知識工学と人工知能 63 － 4
（１９８９．３．１３）

要求仕様における動作制約の部分性

中島俊介, 橋本正明, 門田充弘
ATR 通信システム研究所

リアルタイム・システムの仕様には, 状況 - 動作規則と動作 - 状況変化規則の記述が必要である. また, 要求仕様の記述には, モジュール性と実行可能性が重要である.

本稿では, 状況 - 動作規則の記述におけるモジュール性について論じる. 動作制約の記述に関して, Total な言語 $\mathcal{L}_t$ と Partial な言語 $\mathcal{L}_p$ を定義する. 記述のモジュール性に関して, $\mathcal{L}_p$ が $\mathcal{L}_t$ よりも優れていることを示す. また, 「例外」の記述に関して, 非単調推論と $\mathcal{L}_p$ の違いを指摘する.

Partiality of Event Constraint in Requirement Specifications

Shunsuke NAKAJIMA, Masaaki HASHIMOTO, Michihiro MONDEN
ATR Communication Systems Research Laboratories
Communications Software Department
Twin 21 Bldg. MID Tower, 2-1-61 Shiromi Higashi-ku, Osaka, 540 Japan

A specification of real-time system must be described with 'situation-event' rules and 'event-situation change' rules. Modularity and Executability are needed to describe a requirement.

In this paper, the modularity of 'situation-event' rules description is discussed. Language $\mathcal{L}_t$ and $\mathcal{L}_p$ are defined. $\mathcal{L}_t$ is a total language on the description of event constraint, and $\mathcal{L}_p$ is a partial one. It is shown that the modularity of description in $\mathcal{L}_p$ is better than that in $\mathcal{L}_t$. And differences between 'exception' description of some nonmonotonic reasoning systems and of $\mathcal{L}_p$ is pointed out.

1 はじめに

一般に, リアルタイム・システムは, 刻々と変化する状況に応じて, 異った動作を実行しなければならない. そのため, リアルタイム・システムの仕様には, 状況 - 動作規則と動作 - 状況変化規則が記述される必要がある. 状況 - 動作規則は, 各々の状況に対して, その状況で実行すべき動作を対応させた規則である. この規則を用いることによって, 現時点の状況でどんな動作を実行するかが決定される. 動作 - 状況変化規則は, 各々の動作に対して, その動作の実行による状況の変化を対応させた規則である. この規則を用いることによって, 次の時点の状況がどうなるかが決定される.

また, システムの要求分析では, 利用者から断片的に与えられる要求を統合して行かなければならない. そのため, 要求仕様記述は, モジュール性と実行可能性をもつ必要がある. モジュール性は, システムの一部分に閉じて仕様記述が行なえるという性質である. この性質によって, 要求分析の過程においても要求を記述することができる. 実行可能性は, プロトタイプ・システムとして, 仕様記述を直接実行できるという性質である. この性質によって, 記述された仕様の正しさを利用者が確認することができる.

本稿では, 状況 - 動作規則を記述する言語における記述のモジュール性について論じる. ある状況 - 動作規則を解釈する際に, その規則の状況は動作に対する制約として働く. そして, その動作制約の記述に関して, Total な言語と Partial な言語が考えられる. Total な言語は, 制約の記述が完全 (Total) であることを前提とする言語である. Total な言語の例として, 2.2 節に言語 $\mathcal{L}_t$ を定義し, その記述と解釈の例を 3.2, 3.3 節に示す. Partial な言語は, 制約の記述が不完全 (Partial) であることを前提とする言語である. Partial な言語の例として, 2.3 節に言語 $\mathcal{L}_p$ を定義し, その記述と解釈の例を 3.4, 3.5 節に示す. さらに, 記述のモジュール性に関して, $\mathcal{L}_p$ の記述が $\mathcal{L}_t$ の記述よりも優れていることを 4.1 節で考察する. また, そのようなモジュール性は, $\mathcal{L}_p$ がもつ解釈の非単調性と「例外」の記述に関係がある. そこで, 「例外」の記述に関して, 幾つかの非単調推論との比較を 4.2 節で考察する.

2 言語の定義

本節では, Total な言語 $\mathcal{L}_t$ と Partial な言語 $\mathcal{L}_p$ を定義する.

2.1 諸定義

定義: 項 (Term)

$\mathcal{V}$ を変数記号の集合とする. $n \in \mathbf{N}$ ($\mathbf{N}$ は 0 を含む自然数) に対して, $\mathcal{F}_n$ を n 引数の関数記号の有限集合とする. このとき, 項の集合 T を次のように定義する.

i) $v \in \mathcal{V}$ ならば $v \in T$.
ii) $f_0 \in \mathcal{F}_0$ ならば $f_0 \in T$.
iii) $f_n \in \mathcal{F}_n$ かつ $t_1, \cdots, t_n \in T$ ならば, $f_n(t_1, \cdots, t_n) \in T$.
iv) 上の i - iii で定義されるものだけが項である.

また, 変数記号の現れない項を基底項 (ground term) と言う.

定義: 状態 (State)

T を項の集合とする. $n \in \mathbf{N}$ に対して, $\mathcal{S}_n$ を n 引数の状態記号の有限集合とする. このとき, 状態の集合 Σ_s を次のように定義する.

i) $s_0 \in \mathcal{S}_0$ ならば $s_0 \in \Sigma_s$.
ii) $s_n \in \mathcal{S}_n$ かつ $t_1, \cdots, t_n \in T$ ならば, $s_n(t_1, \cdots, t_n) \in \Sigma_s$.
iii) 上の i, ii で定義されるものだけが状態である.

また, 変数記号の現れない状態を基底状態 (ground state) と言う.

定義: 動作 (Event)

T を項の集合とする. $n \in \mathbf{N}$ に対して, $\mathcal{E}_n$ を n 引数の動作記号の有限集合とする. このとき, 動作の集合 Σ_e を次のように定義する.

i) $e_0 \in \mathcal{E}_0$ ならば $e_0 \in \Sigma_e$.
ii) $e_n \in \mathcal{E}_n$ かつ $t_1, \cdots, t_n \in T$ ならば, $e_n(t_1, \cdots, t_n) \in \Sigma_e$.
iii) 上の i, ii で定義されるものだけが動作である.

また, 変数記号の現れない動作を基底動作 (ground event) と言う.

定義: 状況 (Situation)

Σ_s を状態の集合, Σ_e を動作の集合とし, $\Sigma = \Sigma_s \cup \Sigma_e$ とする. 状況の集合は Σ のべき集合 2^Σ である. また, 変数記号の現れない状況を基底状況 (ground situation) と言う.

定義: 代入 (Substitution)

代入は代入成分 (substitution component) の集合である. 代入成分の形式を次のように定義する.

$$v = t.$$

ここで, v は変数記号であり, t は項である.

表現 (項, 状態, 動作, 状況) E に対して, 代入 θ を適用した結果を $E\theta$ と書く. $E\theta$ は 全ての代入成分 $v = t \in \theta$ に対して, E の中の v の現れを t で置き換えて得られる表現である.

2.2 Total な言語 $\mathcal{L}_t$

$\mathcal{L}_t$ は Total な制約記述を前提とする言語 (Total な言語) である. これは次のことを意味する.

- 必ずシステム全体が記述される.
  (システムの一部分に閉じて記述できない.)
- 制約を満足する動作が衝突することがない.
- 制約を満足する動作を全て実行する.

2.2.1 Syntax

テーマ記述 $\mathcal{D}_t$ は, 言語 $\mathcal{L}_t$ の節の有限集合として与えられる. $\mathcal{L}_t$ の節を Cause-Effect 節と言う.

Cause-Effect 節の形式を次のように定義する.

$$H_1, \cdots, H_m \Leftarrow B_1, \cdots, B_n.$$

ここで, $H_1, \cdots, H_m$ は動作であり, $m \geq 1$ である. $B_1, \cdots, B_n$ は状態ないし動作であり, $n \geq 0$ である. また, 状況 $\{H_1, \cdots, H_m\}$ をその節の Effect 状況, 状況 $\{B_1, \cdots, B_n\}$ をその節の Cause 状況と言う.

2.2.2 Semantics

節 $c \in \mathcal{D}_t$ に対して, 関数 $S_C(c), S_E(c)$ を次のように定義する.

$S_C(c)$ は節 c の Cause 状況を表す関数である.
$S_E(c)$ は節 c の Effect 状況を表す関数である.

定義: 拡張 (Expansion)

拡張 E は基底状態ないし基底動作 s の集合である.

基底状態ないし基底動作 s および拡張 E に対して, 述語 $Hold\_in(s, E)$ を次のように定義する.

i) $s \in E$ ならば $Hold\_in(s, E) = True$.
ii) そうでなければ $Hold\_in(s, E) = False$.

基底状況 S および拡張 E に対して, 述語 $Part\_of(S, E)$ を次のように定義する.

$$Part\_of(S, E) = \bigwedge_{s_i \in S} Hold\_in(s_i, E).$$

基底状況 S に対して, その拡張 E を次のように定義する.

i) $s \in S$ ならば $s \in E$.
ii) $c \in \mathcal{D}_t$ かつ,
  $Part\_of(S_C(c)\theta, E) = True$ かつ,
  $s' \in S_E(c)\theta$ ならば,
  $s' \in E$.
  ここで, θ は $S_C(c), S_E(c)$ に現れる全ての変数記号を基底項で置き換える代入である.
iii) 上の i, ii で定義されるものだけが拡張の要素である.

定義: 解釈

基底状況 S に対する記述 $\mathcal{D}_t$ の解釈 $\mathcal{I}_t(\mathcal{D}_t, S)$ は, S において実行する基底動作の集合である.

拡張 E に対して, $\mathcal{I}_t(\mathcal{D}_t, S)$ を次のように定義する.

$c \in \mathcal{D}_t$ かつ,
$Part\_of(S_C(c)\theta, E) = True$ かつ,
$s' \in S_E(c)\theta$ ならば,
その時に限り $s' \in \mathcal{I}_t(\mathcal{D}_t, S)$.
ここで, θ は $S_C(c), S_E(c)$ に現れる全ての変数記号を基底項で置き換える代入である.

2.3 Partial な言語 $\mathcal{L}_p$

$\mathcal{L}_p$ は Partial な制約記述を前提とする言語 (Partial な言語) である. これは次のことを意味する.

- システムの一部分に閉じて記述できる.
- 制約を満足する動作が衝突することがある.
- 制約を満足する中で適当な動作を実行する.

ここで, 適当な動作を選択する基準が問題となる. $\mathcal{L}_p$ では次のような基準を設定している.

- 衝突する動作の中で最も厳しい制約をもつ動作を採用する.
  (例外的な動作ほど厳しい制約が課せられる)
- 衝突しない動作は全て採用する.

2.3.1 Syntax

テーマ記述 $\mathcal{D}_p$ は, 言語 $\mathcal{L}_p$ の節の有限集合として与えられる. $\mathcal{L}_p$ の節には Cause-Effect 節と Impossible 節がある.

Cause-Effect 節の形式を次のように定義する.

$$H_1, \cdots, H_l \Leftarrow B_1, \cdots, B_m \mid B_{m+1}, \cdots, B_{m+n}.$$

ここで, $H_1, \cdots, H_l$ は動作であり, $l \geq 1$ である. $B_1, \cdots, B_{m+n}$ は状態ないし動作であり, $m, n \geq 0$ である. また, 状況 $\{B_1, \cdots, B_m\}$ をその節の Cause 状況, 状況 $\{B_{m+1}, \cdots, B_{m+n}\}$ をその節の Where 状況, 状況 $\{H_1, \cdots, H_l\}$ をその節の Effect 状況と言う.

Impossible 節の形式を次のように定義する.

$$Impossible\ H_1, \cdots, H_n.$$

ここで, $H_1, \cdots, H_n$ は動作であり, $n \geq 1$ である. また, 状況 $\{H_1, \cdots, H_n\}$ をその節の Impossible 状況と呼ぶ.

2.3.2 Semantics

$\mathcal{D}_p$ に対して, 節集合 C_{CE}, C_I を次のように定義する.

C_{CE} は $\mathcal{D}_p$ に表れる Cause-Effect 節の集合である.
C_I は $\mathcal{D}_p$ に表れる Impossible 節の集合である.

節 $c \in C_{CE}$ に対して, 関数 $S_C(c), S_W(c), S_E(c)$ を次のように定義する.

$S_C(c)$ は節 c の Cause 状況を表す関数である.
$S_W(c)$ は節 c の Where 状況を表す関数である.
$S_E(c)$ は節 c の Effect 状況を表す関数である.

節 $c \in C_I$ に対して, 関数 $S_I(c)$ を次のように定義する.

$S_I(c)$ は節 c の Impossible 状況を表す関数である.

定義: 拡張 (Expansion)

拡張 E は基底状態ないし基底動作 s と基底状況 S の対 $\langle s, S\rangle$ の集合である.

基底状態ないし基底動作 s と拡張 E に対して, 述語 $Hold\_in(s, E)$ を次のように定義する.

i) $\langle s, S\rangle \in E$ ならば $Hold\_in(s, E) = True$.
ii) そうでなければ $Hold\_in(s, E) = False$.

基底状態ないし基底動作 s と拡張 E に対して, 関数 $Cause(s, E)$ を次のように定義する.

$\langle s, S\rangle \in E$ ならば,
その時に限り $Cause(s, E) = S$.

基底状況 S と拡張 E に対して, 述語 $Part\_of$ を次のように定義する.

$$Part\_of(S, E) = \bigwedge_{s_i \in S} Hold\_in(s_i, E).$$

基底状況 S と拡張 E に対して, 関数 $Cause'(S, E)$ を次のように定義する.

$$Cause'(S, E) = \bigcup_{s_i \in S} Cause(s_i, E).$$

基底状況 S に対して, その拡張 E を次のように定義する.

i) $s \in S$ ならば $\langle s, \{s\}\rangle \in E$.
ii) $c \in C_{CE}$ かつ,
$Part\_of((S_C(c) \cup S_W(c))\theta, E) = True$ かつ,
$s' \in S_E(c)\theta$ ならば,
$\langle s', Cause'(S_C(c)\theta, E)\rangle \in E$.
ここで θ は $S_C(c), S_W(c), S_E(c)$ に現れる全ての変数記号を基底項で置き換える代入である.
iii) 上の i, ii で定義されるものだけが拡張の要素である.

定義: 提案 (Proposal)

提案は基底状況 S_1, S_2 の対 $\langle S_1, S_2\rangle$ である.

拡張 E に対して, 提案の集合 P を次のように定義する.

$c \in C_{CE}$ かつ,
$Part\_of((S_C(c) \cup S_W(c))\theta, E) = True$ ならば,
その時に限り $\langle S_E(c)\theta, Cause'(S_C(c)\theta, E)\rangle \in P$.
ここで θ は $S_C(c), S_W(c), S_E(c)$ に現れる全ての変数記号を基底項で置き換える代入である.

定義: 衝突 (Conflict)

基底状況 S_1, S_2 に対して, 述語 $Conflict(S_1, S_2)$ を次のように定義する.

i) $c \in C_I$ かつ,
$S_I(c)\theta \subseteq S_1 \cup S_2$ ならば,
$Conflict(S_1, S_2) = True$.
ここで θ は $S_I(c)$ に現れる全ての変数記号を基底項で置き換える代入である.
ii) そうでなければ $Conflict(S_1, S_2) = False$.

定義: 棄却 (Rejection)

棄却は基底状況 S_1, S_2 の対 $\langle S_1, S_2 \rangle$ である.

提案の集合 P に対して, 棄却の集合 R を次のように定義する.

$\langle S_{11}, S_{12} \rangle, \langle S_{21}, S_{22} \rangle \in P$ かつ,
$\langle S_{11}, S_{12} \rangle \neq \langle S_{21}, S_{22} \rangle$ かつ,
$Conflict(S_{11}, S_{21}) = True$ かつ,
$S_{12} \subset S_{22}$ ならば,
その時に限り $\langle S_{11}, S_{12} \rangle \in R$.

定義: 解釈

基底状況 S に対する記述 $\mathcal{D}_p$ の解釈 $\mathcal{I}_p(\mathcal{D}_p, S)$ は, S において実行する動作の集合である.

提案の集合 P と棄却の集合 R に対して, $\mathcal{I}_p(\mathcal{D}_p, S)$ を次のように定義する.

$\mathcal{I}_p(\mathcal{D}_p, S) = \bigcup_{\langle S_1, S_2 \rangle \in P-R} S_1$.

3 記述と解釈の例

本節では, 2つの言語 $\mathcal{L}_t, \mathcal{L}_p$ による交換機の動作仕様の記述と解釈を例示する.

記述例では, 次に示す仕様に限って, $\mathcal{L}_t$ の記述 $\mathcal{D}_t$ と $\mathcal{L}_p$ の記述 $\mathcal{D}_p$ を示す.

- **項目 1:** 電話機 A の受話器を上げたら, 電話機 A に発信音がつながる.
- **項目 2:** その後, 電話機 A より電話機 B の電話番号をダイヤルしたら, 電話機 A に呼出音がつながり, 電話機 B にベル音がつながる.
- **項目 3:** ただし, 電話機 B が話中時にはそうならず, 電話機 A に話中音がつながる.
- **項目 4:** また, 電話機 A の受話器を上げたとき, 電話機 A が電話機 B へのホットラインなら, すぐに電話機 A に呼出音がつながり, 電話機 B にベル音がつながる.

解釈例では, 次の状況に限って, $\mathcal{L}_t$ の解釈 $\mathcal{I}_t$ と $\mathcal{L}_p$ の解釈 $\mathcal{I}_p$ を示す.

- **状況 1:** (普通の) 電話機の受話器を上げた場合.
- **状況 2:** ホットラインの電話機の受話器を上げた場合.
- **状況 3:** ホットラインの電話機の受話器を上げた時, 相手の電話機が話中の場合.

3.1 各種記号

記述 $\mathcal{D}_t, \mathcal{D}_p$ では, 以下に示すような記号を使用する.

変数記号, 関数記号

- $x_0, x_1, x_2, \cdots$
  変数記号.
- $P0, P1, P2, \cdots$
  電話機を表す関数記号.
- $T0, T1, T2, \cdots$
  音を表す関数記号.
- $0, 1, 2, \cdots$
  番号を表す関数記号.

状態記号

- $Normal(x_0)$
  電話機 x_0 は普通の電話機である.
- $Hotline(x_0, x_1)$
  電話機 x_0 は電話機 x_1 へのホットラインの電話機である.
- $DialNumber(x_0, x_1)$
  電話機 x_0 の電話番号は x_1 番である.
- $Ready(x_0)$
  電話機 x_0 は話中でない.
- $Busy(x_0)$
  電話機 x_0 は話中である.
- $Connected(x_0, x_1)$
  電話機 x_0 に音 x_1 がつながっている.
- $IdleTone(x_0)$
  音 x_0 は無音である.
- $DialTone(x_0)$
  音 x_0 は発信音である.
- $CallTone(x_0)$
  音 x_0 は呼出音である.
- $RingTone(x_0)$
  音 x_0 はベル音である.
- $BusyTone(x_0)$
  音 x_0 は話中音である.

ただし, 状態 $Nomal(x_0)$ は言語 $\mathcal{L}_p$ の記述に使用されない.

動作記号

- $Offhook(x_0)$
  電話機 x_0 の受話器 を上げる.
- $Dial(x_0, x_1)$
  電話機 x_0 より番号 x_1 をダイヤルする.
- $Connect(x_0, x_1)$
  電話機 x_0 に音 x_1 がつながる.
- $Assert(s)$
  現時点の状況において成立していない基底状態 s を次の時点の状況において成立させる.
- $Retract(s)$
  現時点の状況において成立している基底状態 s を次の時点の状態において成立させない.

ここで, 動作 $Assert(s)$ と $Retract(s)$ は引数として状態をとり, 動作の定義に反している. しかし, s が変数記号となることがないので, 次のように考えることができる.

$$Assert(s_n(t_1, \cdots, t_n)) \equiv Assert_{s_n}(t_1, \cdots, t_n).$$
$$Retract(s_n(t_1, \cdots, t_n)) \equiv Retract_{s_n}(t_1, \cdots, t_n).$$

従って, この違反は本質的な問題にならない.

3.2 記述例: $\mathcal{D}_t$

$c_1)$ $Connect(x_0, x_3)$
$\Leftarrow$
$Offhook(x_0), Nomal(x_0),$
$DialTone(x_3).$

$c_2)$ $Connect(x_0, x_3), Connect(x_1, x_5)$
$\Leftarrow$
$Dial(x_0, x_6), DialNumber(x_1, x_6),$
$Connected(x_0, x_2), DialTone(x_2),$
$Ready(x_1),$
$CallTone(x_3), RingTone(x_5).$

$c_3)$ $Connect(x_0, x_3)$
$\Leftarrow$
$Dial(x_0, x_6), DialNumber(x_1, x_6),$
$Connected(x_0, x_2), DialTone(x_2),$
$Busy(x_1), BusyTone(x_3).$

$c_4)$ $Retract(Ready(x_0)), Assert(Busy(x_0))$
$\Leftarrow$
$Offhook(x_0), Ready(x_0).$

$c_5)$ $Retract(Ready(x_1)), Assert(Busy(x_1))$
$\Leftarrow$
$Connect(x_1, x_4), RingTone(x_4),$
$Ready(x_1).$

$c_6)$ $Connect(x_0, x_3), Connect(x_1, x_5)$
$\Leftarrow$
$Offhook(x_0), Hotline(x_0, x_1),$
$Ready(x_1),$
$CallTone(x_3), RingTone(x_5).$

$c_7)$ $Connect(x_0, x_3)$
$\Leftarrow$
$Offhook(x_0), Hotline(x_0, x_1),$
$Busy(x_1), BusyTone(x_3).$

3.3 解釈例: $\mathcal{I}_t$

状況 1, 2, 3 に共通の状況 S_0 を次に示す.

$$S_0 = \{Normal(P0), DialNumber(P0, 10),$$
$$Hotline(P1, P0), DialNumber(P1, 11),$$
$$IdleTone(T0), DialTone(T1),$$
$$CallTone(T2), RingTone(T3),$$
$$BusyTone(T4)\}$$

状況 1 に対する解釈

状況 1 S_1 を次に示す.

$$S_1 = S_0 \cup \{Ready(P0), Connected(P0, T0),$$
$$Ready(P1), Connected(P1, T0),$$
$$Offhook(P0)\}$$

節 c_1, c_4 が使われ, 解釈 $\mathcal{I}_t(\mathcal{D}_t, S_1)$ が次のように決まる.

$$\mathcal{I}_t(\mathcal{D}_t, S_1) = \{Connect(P0, T1),$$
$$Retract(Ready(P0)),$$
$$Assert(Busy(P0))\}$$

状況 2 に対する解釈

状況 2 S_2 を次に示す.

$$S_2 = S_0 \cup \{Ready(P0), Connected(P0, T0),$$
$$Ready(P1), Connected(P1, T0),$$
$$Offhook(P1)\}$$

最初に節 c_4, c_6 が使われ, その後, さらに節 c_5 が使われる. その結果, 解釈 $\mathcal{I}_t(\mathcal{D}_t, S_2)$ が次のように決まる.

$$\mathcal{I}_t(\mathcal{D}_t, S_2) = \{Retract(Ready(P1)),$$
$$Assert(Busy(P1)),$$
$$Connect(P1, T2),$$
$$Connect(P0, T3),$$
$$Retract(Ready(P0)),$$
$$Assert(Busy(P0))\}$$

状況 3 に対する解釈

状況 3 S_3 を次に示す.

$$S_3 = S_0 \cup \{Busy(P0), Connected(P0, T1),$$
$$Ready(P1), Connected(P1, T0),$$
$$Offhook(P1)\}$$

節 c_4, c_7 が使われ, その結果, 解釈 $\mathcal{I}_t(\mathcal{D}_t, S_3)$ が次のように決まる.

$$\mathcal{I}_t(\mathcal{D}_t, S_3) = \{Retract(Ready(P1)),$$
$$Assert(Busy(P1)),$$
$$Connect(P1, T4)\}$$

3.4 記述例: $\mathcal{D}_p$

$c_1)$ $Connect(x_0, x_3)$
$\Leftarrow$
$Offhook(x_0)$ |
$DialTone(x_3)$.

$c_2)$ $Connect(x_0, x_3), Connect(x_1, x_5)$
$\Leftarrow$
$Dial(x_0, x_6), DialNumber(x_1, x_6),$
$Connected(x_0, x_2), DialTone(x_2)$ |
$CallTone(x_3), RingTone(x_5)$.

$c_3)$ $Connect(x_0, x_3) \Leftarrow$
$Connect(x_0, x_2), CallTone(x_2),$
$Connect(x_1, x_4), RingTone(x_4),$
$Busy(x_1)$ |
$BusyTone(x_3)$.

$c_4)$ $Retract(Ready(x_0)), Assert(Busy(x_0))$
$\Leftarrow$
$Offhook(x_0), Ready(x_0)$ |.

$c_5)$ $Retract(Ready(x_1)), Assert(Busy(x_1))$
$\Leftarrow$
$Connect(x_1, x_4), RingTone(x_4),$
$Ready(x_1)$ |.

$c_6)$ $Connect(x_0, x_3), Connect(x_1, x_5)$
$\Leftarrow$
$Offhook(x_0), Hotline(x_0, x_1)$ |
$CallTone(x_3), RingTone(x_5)$.

$c_7)$ $Impossible$
$Connect(x_0, x_2), Connect(x_0, x_3)$.

3.5 解釈例: $\mathcal{I}_p$

状況 1, 2, 3 に共通の状況 S_0 を次に示す.

$$S_0 = \{DialNumber(P0, 10),$$
$$Hotline(P1, P0), DialNumber(P1, 11),$$
$$IdleTone(T0), DialTone(T1),$$
$$CallTone(T2), RingTone(T3),$$
$$BusyTone(T4)\}$$

状況 1 に対する解釈

状況 1 S_1 を次に示す.

$$S_1 = S_0 \cup \{Ready(P0), Connected(P0, T0),$$
$$Ready(P1), Connected(P1, T0),$$
$$Offhook(P0)\}$$

節 c_1, c_4 が使われ, 次に示すような提案 P_1 が求まる.

$$P_1 = \{\langle\{Connect(P0, T1)\},$$
$$\{Offhook(P0)\}\rangle,$$
$$\langle\{Retract(Ready(P0)),$$
$$Assert(Busy(P0))\},$$
$$\{Offhook(P0), Ready(P0)\}\rangle\}$$

提案 P_1 は 2 つの対をその要素とするが, 2 つの対の第 1 引数どうしは衝突しない. その結果, 棄却 R_1 は空集合となる.

$$R_1 = \{\}$$

従って, 解釈 $\mathcal{I}_p(\mathcal{D}_p, S_1)$ は次のように決まる.

$$\mathcal{I}_p(\mathcal{D}_p, S_1) = \{Connect(P0, T1),$$
$$Retract(Ready(P0)),$$
$$Assert(Busy(P0))\}$$

状況 2 に対する解釈

状況 2 S_2 を次に示す.

$$S_2 = S_0 \cup \{Ready(P0), Connected(P0, T0),$$
$$Ready(P1), Connected(P1, T0),$$
$$Offhook(P1)\}$$

最初に, 節 c_1, c_4, c_6 が使われ, その後, さらに節 c_5 が使われる. その結果, 次に示すような提案 P_2 が求まる.

$$
\begin{aligned}
P_2 = \{&\langle\{Connect(P1,T1)\},\\
&\{Offhook(P1)\}\rangle,\\
&\langle\{Retract(Ready(P1)),\\
&\ Assert(Busy(P1))\},\\
&\{Offhook(P1), Ready(P1)\}\rangle,\\
&\langle\{Connect(P1,T2), Connect(P0,T3)\},\\
&\{Offhook(P1), Hotline(P1,P0)\}\rangle,\\
&\langle\{Retract(Ready(P0)),\\
&\ Assert(Busy(P0))\},\\
&\{Offhook(P1), Hotline(P1,P0),\\
&\ RingTone(T3), Ready(P0)\}\rangle\}
\end{aligned}
$$

提案 P_2 は4つの対をその要素とするが, 1, 3番目の対の第1引数どうしが衝突する. その結果, 棄却 R_2 は次のように求まる.

$$
\begin{aligned}
R_2 = \{&\langle\{Connect(P1,T1)\},\\
&\{Offhook(P1)\}\rangle\}
\end{aligned}
$$

従って, 解釈 $\mathcal{I}_p(\mathcal{D}_p, S_2)$ は次のように決まる.

$$
\begin{aligned}
\mathcal{I}_p(\mathcal{D}_p, S_2) = \{&Retract(Ready(P1)),\\
&Assert(Busy(P1)),\\
&Connect(P1,T2),\\
&Connect(P0,T3),\\
&Retract(Ready(P0)),\\
&Assert(Busy(P0))\}
\end{aligned}
$$

状況3に対する解釈

状況3 S_3 を次に示す.

$$
\begin{aligned}
S_3 = S_0 \cup \{&Busy(P0), Connected(P0,T1),\\
&Ready(P1), Connected(P1,T0),\\
&Offhook(P1)\}
\end{aligned}
$$

最初に, 節 c_1, c_4, c_6 が使われ, その後, さらに節 c_3 が使われる. その結果, 次に示すような提案 P_3 が求まる.

$$
\begin{aligned}
P_3 = \{&\langle\{Connect(P1,T1)\},\\
&\{Offhook(P1)\}\rangle,\\
&\langle\{Retract(Ready(P1)),\\
&\ Assert(Busy(P1))\},\\
&\{Offhook(P1), Ready(P1)\}\rangle,\\
&\langle\{Connect(P1,T2), Connect(P0,T3)\},\\
&\{Offhook(P1), Hotline(P1,P0)\}\rangle,\\
&\langle\{Connect(P1,T4)\},\\
&\{Offhook(P1), Hotline(P1,P0),\\
&\ CallTone(T2), RingTone(T3),\\
&\ Busy(P0)\}\rangle\}
\end{aligned}
$$

提案 P_3 は4つの対をその要素とするが, 1, 3, 4番目の対の第1引数が互いに衝突する. その結果, 棄却 R_2 は次のように求まる.

$$
\begin{aligned}
R_3 = \{&\langle\{Connect(P1,T1)\},\\
&\{Offhook(P1)\}\rangle,\\
&\langle\{Connect(P1,T2), Connect(P0,T3)\},\\
&\{Offhook(P1), Hotline(P1,P0)\}\rangle\}
\end{aligned}
$$

従って, 解釈 $\mathcal{I}_p(\mathcal{D}_p, S_3)$ は次のように決まる.

$$
\begin{aligned}
\mathcal{I}_p(\mathcal{D}_p, S_3) = \{&Retract(Ready(P1)),\\
&Assert(Busy(P1)),\\
&Connect(P1,T4)\}
\end{aligned}
$$

4 考察

4.1 記述のモジュール性

本節では, 言語 $\mathcal{L}_t, \mathcal{L}_p$ による記述のモジュール性について考察する.

3.2, 3.4節の記述 $\mathcal{D}_t, \mathcal{D}_p$ のモジュール性を比較する. モジュール性の優劣の判定は, 普通の電話機の仕様(項目1-3)に関する記述と, ホットラインの電話機の仕様(項目4)に関する記述に, 分離できるか否かによって行なう.

ケーススタディ 1

電話機の受話器を上げた時に実行する動作は, その電話機が普通の電話機であるか, ホットラインの電話機であるかによって異る. この区別のために状態 $Nomal(x_0)$ と状態 $Hotline(x_0, x_1)$ が使われる. $\mathcal{D}_t$ では, 普通の電話機に関する節 c_1 に $Nomal(x_0)$ が現れ, ホットラインの電話機に関する節 c_6, c_7 に $Hotline(x_0, x_1)$ が現れる. 一方, $\mathcal{D}_p$ では, ホットラインの電話機に関する節 c_6 に $Hotline(x_0, x_1)$ が現れるだけである.

ところで, 普通の電話機とホットラインの電話機の区別が必要になるのは, ホットラインの電話機の仕様が与えられた時である. これは普通の電話機のみの仕様では, 受話器を上げた時に実行する動作が一意に決まり, そのような区別を必要としないことからも分かる. そう考えると, $\mathcal{D}_t$ における c_1 中の $Nomal(x_0)$ は, ホットラインの電話機の仕様追加に伴う変更であることが分かる. それに対し, $\mathcal{D}_p$ では, c_1 に $Nomal(x_0)$ が現れないので, そのような変更が不要であることが分かる.

ケーススタディ 2

普通の電話で相手の電話番号をダイヤルした時と，ホットラインの電話機の受話器を上げた時に実行する動作は，相手が話し中か $Busy(x_1)$ 否か $Ready(x_1)$ によって異なる．$\mathcal{D}_t$ では，話し中でない場合と話し中の場合の記述が，普通の電話機に関する節 c_2, c_3 とホットラインの電話に関する節 c_6, c_7 に現れる．一方，$\mathcal{D}_p$ では，話し中でない場合と話し中の場合の記述が，普通の電話機に関する節 c_2, c_3 とホットラインの電話に関する節 c_6 に現れる．

ところで，ホットラインの仕様は，項目 4 に記されている通り，次の 2 点を要求しているだけである．

- 相手の電話機があらかじめ決っている．
- 受話器を上げたら，すぐに相手の電話機を呼び出す．

従って，相手の電話機を呼び出そうとした時，相手が話し中であったならば実行する動作は，厳密に言えば，ホットラインの仕様ではないことになる．そう考えると，$\mathcal{D}_t$ における c_7 は，ホットラインの電話機の仕様追加に伴う追加であり，普通の電話機の仕様を維持するために必要であることが分かる．それに対し，$\mathcal{D}_p$ では，普通の電話，ホットラインの電話機に関わらず，相手の電話機が話し中の場合には c_3 が使われ，そのような追加が不要であることが分かる．

上の 2 つのケーススタディから，普通の電話機の仕様に関する記述とホットラインの電話機に関する記述の分離が，$\mathcal{D}_t$ では行なえず $\mathcal{D}_p$ では可能であることが分かる．従って，$\mathcal{D}_p$ が $\mathcal{D}_t$ よりもモジュール性に優れていることが結論される．そして，このようなモジュール性によって，言語 $\mathcal{L}_p$ では，システムの一部に閉じた記述が可能となる．

4.2 非単調推論との比較

言語 $\mathcal{L}_p$ の記述 $\mathcal{D}_{p1}, \mathcal{D}_{p2}$ および状況 S に対して，その解釈 $\mathcal{I}_p$ は非単調性 [1] をもつ．すなわち，$\mathcal{D}_{p1} \subseteq \mathcal{D}_{p2}$ かつ $\mathcal{I}_p(\mathcal{D}_{p1}, S) \not\subseteq \mathcal{I}_p(\mathcal{D}_{p2}, S)$ であるような $\mathcal{D}_{p1}, \mathcal{D}_{p2}$ が存在する．

また，言語 $\mathcal{L}_p$ は「例外」の記述を行なうことができる．このことは，3.4 節の記述 $\mathcal{D}_p$ に対して，「受話器を上げた時の動作に関してホットラインの電話機は電話機の例外である」というような解釈がなされていることからも分かる．

以下では，「例外」の記述の関して，幾つかの非単調推論との比較について考察する．

Reiter の Default Logic [1] [2]

Reiter の Default Logic による「鳥とペンギン」（ただし，ペンギンの例外も考慮する）の記述を次に示す．

$d_1)$ $Bird(x) : \mathbf{M}Fly(x) \ / \ Fly(x)$
$d_2)$ $Penguin(x) : \mathbf{M}\neg Fly(x) \ / \ \neg Fly(x)$
$w_1)$ $\forall x.Bird(x) \leftarrow Penguin(x)$
$w_2)$ $Penguin(Tweety)$

$D = \{d_1, d_2\}$ を Default の集合とする．$W = \{w_1, w_2\}$ を公理の集合とする．このとき，閉正規 Default Theory (D, W) には，2 つの Extension が存在する．一方の Extension では Tweety が飛び，他方の Extension では Tweety が飛ばないので，結局 Tweety が飛ぶか否かは決まらない．

このように，「飛ぶことに関してペンギンが鳥の例外である」という関係は，Defalt だけで表すことができない．

Circumscription [1] [3]

一階述語論理式による「鳥とペンギン」の記述を次に示す．

$a_1)$ $\forall x.Bird(x) \leftarrow Penguin(x)$
$a_2)$ $\forall x.Fly(x) \leftarrow Bird(x) \wedge \neg Ab_1(x)$
$a_3)$ $\forall x.\neg Fly(x) \leftarrow Penguin(x) \wedge \neg Ab_2(x)$
$a_4)$ $Penguin(Tweety)$

$A = \{a_1, a_2, a_3, a_4\}$ を公理の集合とする．このとき，次の並列 circumscription から Tweety が飛ばないことが結論される．

$$Circum(A; Ab_2; Ab_1, Fly)$$
$$\wedge Circum(A; Ab_1; Fly)$$

このように，「飛ぶことに関してペンギンが鳥の例外である」という関係は，並列 circumscription により，もとの一階述語論理式と独立に記述される．

Poole の Default Logic [4] [5]

Poole の Default Logic による「鳥とペンギン」の記述を次に示す．

$\delta_1)$ (x) $Assume$ $Fly(x) \leftarrow Bird(x)$
$\delta_2)$ (x) $Assume$ $\neg Fly(x) \leftarrow Penguin(x)$
$f_1)$ $\forall x.Bird(x) \leftarrow Penguin(x)$
$f_2)$ $Penguin(Tweety)$

$\Delta = \{\delta_1, \delta_2\}$ を Defalt の集合とする. $F_n = \{f_1\}$ を必然的事実の集合とする. $F_p = \{f_2\}$ を偶然的事実の集合とする. このとき, 2つの Solution が存在する. 一方の Solution では Tweety が飛び, 他方の Solution では Tweety が飛ばないので, Theory Comparator により最も特殊な Solution を選ぶ. その結果, Tweety が飛ばないことが結論される.

このように, 「飛ぶことに関してペンギンが鳥の例外である」という関係は, 記述中に暗に与えられ, Theory Comparator により導かれる. しかし, Poole の定義による Theory Comparator は矛盾しない結論を全て結論することができない. これは, 次の示す Δ, F_n, F_p に対して, G_1 が結論されないことから分かる.

$$\Delta = \{C \leftarrow B, \neg C \leftarrow A, \\ F \leftarrow E, \neg F \leftarrow D\}$$
$$F_n = \{B \leftarrow A, E \leftarrow D, \\ G_1 \leftarrow \neg C \wedge \neg F, G_2 \leftarrow C \wedge F\}$$
$$F_p = \{A, D\}$$

言語 $\mathcal{L}_p$

言語 $\mathcal{L}_p$ と一階述語論理式による「鳥とペンギン」の記述を次に示す.

$c_1)$ $Fly(x) \Leftarrow Bird(x)$ |.
$c_2)$ $Not\_Fly(x) \Leftarrow Bird(x), Penguin(x)$ |.
$c_3)$ $Impossible\ Fly(x), Not\_Fly(x)$.
$a_1)$ $\forall x. Bird(x) \leftarrow Penguin(x)$.
$a_2)$ $Penguin(Tweety)$.

$\mathcal{D}_p = \{c_1, c_2, c_3\}$ を言語 $\mathcal{L}_p$ の記述とする. $A = \{a_1, a_2\}$ を公理の集合とする. $Th(A)$ を公理 A から導ける定理の集合とする. このとき, 解釈 $\mathcal{I}_p(\mathcal{D}_p, Th(A))$ から Tweety が飛ばないことが結論される.

このように, 「飛ぶことに関してペンギンが鳥の例外である」という関係は, $\mathcal{D}_p$ 中に陽に記述される.

5 おわりに

本稿では, 状況 - 動作規則を記述する言語における記述のモジュール性について論じた. 2節において, Total な言語 $\mathcal{L}_t$ と Partial な言語 $\mathcal{L}_p$ を定義した. 3節において, $\mathcal{L}_t, \mathcal{L}_p$ による交換機の動作仕様の記述と解釈を例示した. 4.1節では, $\mathcal{L}_t, \mathcal{L}_p$ の記述のモジュール性について考察した. その結果, 普通の電話機の仕様とホットラインの電話機の仕様の分離可能性から, $\mathcal{L}_p$ の記述が $\mathcal{L}_t$ の記述よりもモジュール性に優れていることを示した. 4.2節では, 「例外」の記述に関して, 幾つかの非単調推論との比較について考察した. その結果, 「例外」の記述が可能か否か, 陽に現れるか否か, あるいは, 独立に与えられるか否かという点において, それらの間に違いがあることを指摘した.

最後に, 今後の課題について述べる. まず, 言語 $\mathcal{L}_p$ を解釈するプログラムを作成し, 種々のシステムの動作仕様の記述を試みる必要がある. それによって, $\mathcal{L}_p$ の記述性の良さを検証するとともに, その限界を明らかにしたい. また, 動作 - 状況変化規則の記述言語についても検討する必要がある. それによって, 要求仕様記述言語へと $\mathcal{L}_p$ を拡張して行きたい.

謝辞 本研究の機会を与えて頂きました葉原会長ならびに山下社長に感謝致します.

参考文献

[1] 有馬 淳, 佐藤 健: 非単調推論, 知識プログラミング, 共立出版, 1988.
[2] 相田 仁: デフォルトを用いた推論と非単調論理, 人工知能学会誌 Vol.2 No.1, 1987.
[3] 中川裕志: 論理 + サーカムスクリプション = 常識推論, 人工知能学会誌 Vol.2 No.1, 1987.
[4] 國藤 進: 仮説推論, 人工知能学会誌 Vol.2 No.1, 1987.
[5] Poole, D. L.: On the Comparison of Theories: Preferring the Most Specific Explanation, Proc. of the 9th IJCAI, 1985.