\{A\}\ \mathsf{skip}\ \{A\}}\ (\mathsf{skip})$$

$$\frac{}{\{A\,[a/x]\}\ x := a\ \{A\}}\ (\mathsf{assignment})$$

$$\frac{\{A\}\ P_1\ \{C\} \qquad \{C\}\ P_2\ \{B\}}{\{A\}\ P_1; P_2\ \{B\}}\ (\mathsf{sequential\ composition})$$

$$\frac{\{A \wedge b\}\ P_1\ \{B\} \qquad \{A \wedge \neg b\}\ P_2\ \{B\}}{\{A\}\ \mathsf{if}\ b\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2\ \{B\}}\ (\mathsf{if})$$

$$\frac{\{A \wedge b\}\ P\ \{A\}}{\{A\}\ \mathsf{while}\ b\ \mathsf{do}\ P\ \{A \wedge \neg b\}}\ (\mathsf{while})$$

$$\frac{A \Rightarrow A' \qquad \{A'\}\ P\ \{B'\} \qquad B' \Rightarrow B}{\{A\}\ P\ \{B\}}\ (\mathsf{consequence})$$

例 2 たとえば

$$\dfrac{\dfrac{x{\geq}0{\wedge}x{>}0 \Rightarrow x{-}1{\geq}0 \qquad \dfrac{}{\{x{-}1{\geq}0\}\ x{:=}x{-}1\ \{x{\geq}0\}}\ (a)}{\{x{\geq}0{\wedge}x{>}0\}\ x{:=}x{-}1\ \{x{\geq}0\}}\ (c) \qquad \dfrac{x{\geq}0{\wedge}\neg(x{>}0) \Rightarrow x{\geq}0 \qquad \dfrac{}{\{x{\geq}0\}\ x{:=}x\ \{x{\geq}0\}}\ (a)}{\{x{\geq}0{\wedge}\neg(x{>}0)\}\ x{:=}x\ \{x{\geq}0\}}\ (c)}{\{x{\geq}0\}\ \text{if } x{>}0 \text{ then } x{:=}x{-}1 \text{ else } x{:=}x\ \{x{\geq}0\}}\ (i)$$

注意 ループ不変量とは、以下のような while 文の証明図に現れる I のこと。つまり、

1. ループの入口で事前条件によって含意され（$A \Rightarrow I$）、
2. 各回のループ実行時に常に保たれ（$\{I \wedge \varphi\}\, C\, \{I\}$）、
3. ループの出口で事後条件を含意する（$I \wedge \neg\varphi \Rightarrow B$）

ような論理式 I のことである。

$$\dfrac{A \Rightarrow I \qquad \dfrac{\{I \wedge \varphi\}\, C\, \{I\}}{\{I\}\ \text{while } \varphi \text{ do } C\ \{I \wedge \neg\varphi\}}\ (\text{while}) \qquad I \wedge \neg\varphi \Rightarrow B}{\{A\}\ \text{while } \varphi \text{ do } C\ \{B\}}\ (\text{conseq})$$

先ほどの「階乗を計算するプログラム」のホーア論理による証明図を書くと（長くなるので省略するが）I のところに現れる論理式は「$k \cdot n! = N!$」となる。これが先ほどのプログラムの while 文の「ループ不変量」にあたる。

1.2 述語変換子

述語変換子は、述語 Q を別の述語 $wp[\![C]\!](Q)$ に変換します。これは「プログラム C について、事後条件 Q を成り立たせる事前条件のうち、論理的含意関係に関して最も弱いもの」、**最弱事前条件** (weakest precondition) です。

定義 3（述語変換子意味論） 状態集合を S とする。S 上の述語の集合を $\mathrm{Pred}(S) = \{Q \mid Q : S \to \{0,1\}\}$ と定める。なお、各述語 $Q : S \to \{0,1\}$ は、$Q = \{s \in S \mid s \models Q\}$ だと考えればよい。述語どうしの順序は、含意関係によって定めるものとする。そうすると、$(\mathrm{Pred}(S), \leq)$ は完備束をなす。

$$P \leq Q \quad \text{iff} \quad P \Rightarrow Q$$

プログラム C の述語変換子 (predicate transformer) とは、写像

$$wp[\![C]\!] : \mathrm{Pred}(S) \to \mathrm{Pred}(S)$$

であって単調性 $Q_1 \leq Q_2 \;\Rightarrow\; wp[\![C]\!](Q_1) \leq wp[\![C]\!](Q_2)$ を満たすものである。具体的には以下のように帰納的に定義される。

- $wp[\![\text{skip}]\!](Q) = Q$
- $wp[\![x := a]\!](Q) = Q[a/x]$
- $wp[\![C_1; C_2]\!](Q) = wp[\![C_1]\!](wp[\![C_2]\!](Q))$
- $wp[\![\text{if } \varphi \text{ then } C_1 \text{ else } C_2]\!](Q) = (\varphi \wedge wp[\![C_1]\!](Q)) \vee (\neg\varphi \wedge wp[\![C_2]\!](Q))$
- $wp[\![\text{while } \varphi \text{ do } C]\!](Q) = \mu X.\ \underbrace{((\varphi \wedge wp[\![C]\!](X)) \vee (\neg\varphi \wedge Q))}_{\text{loop characteristic function } \Phi_Q(X)}$

注意 ここで μ は順序 $\leq$ についての最小不動点をとる操作である。$\Phi_Q : \mathrm{Pred}(S) \to \mathrm{Pred}(S)$ はスコット連続なので、クリーネの不動点定理より、$\mu X.\, \Phi_Q(X) = \bigvee_{n \in \mathbb{N}} \Phi_Q^n(\bot)$ が得られる。

以上のように、ホーア論理と述語変換子という２つのアプローチがあります。ここで両者を整理しておきましょう。

$$\begin{cases} \{P\}\, C\, \{Q\} & \text{Hoare-style} \\ P \Rightarrow wp[\![C]\!](Q) & \text{Dijkstra-style} \end{cases}$$

両者の関係としては、以下が同値になることが知られています。

$$\{P\}\, C\, \{Q\} \quad \text{iff} \quad P \Rightarrow wp[\![C]\!](Q)$$

より詳しく見ると、実は以下の違いがあります。

- ホーア三つ組は「関係的」(relational)。
  各述語 P に対し、$\{P\}\, C\, \{Q\}$ が成り立つような述語 Q は複数ありうる。
  各述語 Q に対し、$\{P\}\, C\, \{Q\}$ が成り立つような述語 P は複数ありうる。
- 最弱事前条件は「関数的」(functional)
  各述語 Q に対し、述語 $wp[\![C]\!](Q)$ は一意に定まる。

他方、wp から Hoare triple を「再現」することができます。

- $\{\, wp[\![C]\!](Q)\,\}\, C\, \{Q\}$ はいつでも成り立つ。

さて、これら Hoare-style と Dijkstra-style には、それだけでなく、より本質的な違いがあります。それは、部分正当化 (partial correctness) か、全正当性 (total correctness) か、の違いです。

1.3 部分正当性と全正当性

プログラムの正しさを検証するうえで、**部分正当性** (partial correctness) と**全正当性** (total correctness) という２種類の「正しさ」の考え方があります。ホーア三つ組 $\{P\}\, C\, \{Q\}$ は前者（部分正当性）を主張するものです。ここではさしあたり、後者（全正当性）のほうを $[P]\, C\, [Q]$ と表記して区別することにして、両者の違いを比べてみましょう。

- 部分正当性 (partial correctness) $\{P\}\, C\, \{Q\}$
  ▷「事前条件 P から始めて、もしプログラム C が停止したら、その実行結果は事後条件 Q を満たす。」
  ▷ プログラムが停止しないときには何も保証しない。
- 全正当性 (total correctness) $[P]\, C\, [Q]$
  ▷「事前条件 P から始めて、プログラム C が必ず停止し、かつ、その実行結果は事後条件 Q を満たす。」
  ▷ プログラムの停止性 (termination) も保証する。

ホーア三つ組は部分正当性 (partial correctness) なのに対し、述語変換子は全正当性 (total correctness) です。この違いは、while ループの不動点のとり方に端を発しています。述語変換子の while の意味論は、最小不動点で与えられていたことを思い出しましょう。

$$wp[\![\text{while } \varphi \text{ do } C]\!](Q) = \mu X.\ \underbrace{((\varphi \wedge wp[\![C]\!](X)) \vee (\neg\varphi \wedge Q))}_{\text{loop characteristic function } \Phi_Q(X)}$$

それに対し、ホーア論理の while 文は、実は、最大不動点によって与えられていたのです！

$$\dfrac{P \Rightarrow I \qquad \dfrac{\{I \wedge \varphi\}\, C\, \{I\}}{\{I\}\ \text{while } \varphi \text{ do } C\ \{I \wedge \neg\varphi\}}\ (\text{while}) \qquad I \wedge \neg\varphi \Rightarrow Q}{\{P\}\ \text{while } \varphi \text{ do } C\ \{Q\}}\ (\text{conseq})$$

命題 4　L を完備束とし、$f : L \to L$ を単調写像とする。このとき、以下の帰結関係（上から下）が成り立つ。

$$\frac{p \leq i \qquad i \leq f(i) \qquad i \leq q}{p \leq \nu(f(-) \wedge q)}$$

証明　$\wedge$ の普遍性より、以下の同値関係が成り立つ。

$$\frac{i \leq f(i) \qquad i \leq q}{i \leq f(i) \wedge q}$$

ナスター・タルスキの定理より、

$$\nu(f(-) \wedge q) = \bigvee \{x \in L \mid x \leq f(x) \wedge q\}$$

であるから、特に任意の $i \in L$ について以下の帰結関係が成り立つ。

$$\frac{i \leq f(i) \wedge q}{i \leq \nu(f(-) \wedge q)}$$

ここで、条件 $p \leq i$ と推移性から、最終的に

$$\frac{p \leq i \qquad i \leq f(i) \qquad i \leq q}{p \leq \nu(f(-) \wedge q)}$$

が得られた。 □

ここで、p は事前条件、q は事後条件、i はループ不変量に、おおよそ相当しています。いま、

- $\boxed{f(i)}$ を $\boxed{\varphi \Rightarrow wp[\![C]\!](I)}$ に読み替える
- $\boxed{q}$ を $\boxed{\neg\varphi \Rightarrow Q}$ に読み替える

とすれば、先ほどの while ループの規則に対応していることがわかりやすいのではないかと思います。

$$\frac{p \leq i \qquad i \leq f(i) \qquad i \leq q}{p \leq \nu(f(-) \wedge q)} \quad \rightsquigarrow \quad \frac{P \Rightarrow I \qquad \dfrac{\{I \wedge \varphi\}\, C\, \{I\}}{\{I\}\,\text{while } \varphi \text{ do } C\, \{I \wedge \neg\varphi\}} \qquad I \wedge \neg\varphi \Rightarrow Q}{\{P\}\,\text{while } \varphi \text{ do } C\, \{Q\}}$$

あえて述語変換子の言葉に寄せると、以下のように書くこともできます。

$$P \Rightarrow \nu X.\, ((\varphi \Rightarrow wp[\![C]\!](X)) \wedge (\neg\varphi \Rightarrow Q))$$

つまり、同じものについて、ホーア論理では最大不動点 (gfp, ν) を、最弱事前条件では最小不動点 (lfp, μ) をとっていたことがわかります。(なお、述語変換子の中にも、while の部分正当化を行うものもあり、最弱リベラル事前条件 (weakest liberal precondition) と呼ばれます。)

以上のことをまとめると、以下の表の通りです。

	Hoare logic	predicate transformer
gfp	partial correctness	weakest liberal precondition
lfp	total correctness	weakest precondition

さて、この節で扱った述語変換子の考え方は、圏論的に扱うととても見通しがよくなります。そのための準備として、次節では、圏論の関手を使ってプログラムの振舞いを捉える方法について紹介します。

２　圏論の関手でシステムの振舞いを捉える

3　最弱事前条件をどのように手に入れるか

参考文献

[1]