計算機科学における様相論理

佐藤雅彦

京都大学情報学研究科

数学基礎論サマースクール
神戸大学
2015 年 8 月 20 日

素朴な疑問

何故計算機科学において様相論理が使われるのか?

素朴な疑問

何故計算機科学において様相論理が使われるのか?

- 計算機科学の中心課題は「計算」という行為の解析.

素朴な疑問

何故計算機科学において様相論理が使われるのか?

- 計算機科学の中心課題は「計算」という行為の解析.
  $\Longrightarrow$
  「時間」と「空間」の中に置かれた「対象」(エージェントやプログラムといった データ) の動的な振舞いを解析する必要がある.

素朴な疑問

何故計算機科学において様相論理が使われるのか?

- **計算機科学の中心課題は「計算」という行為の解析.**
  $\Longrightarrow$
  「時間」と「空間」の中に置かれた「対象」(エージェントやプログラムといった データ) の動的な振舞いを解析する必要がある.
- **解析の道具として,「適度な抽象度」を持つ** Kripke **モデルが有用である.**

素朴な疑問

何故計算機科学において様相論理が使われるのか?

- **計算機科学の中心課題は「計算」という行為の解析.**
  $\Longrightarrow$
  「時間」と「空間」の中に置かれた「対象」(エージェントやプログラムといった データ) の動的な振舞いを解析する必要がある.
- **解析の道具として,「適度な抽象度」を持つ** Kripke **モデルが有用である.**
- Kripke **モデル という「意味論的枠組」を様相論理の言語を用いて「証明論」により形式的,論理的な解析ができるようになる.(健全性と完全性が要求される)**

素朴な疑問

何故計算機科学において様相論理が使われるのか?

- 計算機科学の中心課題は「計算」という行為の解析.
  $\Longrightarrow$
  「時間」と「空間」の中に置かれた「対象」(エージェントやプログラムといった **データ**) の動的な振舞いを解析する必要がある.
- 解析の道具として,「**適度な抽象度**」を持つ Kripke モデルが有用である.
- Kripke モデル という「意味論的枠組」を様相論理の言語を用いて「**証明論**」により形式的,論理的な解析ができるようになる. (健全性と完全性が要求される)
- $\Box$ が (implicit な) **変数**の適切な有効範囲 (scope) を提供する.

本講演の目標

計算機科学における様相論理の応用について，以下の事例を通して紹介する．

1. **証明**に関連する様相
   - 知識の論理 (知識の改訂，知識の共有)
   - 証明支援系, 構成的プログラミング (Reflective Proof Theory)
2. **変数**に関連する様相
   - プログラムの意味論 (referential transparency)
   - 環境および文脈計算 (Explicit Environment)
   - メタ変数
3. **データ**に関連する様相
   - 多段階計算
   - 安全性 (アクセス権限，情報流解析)

McCarthy の知識の公理化

「不貞の妻達のパズル」の問題設定.

ある王国で，100 万組の夫婦のうち 40 人の妻が不貞であった．ある日 (1 日目とする)，王様が以下の宣告をした.

1. **すくなくとも一人の不貞の妻がいる.**
2. **どの夫も，他の夫の妻が不貞かどうかを知っている.**
3. **今日から毎夜，すべての夫はそれまでの自分の知識を用いて，自分の妻が不貞かどうかを推論せよ.**
4. **もし，自分の妻が不貞であることを推論できたら，翌朝その妻の首を刎ねよ.**
5. **どの夫も毎朝誰かが妻の首を刎ねるかどうかを見よ.**

McCarthy の知識の公理化 (続)

このパズルの主要な構成要素は，

1. **ある空間 (王国) における時間の経過と，**
2. **それに伴なうエージェント (すべての夫) の知識の変化および，**
3. **その知識から帰結する首を刎ねるという 行為である．**

したがって，このパズルを形式論理を用いて解析するのに様相論理を用いることが考えられる． McCarthy [McCarthy-S-Hayashi-Igarashi 1978] **はパズルを形式的に記述するための様相演算子を与えて，それらが満たすべき** Hilbert **流の公理体系も与えた．**

[S 1977] **は，体系を** Gentzen **流に，** sequent calculus **として定式化し，対応する** Kripke **意味論のもとでの，健全性，強完全性を証明してパズルの形式的な解答を与えた．**

McCarthy の知識の公理化 (続)

形式体系の言語

- 王国の夫妻が k 組あるとし，それぞれの夫を S_i $(i = 1, \ldots, k)$ で表し，$[S_i]_n P$ は「S_i が n 日目に P を知っている」という命題を表わすとする．
- 命題定数 p_i $(i = 1, \ldots, k)$ を用意し，「S_i の妻が不貞である」という命題を表すとする．
- ideal なエージェントとして Fool を導入し，O で表す．このとき，$[O]_n P$ は「すべての夫が n 日目に P を知っている」という命題を表わすとする．common knowledge を表現するための McCarthy の秀逸な工夫である．

McCarthy の知識の公理化 (続)

記法

1. $\mathbf{A} := \{p_1, \ldots, p_k\}$.
2. $\mathbf{S} := \{S_1, \ldots, S_k, O\}$.
3. S は O または S_i を表す.
4. P は A, S 上の (様相) 命題を表す
5. $\langle S \rangle_n P := \neg [S]_n \neg P$.
6. $\{S\}_n P := [S]_n P \vee [S]_n \neg P$.

McCarthy の知識の公理化 (続)

公理化

各様相演算子は，S5 **の公理 (のスキーマ)**

- $[S]_n P \to P$.
- $[S]_n (P \to Q) \to [S]_n P \to [S]_n Q$.
- $[S]_n P \to [S]_n [S]_n P$.
- $\neg [S]_n P \to [S]_n \neg [S]_n P$.

と推論規則を満たすとし，更に以下の公理 (のスキーマ) を設ける．

- $[S]_n P \to [S]_m P \quad (n < m)$.
- $[O]_n P \to [O]_n [S]_n P$.

McCarthy の知識の公理化 (続)

Kripke モデル

W を空でない (可能世界の) 集合とするとき，W 上の Kripke モデル $\langle W; r, v\rangle$ を以下の条件で定める．

1. $r : \mathbf{S} \times \mathbb{N}^+ \to \wp(W \times W)$.
2. **r は W 上の同値関係．**
3. $m < n \Longrightarrow r(S, m) \supseteq r(S, n)$.
4. $r(O, n) \supseteq r(S, n)$.
5. $v : \mathbf{A} \times W \to \{\top, \bot\}$.

とくに，このパズルに対応するため，

1. **$W := \{+, -\}^k - \{(-, \ldots, -)\}$ と定め，**
2. **また，$w = (\varepsilon_1, \ldots, \varepsilon_k)$ に対して $[w] := \wedge_{i=1}^{k} p_i^{\varepsilon_i}$ と置く．**

McCarthy の知識の公理化 (続)

王様の宣告をこの体系でどう表現するかが問題

- **王様の宣告は，すべての夫がその内容を知った上で従うことを要求される，言語行為である．**
- **王様の宣告は，すべての夫が共有する** common knowledge **となる．**
- **どの夫の知識も毎日拡大するが，「どのように拡大するか」は** common knowledge **である．**
- common knowledge **と，各夫の知識との関係を正確に記述する必要がある．**
- **宣告は自己言及的である．**

McCarthy の知識の公理化 (続)

王様の宣告を適切に表現する論理式の集合 Γ を次のようにして定めた.

1. $P \in \Gamma$ は 1 日目の common knowledge なので $[O]_1 P$ はモデルの各 w で成立すべきである.
2. そのような Γ が存在すると仮定.
3. 夫 S_i の n 日目の知識全体を deductive に生成できる論理式の集合 $B_w(i, n)$ を Γ を用いて定義.
4. Γ を $B_w(i, n)$ を用いた集合として特徴付ける.

$B_w(i, n)$ の「定義」に Γ を用いているので，このままでは，Γ の定義として不完全.

McCarthy の知識の公理化 (続)

$$\Gamma = \bigvee_{i=1}^{k} p_i \cup \{\{S_i\}_1 p_j \mid j \neq i\} \cup$$
$$\{[w] \to [O]_{n+1}[S_i]_n p_i \mid B_w(i,n) \vdash p_i\} \cup$$
$$\{[w] \to [O]_{n+1}\neg[S_i]_n p_i \mid B_w(i,n) \nvdash p_i\} \cup$$
$$\{[w] \to [S_i]_n P \mid B_w(i,n) \vdash P\} \quad (1)$$

$$B_w(i,1) = [O]_1\Gamma \cup \{p_j^{w_j} \mid j \neq i\} \quad (2)$$

$$B_w(i,n+1) =$$
$$B_w(i,n) \cup \{[S_j]_n p_j \mid B_w(j,n) \vdash p_j\} \cup$$
$$\{\neg[S_j]_n p_j \mid B_w(j,n) \nvdash p_j\} \quad (3)$$

(1), (2), (3) **を未知数** Γ, $B_w(i,n)$ $(i = 1, \ldots, k,\ n \in \mathbb{N}^+)$ **に関する無限個の連立方程式系と見ることができる.**

McCarthy の知識の公理化 (続)

Theorem

方程式系 (1),(2),(3) **に対して** Γ **が無矛盾になるような解** Γ, $B_w(i,n)$ **が唯一存在する．**

Theorem

上の解 $B_w(i,n)$ **に対応して，すべての** P **について，**

$$B_w(i,n) \vdash P \iff w \models [S_i]_n P$$

となる Kripke **モデルが一意的に定まる．**

McCarthy の知識の公理化 (続)

前記の Kripke **モデルは以下のように定められる.**

- $W := \{+,-\}^k - \{(-,\ldots,-)\}$.
- $(w,w') \in r(i,n) \iff w = w'$ **または** $w \oplus w' = e_i$ **かつ** $n < \max\{\|w\|, \|w'\|\}$.
- $r(O,n) := r(S_i,n)$ **をすべて含む最小の同値関係**.
- $v(p_i,(\varepsilon_1,\ldots,\varepsilon_k)) = \top \iff \varepsilon_i = +$.

このモデルで以下が成立する.

1. $w_i = +$ **のとき** $w \models [S_i]_n p_i \iff \|w\| \geq n$.
2. $w_i = -$ **のとき** $w \models [S_i]_n \neg p_i \iff \|w\| > n$.

Reflective Proof Theory

Aczel によるフレーゲ構造に「証明」を対象として追加した構造を定義し，その上で自己反映的な証明の理論 (RPT) を展開した [S 1991]．

- フレーゲ構造は，命題 (propositions) とその一部としての真命題 (truths) を型の無い λ 計算のモデルの内部に実現したもの．
- Aczel はこれにより，フレーゲが「算術の基本法則 (Grundgesetze der Arithmetics)」集合を概念の外延として定義しようとして失敗した試みを修復しようとした．
- RPT はフレーゲ構造に，証明を対象として追加し，更にこれらの対象について RPT の内部で言及できるような構文的対象を持つようにした．

以下では RPT の概略を紹介する．

RPT の構成

- Λ を，対の構成子の他に論理記号，等号，自然数等を定数として持つ λ-計算とし，$\Lambda_=$ をその項モデルとする．
- $\Lambda_=$ の対象 a, b が等しいことを $Eq(a, b)$ で表し，RPT では $a = b$ と書く．
- $\Lambda_=$ の上に，自然数 i をパラメータとする 1 項関係 $Prop_i(a)$ と 2 項関係 $Proves_i(p, a)$ を相互再帰的に定義する．
- 上の 2 つの関係を，RPT の内部では，$\models_i a$ および $p \vdash_i a$ と書く．
- $\Lambda_=$ の関数適用を $App(f, a)$ で表し，RPT では $f(a)$ と書く．
- $a = b, \models_i a, p \vdash_i a, f(a)$ 等はすべて RPT の項であり，同時に $\Lambda_=$ の対象を表す．

RPT の特徴

1. RPT が扱う対象は，すべて有限の対象であり，その中には命題や証明が含まれる．
2. RPT は形式的体系ではなく，命題や証明の意味は意味論的に与えられる．
3. 意味論は Martin-Löf 流の直観主義的意味論として与えることもでき，また，古典論理の上での集合論的意味論を与えることもできる．
4. したがって，不完全性定理により RPT の完全な形式化はできないが，実用的に十分な形式化をすることはできる．
5. 全称量化，特称量化は Church の方法で λ-項として表現する．
6. 証明も，Curry-Howard の同型を用いて λ-項として表現する．
7. 「レベル i の命題 a が証明可能である」という命題は，「$\vdash_i a$」というレベル $i+1$ の命題となる．

命題と証明の階層

$\Lambda_=$ 上のメタな関係 $\mathit{Prop}_i(a)$ と $\mathit{Proves}_i(p, a)$ はパラメータ i により階層化される：

1. $i < j$ のとき，$\mathit{Prop}_i(a) \Longrightarrow \mathit{Prop}_j(a)$.
2. $i < j$ のとき，$\mathit{Proves}_i(p, a) \Longrightarrow \mathit{Proves}_j(p, a)$.

上のことは，RPT で以下のように internalize される．

1. $\vdash_\omega \forall i, j.\ i, j \in \mathbf{N} \to i{<}j \to {\models_i}\ a \to {\models_j}\ a$.
2. $\vdash_\omega \forall i, j, p.\ i, j \in \mathbf{N} \to i{<}j \to p \vdash_i a \to p \vdash_j a$.
3. $\vdash_\omega \forall i, j.\ i, j \in \mathbf{N} \to i{<}j \to {\vdash_i}\ a \to {\vdash_j}\ a$.

Remark. 項目 3 の $\vdash_i a$ は $\exists p.\ p \vdash_i a$ の略記であり，様相演算子とみなせる．$\vdash_j, \vdash_\omega$ も同様．

RPT における集合

RPT では集合も階層化して定義する．a がレベル i 集合であることと，関連する記法を以下のように定める．

- $\boldsymbol{\sigma_i(a)} := \forall x.\ \models_i a(x).$
- $\boldsymbol{\{x \mid A\}} := \boldsymbol{\lambda x.\ A}.$
- $\boldsymbol{b \in_i a} := \sigma_i(a) \wedge a(b).$

このとき，以下が RPT で成立する．

- $\vdash_{i+1} \forall a.\ \sigma_i(a) \to \sigma_{i+1}(a).$
- $\vdash_{i+2} \forall a, x.\ \sigma_i(a) \to x \in_i a \leftrightarrow x \in_{i+1} a.$
- (Predication) $\vdash_{i+1} \forall a, b.\ \sigma_i(a) \to \models_i b \in_i a.$
- (Comprehension) $\vdash_{i+1} \forall a, x.\ \sigma_i(a) \to x \in_i a \leftrightarrow a(x).$

RPT における集合 (続)

$V := \{x \mid x = x\}$ とするとき，$\vdash_1 \sigma_0(V)$ および $\vdash_1 V \in_0 V$ が成立する．

Russel set R_i を $R_i := \{x \mid x \notin_i x\}$ とするとき，

$$R_i \in_i R_i \iff \sigma_i(R_i) \text{ かつ } R_i(R_i)$$

である．したがって，もし $\sigma_i(R_i)$ であるとすれば，

$$R_i \in_i R_i \iff R_i(R_i) \iff R_i \notin_i R_i$$

となり矛盾が得られる．故に $\neg\sigma_i(R_i)$ である．よって，上の同値性から，$R_i \notin_i R_i$ となる．この議論を形式化すると

$$\vdash_{i+1} R_i \notin_i R_i$$

が得られる．

指示透明性

命題 P をその中の項 a の出現に注目して $P(a)$ と書く．このとき，**代入原理**

$$\frac{P(a) \quad a = b}{P(b)}$$

が成り立つならば，文脈 $P(\ )$ は**指示透明性を持つ**という [Quine 1961].

文脈 $L(\ ) :=$「$(\ )$ に生命が存在する」，$a :=$「宵の明星」，$b :=$「明けの明星」とする．このとき，以下の左の推論は成立するが，右は成立しない．

$$\frac{L(a) \to L(a) \quad a = b}{L(a) \to L(b)} \qquad \frac{\Box(L(a) \to L(a)) \quad a = b}{\Box(L(a) \to L(b))}$$

文脈 $L(a) \to L(\ )$ は指示透明性を持つが，**様相的文脈** $\Box(L(a) \to L(\ ))$ は指示透明性を持たない (referentially opaque).

プログラミング言語の指示透明性

指示透明性は分析哲学由来の概念であるが，計算機科学においてもプログラムの性質の検証やプログラムの等価変換の研究に有用な概念である．

一般に，変数への値の代入 (assignment) **を許す言語は指示透明性を持たないとされている．文脈**

$$x := x + 1; (\ )$$

に関して，以下に見るように代入原理が成立しないからである．

$$\frac{(x := x + 1; x) = (x := x + 1; x) \quad x = 0}{(x := x + 1; x) = (x := x + 1; 0)}$$

代入を許しかつ指示透明なプログラミング言語

[S 1994] はRPTのλ-計算を拡張し，代入命令を持ちながら指示透明な関数型言語を与えた．

アイディアは簡単で，代入命令は常に代入される変数についてのlet文の有効範囲の中に出現するように言語の文法を設計した．

$$\frac{\mathsf{let}\ x = 0(x := x + 1; x) = \mathsf{let}\ x = 0(x := x + 1; x) \quad x = 0}{\mathsf{let}\ x = 0(x := x + 1; x) = \mathsf{let}\ x = 0(x := x + 1; x)}$$

上のlet文中の x は**束縛変数**であり，上の推論の2つ目の前提 $x = 0$ の**自由変数** x とは無関係であるため，代入原理が成立している．

環境を持つ単純型付文脈計算

これまで見てきたように，「文脈」$C(\ )$ は参照透明性の概念において中心的な位置を占めている．しかし，文脈自身は，考察の対象であるプログラミング言語を考察するためのメタレベルの存在者である．

また let 文の意味を考察するためには，局所的な変数とそれが一時的に保持する値の組の有限集合としての「環境」の概念が重要である．

[S-Sakurai-Burstall 2001]，[S-Sakurai-Kameyama 2002] では環境と文脈を言語の一級の構成要素としてもつ型付の λ-計算を提案し，それが参照透明性を持つことを示した．

$\lambda\kappa\epsilon$：環境を持つ単純型付文脈計算

[Sato-Sakurai-Kameyama 2002] の体系 $\lambda\kappa\epsilon$ の概要．

型の定義

$$A, B, C ::= K \mid E \mid A \to B \mid A^E$$
$$K ::= \text{atomic types}$$
$$E ::= \{x^A, \ldots, z^C\}$$

$E = \{x^A, \ldots, z^C\}$ は環境の型で，変数 x^A 等は名前の一部として型を持つ．A^E は文脈の型．

Remark. A^E を $[E]A$ と書くことにより，Curry-Howard の同型により，文脈の型を様相命題とみなすことができる．

$\lambda\kappa\epsilon$: 環境を持つ単純型付文脈計算

型付け規則

$$\frac{}{x^A : A}\ \text{axiom}$$

$$\frac{b : B}{\lambda x^A.\, b : A \to B} \to \mathsf{I} \qquad \frac{b : A \to B \quad a : A}{b(a) : B} \to \mathsf{E}$$

$$\frac{a : A}{\kappa\{x, \ldots, z\}.\, a : A^{\{x,\ldots,z\}}}\ \textbf{文脈}\,\mathsf{I} \qquad \frac{a : A^E \quad e : E}{a \cdot e : A}\ \textbf{文脈}\,\mathsf{E}$$

$$\frac{a : A \quad \cdots \quad c : C}{\{a/x^A, \ldots, c/z^C\} : \{x^A, \ldots, z^C\}}\ \textbf{環境}\,\mathsf{I} \qquad \frac{e : E \quad a : A^E}{e[\![a]\!] : A}\ \textbf{環境}\,\mathsf{E}$$

多段階計算

多段階計算 (staged computaion) **は部分評価** (partial evaluation) **の発展形とみなすことができる．**
プログラムをデータとして扱い，コードの変形を段階に分けて計算する．
異なる段階の間の関係を Kripke **モデルに対応させることにより，型付の様相論理や時相論理を設計することができる．**
異る段階での変数を考えるため，メタ変数を形式化する．

参考文献

[McCarthy-S-Hayashi-Igarashi 1978] John, McCarthy, Masahiko Sato, Takeshi Hayashi and Shigeru Igarashi, On the Model Theory of Knowledge, *Stanford Artificial Intelligence Laboratory*, AIM-312, 1978.
`http://i.stanford.edu/pub/cstr/reports/cs/tr/78/657/CS-TR-78-657.pdf`
[Quine 1961] Quine, W.V., Reference and Modality, In W.V. Quine, *From a Logical Point of View*, second edition, 130-138. Harvard University Press.
[S 1977] A study of Kripke-type models for some modal logics by Gentzen's sequential method, *Publ. RIMS, Kyoto U.*, **13**, 381-468.
`https://www.jstage.jst.go.jp/article/kyotoms1969/13/2/13_2_381/_pdf`

参考文献

[S 1991] Masahiko Sato, Adding Proof Objects and Inductive Definition Mechanisms to Frege Structures, 1991, in T. Ito, A.R. Meyer eds., *Theoretical Aspects of Computer Science, International Conference TACS'91 Proceedings*, Lecture Notes in Computer Sciecne 526, 53-87.

[S 1994] Masahiko Sato, A Purely Functional Lnaguage with Encapsulated Assignment, 1994, in M. Hagiya, J.C. Mitchell eds., *Theoretical Aspects of Computer Software, International Symposium TACS'94 Proceedings*, Lecture Noter in Computer Science 789, 179-202.

[S-Sakurai-Burstall 2001] Masahiko Sato, Takafumi Sakuai, Rod Burstall, Explicit Environments Fundamenta Informaticae vol. 45, 79–115, 2001.

[S-Sakurai-Kameyama 2002] Masahiko Sato, Takafumi Sakurai, Yukiyoshi Kameyama, *A Simply Typed Context Calculus with First-Class Environments*, Journal of Functional and Logic Programming, Vol. 2002, No. 4, March 2002.