様相論理に基づくプロトコルの仕様記述

中島俊介　　長谷川晴朗
沖通信システム (株)　　沖電気工業 (株)

1　はじめに

並列プロセス間のプロトコルを記述するために，プロセス代数や時相論理などの形式的な枠組が提案されている．これらの枠組は，検証の基礎になる重要なものであるが，実際の複雑なプロトコルに対して，十分な記述性をもっているとは言えない．

本稿では，主に記述性の観点から，実験的に検討・試作しているプロトコル仕様記述言語 Ack について紹介する．なお，説明や記述の例としては，電話交換機のプロトコルを用いる．

2　基本的な考え方

Ack では，複雑なプロトコルを記述するための枠組を様相論理 (制限された一階述語論理) 上に構築している．

様相論理では，通常の論理オペレータである ¬ (否定)，∨ (選言)，∧ (連言)，→ (含意) に加えて，2 つの様相オペレータ □ (必然)，◇ (可能) を扱う．

様相論理のセマンティクスとして，可能世界モデルに基づく解釈を考えたとき，Ack では，ひとつの可能世界があるプロセスの状態遷移の一部分に対応し，各命題 (原子論理式) はその状態遷移に関する言及であるとする．

例えば，次の式について考える．

$$\Box(hook\_a{:}\,a\_offhook \rightarrow \Diamond phone\_a{:}\,a\_onhook)$$

この式の意図する意味は，「プロセス $hook\_a$ にアクション a_onhook を起こす状態遷移が存在するなら，プロセス $phone\_a$ にも，a_onhook を起こす状態遷移が必要である」ということである．この式に対して，図 1 の実線のアークが示す状態遷移が与えられると，前件を満たす 3 つの場合に対して，後件に対応する点線のアークが示す状態遷移の存在が導かれる．

このように，Ack では，すべての状態遷移を具体的に記述する必要がなく，ひとつの式からいろいろな場合の状態遷移が演繹的に導かれる．このことが記述性の点でメリットになる．

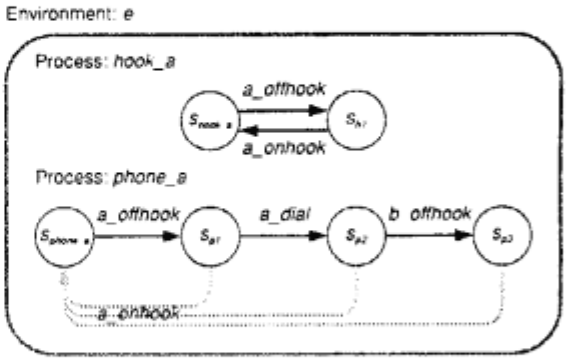

図 1: 状態遷移ダイアグラム

3　並列と同期

Ack で仕様記述される並列プロセスは，以下に述べる同期メカニズムを前提としている．

各プロセスは，プロセス毎に定められた状態遷移ダイアグラムに従い，並列に動作する．このとき，プロセスの各状態は必ずある環境に属すものと考え，環境をそれに属するプロセスの集合と同一視する．そして，遷移の前状態の環境に属する他プロセスが，その遷移と同一のアクションを行う遷移をもつならば，それらの遷移は同時に行わなければならない．

図 1 はその一例であり，2 つのプロセス $hook\_a$ と $phone\_a$ の並列動作を表しており，すべての状態は環境 e に属している．従って，アクション $a\_offhook$ と a_onhook を行う遷移は，両方のプロセスで同期する．

4　仕様記述言語

Ack によるプロトコルの仕様記述は，次に示す式の集合として与えられる．

$$\boldsymbol{A}_1[\boldsymbol{x}] \wedge \cdots \wedge \boldsymbol{A}_m[\boldsymbol{x}] \rightarrow \boldsymbol{B}_1[\boldsymbol{x} \cup \boldsymbol{y}] \vee \cdots \vee \boldsymbol{B}_n[\boldsymbol{x} \cup \boldsymbol{y}]$$

ここで，$\boldsymbol{x}$，$\boldsymbol{y}$ は式の中に現れる (自由) 変数の集合で，$\boldsymbol{x} \cap \boldsymbol{y} = \emptyset$ とする．また，各 $\boldsymbol{A}_i$，$\boldsymbol{B}_j$ は原子論理式の連言である．

この式は，次の一階述語論理式と意味的に等価であると定める．

$$\forall \boldsymbol{x} \exists \boldsymbol{y} (\boldsymbol{A}_1[\boldsymbol{x}] \wedge \cdots \wedge \boldsymbol{A}_m[\boldsymbol{x}] \rightarrow \boldsymbol{B}_1[\boldsymbol{x} \cup \boldsymbol{y}] \vee \cdots \vee \boldsymbol{B}_n[\boldsymbol{x} \cup \boldsymbol{y}])$$

Protocol Specification Description based on Modal Logic
Shunsuke NAKAJIMA*, Haruo HASEGAWA**
*Oki Telecommunication Systems Co., Ltd.
**Oki Electric Industry Co., Ltd.

実際の仕様記述では，各 A_i, B_j はそれぞれ状態遷移を表し，この式の直観的な意味は，「状態遷移 A_i がすべて存在するならば，状態遷移 B_j のいずれかが必要である」となる．

仕様記述に用いる主な関数，原子論理式，公理を以下に定義する．なお，変数は大文字，定数は小文字のシンボルで表す．

関数

- $act(P)$ — プロセス P が起こすアクションの集合
- $add(T)$ — 遷移 T による環境の増分
- $del(T)$ — 遷移 T による環境の減分
- $env(S)$ — 状態 S が属する環境

原子論理式

- $=, \in$ などの関係演算
- $P\colon S \triangleright T \triangleright S'$ — 「プロセス P が状態 S から遷移 T で状態 S' になる」
- $T!A$ — 「遷移 T でアクション A が起こる」
- $S_a|S_b$ — 「あるプロセスが状態 S_a のとき，他のプロセスが状態 S_b になることがある」

公理

- 遷移による環境変化の関係

$$
\begin{aligned}
&P_a\colon S_a \triangleright T_a \triangleright S'_a \wedge P_b\colon S_b \triangleright T_b \triangleright S'_b \\
&\quad \rightarrow P_b \notin env(S_a) \wedge P_b \notin add(T_a) \\
&\qquad \wedge P_b \notin del(T_a) \wedge P_b \notin env(S'_a) \\
&\quad \vee P_b \notin env(S_a) \wedge P_b \in add(T_a) \\
&\qquad \wedge P_b \notin del(T_a) \wedge P_b \in env(S'_a) \\
&\quad \vee P_b \in env(S_a) \wedge P_b \notin add(T_a) \\
&\qquad \wedge P_b \in del(T_a) \wedge P_b \notin env(S'_a) \\
&\quad \vee P_b \in env(S_a) \wedge P_b \notin add(T_a) \\
&\qquad \wedge P_b \notin del(T_a) \wedge P_b \in env(S'_a)
\end{aligned}
$$

- 並列プロセスの同期的遷移に関する条件

$$
\begin{aligned}
&P_a\colon S_a \triangleright T_a \triangleright S'_a \wedge P_b\colon S_b \triangleright T_b \triangleright S'_b \\
&\qquad \wedge S_a|S_b \wedge P_b \in env(S'_a) \wedge T_a!A \wedge T_b!A \\
&\qquad \rightarrow S'_a|S'_b \\
&P_a\colon S_a \triangleright T_a \triangleright S'_a \wedge P_b\colon S_b \triangleright T_b \triangleright S'_b \\
&\qquad \wedge S_a|S_b \wedge P_b \in env(S'_a) \wedge T_a!A_a \\
&\qquad \rightarrow A_a \in act(P_b) \vee S'_a|S_b \\
&P_a\colon S_a \triangleright T_a \triangleright S'_a \wedge P_b\colon S_b \triangleright T_b \triangleright S'_b \\
&\qquad \wedge S_a|S_b \wedge P_b \in env(S_a) \wedge T_b!A_b \\
&\qquad \rightarrow A_b \in act(P_a) \vee S_a|S'_b
\end{aligned}
$$

5 仕様記述例

実際の仕様記述の例を以下に示す．また，その記述が表す状態遷移を図 1 に示す．

$$
\begin{aligned}
&\rightarrow act(hook\_a) = \{a\_offhook, a\_onhook\} \\
&\quad \wedge env(s_{hook\_a}) = \emptyset \\
&\quad \wedge hook\_a\colon s_{hook\_a} \triangleright T_0 \triangleright S_1 \wedge T_0!a\_offhook \\
&\quad \wedge add(T_0) = \emptyset \wedge del(T_0) = \emptyset \\
&\quad \wedge hook\_a\colon S_1 \triangleright T_1 \triangleright s_{hook\_a} \wedge T_1!a\_onhook \\
&\quad \wedge add(T_1) = \emptyset \wedge del(T_1) = \emptyset \qquad (1) \\
&\rightarrow act(phone\_a) = \{a\_offhook, \ldots, b\_dial\} \\
&\quad \wedge env(s_{phone\_a}) = \{hook\_a\} \wedge s_{phone\_a}|s_{hook\_a} \\
&\quad \wedge phone\_a\colon s_{phone\_a} \triangleright T_0 \triangleright S_1 \wedge T_0!a\_offhook \\
&\quad \wedge add(T_0) = \emptyset \wedge del(T_0) = \emptyset \\
&\quad \wedge hook\_a\colon S_1 \triangleright T_1 \triangleright S_2 \wedge T_1!a\_dial \\
&\quad \wedge add(T_1) = \{phone\_b\} \wedge del(T_1) = \emptyset \\
&\quad \wedge phone\_a\colon S_2 \triangleright T_2 \triangleright S_3 \wedge T_2!b\_offhook \\
&\quad \wedge add(T_2) = \emptyset \wedge del(T_2) = \emptyset \qquad (2) \\
&hook\_a\colon S \triangleright T \triangleright S' \wedge T!a\_onhook \\
&\quad \wedge phone\_a\colon S_0 \triangleright T_0 \triangleright S_1 \wedge S_1|S \\
&\quad \rightarrow phone\_a\colon S_1 \triangleright T_1 \triangleright s_{phone\_a} \wedge T_1!a\_onhook \qquad (3)
\end{aligned}
$$

(1) 式はプロセス $hook\_a$ の状態遷移を表し，次の式はプロセス $phone\_a$ の実線のアークが示す状態遷移を表す．このとき，先に述べた両方の公理から，状態 s_{p1}, s_{p2}, s_{p3} について，以下のことが分かる．

$$hook\_a \in env(s_{pi}) \wedge s_{pi}|s_{h1} \quad (i = 1, 2, 3)$$

従って，その各状態で (3) 式の前件が満たされ，最終的に，点線のアークが示す状態遷移が導かれる．

6 おわりに

本稿では，複雑なプロトコルを記述するために有用な仕様記述言語 Ack について紹介した．

本研究は第五世代コンピュータ・プロジェクトの一環として行なわれているものであり，同プロジェクトで開発された自動定理証明システム MGTP[1] を用いて，Ack 仕様から状態遷移ダイアグラムを自動作成する処理系の実現についても検討を行っている．

謝辞 日頃御指導頂く ICOT 研究部長代理長谷川隆三氏を始め，自動定理証明グループの方々に感謝します．

参考文献

[1] Fujita, H., Hasegawa, R.: "A Model Generation Theorem Prover Using A Ramified-Stack Algorithm", ICOT TR-606, 1990.