様相論理の証明支援系上の実装

矢田部俊介

産業技術総合研究所システム検証研究センター

2008 年 3 月 3 日

要旨

- **目的: 知識様相論理の意味論を証明支援系に実装**
- **二つの問題点**
  - (1) **多変数述語を持つ型のデータ型の定義が難しい**
  - (2) **充足関数の再帰的定義が停止判定に合格しない**
- **解決法**
  - (1) **ベクトルを使用**
  - (2) **論理式のデータ型の情報量を増やす**
    $P(x, y)$ 二変数述語とする。論理式 $(\forall x)P(x, y)$ のデータ型として
    - **従来: Fml 1 （一変数述語）**
    - **今回: $V_{2,1}$ （二変数のうち一変数が束縛されている）**

    この増えた情報量によって、帰納法を行うことで、停止判定を合格させる定義が可能になる

使用する証明支援系 Agda

- Agda: Martin-Löf の直観主義的型理論に基づいた証明支援系
- 述語論理のかなりの部分（命題論理のすべてと、述語論理の一部分）をそのまま定義することが可能である。
- 述語論理における論理的帰結関係と全称量化子の取り扱い：
  - 命題 p は $p : \mathrm{Set}$ として表現される
  - 命題（集合）A, B に関して、$A \to B$ は、集合 A から B への関数として定義される。
  - 全称量化子 $(\forall x \in D)p(x)$ は、$(x : D) \to px$ と、依存型として定義される。

Agda の特徴

- **依存型理論と一般の再帰法** (general recursion) **は相性が悪い。**

$$\frac{f : \forall x : \mathrm{N} \to T[x], \quad s : \mathrm{N}}{f\,s : T[s]}$$

  - $f(2+2)$ は $f(4)$ と型が一致し、また $T[2+2]$ は $T[4]$ と型が一致する必要がある。
  - この計算が停止しない場合、証明支援系は型付けに失敗する。

- Agda では依存型理論に基づきながら、原始再帰法による定義だけでなく、一般の再帰法による定義を許す。
  - 型検査の終了後に、別の検査「停止判定」を行う。
  - その関数の型の判定に計算が含まれている場合、その計算が停止するかどうかを判定する。
- Agda では停止判定にパスするかどうかが重要

実装の対象: 知識様相論理の体系の充足関数

- 実装の対象となる体系: 知識様相論理
  - この知識様相論理の体系は、伝聞に関する知識を表現する様相論理の体系である。
  - ネットワーク上の認証の安全性証明への応用が期待される。
- 表記法: 全称量化子を forall 、論理的帰結関係 $\Rightarrow$
- 充足関数は、述語論理の上で再帰的に定義されている: 全称量化子（forall） のケース

$$\langle \overline{s}, \varphi, \psi \rangle \models (\text{forall } x) P(x)$$

$$\Longleftrightarrow \langle \overline{s}, \varphi \wedge ((\exists y) Q(y) \rightarrow Q), \psi \wedge ((\exists y) R(y) \rightarrow R) \rangle \models P(y)$$

が任意の $Q(x), R(x)$ に成立する

- 人間の目には、この再帰法が有限回で停止するのは明らかである（この点が Agda にとっての問題となる）。

ポイント: 必要なこと

ポイントになるのは、以下の二点が要求されるということである。

- 述語論理の実装：
  任意有限個の変数をもつ述語を、変数を束縛する機構（量化子・個体記号の変数への代入操作）を持つデータ型として定義すること。

- 知識様相論理の充足関数の再帰的定義：
  量化子を持つ文を代入操作を持つ文に還元することにより、述語論理上で関数を再帰的に定義し、そしてこの定義が帰納的定義になっていることを証明支援系に理解させる（型検査と停止判定にパスする）こと。

問題点その一：任意有限個の変数を持つ述語のデータ構造を定義することの困難（述語論理）

- 通常、Agda では以下のように述語を構成する。
  - p が 1 変数述語（domain として D をとる）場合は $p : D \to \mathbf{Set}$ と型付けする。
  - p' が 2 変数述語の場合は $p' : D \to D \to \mathbf{Set}$, etc.

-
  - しかし、以下のデータ型 $\mathbf{Pred}\ n$ は型検査に合格しない (Ver. 2.0.0)。
    - $\mathbf{Pred}\ 0 = \mathbf{Set}$
    - $\mathbf{Pred}\ n + 1 = D \to \mathbf{Pred}\ n$
  - 右辺 $(D \to \mathbf{Pred}\ n)$ の値域 $(\mathbf{Pred}\ n)$ が明示的に（Set などの）ソート（Sort）になっていないと合格しない。

- 量化・代入の順番が一通りに限られる（$\forall y \exists x P(x, y)$ などが明示的に表現できない）

問題点その二：関数の一般再帰的定義の停止判定にまつわる困難（様相論理）

- ⊨ の定義: 左辺を右辺に還元している

$$\langle \bar{s}, \varphi, \psi \rangle \models (\text{forall } x) P(x)$$
$$\iff \langle \bar{s}, \varphi \wedge ((\exists y) Q(y) \to Q), \psi \wedge ((\exists y) R(y) \to R) \rangle \models P(y)$$
が任意の $Q(x), R(x)$ に成立する

- しかし、証明支援系にとっては、forall x Px も subst y P も、データの複雑さの点では同じ。

- そのため、前者のケース ⊨ forall x p を後者のケース ⋯ ⊨ subst y P に還元する上記の再帰的定義は、Agda の停止判定メカニズムが循環的定義であると判断してしまう。

解決法その一：ベクトルによる述語のデータ型の定義（述語論理）

- 型検査に合格するため、多変数の述語は $X \to \mathrm{Set}$ の形で定義しなければならない。
- そのため、n 変数述語の解釈のデータ型 Pred' を、
  - $\mathrm{N} \to \mathrm{Set1}$ の依存型を持ち、
  - $\mathrm{Pred}'\,n = D^n \to \mathrm{Set}$ （D の元の長さ n のベクトル D^n を定義域とし Set を値域とする関数）として定義する。

論理式の定義（述語論理）

	オブジェクト	データ型	データ型のソート
解釈	Q :	$\mathrm{Pred}'(n+1)$:	Set1 (Set1 ≠ Set)
述語記号	q :	PredSym $(n+1)$:	Set
論理式	fml q :	Fml $(n+1)$:	Set
代入操作	fml-subst t q	Fml n	
量化	$\forall x$ q	Fml n	

ベクトル操作のための関数の定義

- ベクトルを領域として述語を定義する場合は、以下が問題になる。
  - Q を 2 変数述語とする。片方の変数に $t : D$ を代入した結果は 1 変数述語 $Q(x, t)$ となる。
  - しかし $q : D^2 \to \mathrm{Set}$ では、$t : D$ のみを代入できない。
- 解決策は以下の通り。
  - ベクトルを操作する関数 $\mathrm{gt}\,(k, \vec{t}, c)$ はベクトル $\vec{t}$ の k 番目の箇所に、新しい個体 c を付け加える。
  - 以下のように述語記号 $p : \mathrm{PredSym}\ n + 1$ に対して、以下のように解釈 int の値を定める。
    - 代入操作: $\mathrm{int}\,(\text{fml-subst}\ t\ p) = (ts : D^n) \to (\mathrm{int}\ p)(\mathrm{gt}\,(0, ts, t))$
    - 全称量化子: $\mathrm{int}\,(\forall x\ p) = (ts : D^n) \to (\mathrm{int}\ p)(\mathrm{gt}\,(0, ts, y))$、ただし y はフレッシュな変数（を表す D の元）とする。

解決法その二：論理式のデータ型の情報量を増やす（様相論理）

- データ型 $V : \{n : \mathrm{N}\} \to \{m \leq n\} \to \mathrm{Set}$ を定義する。
  - n: 述語の変数の個数
  - m: 束縛変数の個数

  V は、コンストラクター $\mathrm{trans} : \{n : \mathrm{N}\} \to \mathrm{PredSym}\ n \to V_{n,0}$ を持つ。

通常の表記	オブジェクト	データ型
解釈	$\mathrm{int}\ (q)$	$\mathrm{Pred}'2$
述語記号	q	$\mathrm{PredSym}\ 2$
$q(\_, \_)$	$\mathrm{trans}\ q$	$V_{2,0}$
$\forall x\ q(x, \_)$	$\mathrm{forall}\ x\ (\mathrm{trans}\ q)$	$V_{2,1}$
$\forall x\ q(x, t)$	$\mathrm{subst}\ t\ \mathrm{forall}\ x\ (\mathrm{trans}\ q)$	$V_{2,2}$

充足関数 Sat の定義

- $\mathrm{Sat}(\vec{t}, \varphi, \psi, n, n, P, ())$ は、$\models$ の表記法では
  - $\langle \vec{t}, \varphi, \psi \rangle \models P$ であり、
  - また $P : V_{n,n}$ の閉論理式である

  ことを表現する。

充足関数の計算の実例

$(\forall x)Q(x, t)$ に対応する論理式 subst t forall x (trans q) を考える。

1. まず外側の subst t を除去

$$\mathrm{Sat}((\vec{t}, \varphi, \psi, 2, 2, (\mathrm{subst}\ t\ \mathrm{forall}\ x\ (\mathrm{trans}\ q)), ()) = \mathrm{Sat}((\vec{t}, \varphi, \psi, 2, 1, (\mathrm{forall}\ x\ (\mathrm{trans}\ q)), (t))$$

2. 次に forall x を除去

$$\mathrm{Sat}((\vec{t}, \varphi, \psi, 2, 1, (\mathrm{forall}\ x\ (\mathrm{trans}\ q)), (t)) = (n : \mathrm{N}) \to (Q, R : \mathrm{Fml}\ n) \to \mathrm{Sat}((\vec{t}, \varphi \wedge ((\exists y)Q(y) \to Q), \psi \wedge ((\exists y)R(y) \to R), 2, 0, (\mathrm{trans}\ q), (y, t))$$

3. $m = 0$ なので終了処理（$\mathrm{int}\ (q) : D^2 \to \mathrm{Set}$）

$$\mathrm{Sat}((\vec{t}, \varphi \wedge \ldots, \psi \wedge \ldots, 2, 0, (\mathrm{trans}\ q), (y, t)) = (\mathrm{int}\ (q))(y, t)$$

有限ステップの変形で $(\mathrm{int}\ (q))(y, t) : \mathrm{Set}$ が値として出力される。

停止判定に合格する理由

- forall と subst はどちらも $V_{n,m} \to V_{n,m+1}$ の型を持つ。除去することで、添え字が $m+1$ から m へと明示的に 1 減る。
- $m = 0$ のとき、必ず終了処理をして、計算は停止する。
- このことが、再帰法による計算が、有限回で必ず停止することを明示的に保証する。このことは Agda も理解することができ、Agda の停止判定に合格することができた。

まとめ

- 結果: 知識様相論理の意味論を証明支援系に実装した。
- 実装中、二つの問題点が見つかった。
  (1) 多変数述語を持つ型のデータ型の定義が難しい
  (2) 充足関数の再帰的定義が停止判定に合格しない
- 以下のように解決した。
  (1) 多変数述語をベクトルを使用して定義した。
  (2) 論理式のデータ型の情報量を増やした。
    - $V_{n,m}$ として、n 変数述語のうち m 個の変数が束縛されていることを表現するようにした。
    - この増えた情報量によって、帰納法を行うことで、停止判定を合格させる定義が可能になった。