人為的欠陥の意味と耐性解析

永藤 直行　　渡部 卓雄

人為的欠陥に対する作業手順の耐性をモデル検証の手法を用いて解析する枠組を提案する．多くの場合，作業手順には人為的欠陥による誤り検出や復旧のためのタスクが埋め込まれている．作業手順の信頼性を保持するためには検出や復旧タスクの有用性を検証する必要がある．有用であることを示すために, 耐性を解析する時には欠陥を埋め込む．ここでは，作業について訓練された人間が引き起こす人為的欠陥の分類をもちい，各分類の特徴の意味を遷移システム上で定義する．解析時には，欠陥を含まない振舞モデルを人為的欠陥の意味にしたがって変換し，そのモデルをモデル検証の手法をもちいて検査する．検査する性質は作業手順の目的の否定となる．もし反例が見つかれば，そのときにはその作業手順は埋め込んだ人為的欠陥に対して耐性がないと云うことができる．

1　はじめに

工業プラント，飛行機，病院など多くの状況で作業手順が存在している．手順の信頼性は，高度に訓練された人間のスキルだけでなく手順自身がディペンダブルであることによって達成される．よく訓練された人間でも人為的欠陥を引き起こし得るので，作業手順が人為的欠陥による誤りを発見し，修復するように設計されていなければならない．

人為的欠陥は，悪意をもって行われるかや故意によるかと云う視点から分類がなされている ([1] の図.6 参照)．本稿では故意に悪意のある人為的欠陥は対象としない．良く訓練を受けた人間が運用時に不作為に引き起こす人為的欠陥を対象とする．

心理学的な視点で思考のレベルによる人為的欠陥の分類がなされている ([8],pp.53–96 参照)．思考のレベルを skill-base, rule-base, knowledge-base にわけ，内面から考察している．しかし，我々はいま思考の結果として得られる人間の出力としての行動に興味がある．Discrete Action Classification(DAC) [10] は出力として (システムへの入力として) の正しくない行動を分類し特徴づけている．訓練を受けた人間が起こす人為的欠陥となる正しくない行動を分類している．その分類を以下に示す．人為的欠陥は omission と commission の 2 つに分類され，commission faults はさらに詳細に分類されている．

- Faults of Omission:
  - omits entire task
  - omits a step in a task
- Faults of Commission:
  - Selection fault: selects wrong control, mispositions control or, issues wrong command and information.
  - Sequence fault: out of order.
  - Timing fault: too early or too late.
  - Qualitative fault: too much or too little

本論文は DAC にしたがって議論する．

Omission fault は正しいタスクを何らかの理由で行わない人為的欠陥である．さらに，タスク全体を行わないか，タスクの一部を行わないかで分類されている．我々は，タスクはアトミックであると仮定し

Semanitcs of Human-Made Fault and Robustness Analysis

Naoyuki Nagaotu, 有限会社プレシステム, PRESYSTEMS Inc..

Takuo Watanabe, 東京工業大学情報理工学研究科計算工学専攻, Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology.

てモデル化を行う．よって，”omits entire task” について意味を定義し， ”omits a step in a task” は対象としない．Commission fault は何かタスクは実行しているが正しくない何かをしてしまう人為的欠陥である．我々は commission fault のうち selection fault と sequence fault について，その意味を定義する．利用するモデル検証器のモデリング言語では時間の概念を持っていないので timing fault は対象としない．また，タスクの質に関する形式化はここでは対象としない．よって， qualitative fault は対象としない．

Krishnan [4] は多重化されたシステムにおける欠陥を omission fault と value fault と addition fault であるとしている．彼の omission fault は，同期を取るようなシステムにおいてメッセージを送信しない欠陥としている．Value fault は期待される動作とは異なる動作をする欠陥としている． Addition fault は期待されないメッセージを送信する欠陥であるとしている．その意味を CCS における refinement 関数を用いて形式化し，耐性システムの特徴を捉えようとしている．この他にも欠陥を定義して partial model checking や高階論理と型を元に議論した報告もある [9] [2] [3]．我々は，システムの耐性ではなく，人間が動作主体となる作業手順の耐性について議論する．

先に述べたように， DAC の分類のうち ”omits entire task[†1]” と ”selection fault” と ”sequence fault” を対象に，その意味をプロセス代数 CCS [5] の上で定義する．期待される動作であるタスクはアクションとしてあらわすことにする．Omission fault は期待されるタスクをするかどうかであるので，対応するアクションを含むプロセス式と含まないプロセス式の選択合成であらわす． Selection fault は期待されないタスクに対応するアクションを含むプロセス式との選択合成であらわす． Sequence fault はタスクに対応したプロセスの同期合成を用いてあらわす．

CCS 上で定義された意味はモデルに欠陥を埋め込むために利用される．モデルは期待されたタスクの列としての作業手順をプロセス式であらわしたもので

†1 以降では，これを omission fault と呼ぶことにする．

Scenario

1. A patient gives a clerk in the room his id-card.
2. The clerk makes the receipt system read the id card.
3. The system gives the clerk two receipt sheets and gives a nurse labels.
4. The clerk gives the patient one of two and gives a nurse another.
5. The nurse prepares tubes.
6. The nurse makes the system read the labels.
7. The nurse puts the labels on the tubes.
8. The system calls a receipt id.
9. The nurse confirms the name on the labels with a name on the sheet.
10. The nurse receives a sheet from the patient after having a seat.
11. The nurse compares the name on the sheet, given by the clerk, with his name.
12. The nurse extracts blood samples from the patient.
13. The nurse takes the blood samples to a lab.

図 1　採血手順の例

ある．人為的欠陥が起こると重大な事故を引き起こすであろうと予測されるタスクが与えられるとして，そのタスクのところに欠陥を埋め込む．埋め込んだモデルに対して，作業手順の目的をあらわした時相論理式に反する例をモデル検証を利用して解析する．何らかの反例が見つかった時には埋め込んだ欠陥に対してその作業手順は耐性を持っていないと判断できる．簡単な適用例として採血の手順を対象にした検証結果を示す．

本稿は以下のように構成される． 2 節では提案手法が必要となることを例で示す． 3 節ではモデルの記述と手順の目的を記述するための体系について述べる． 4 節では人為的欠陥の形式的な意味を与える．5 節でモデル検証を利用する枠組について述べる．最後に 7 節で本研究の概要をまとめ将来の展望について述べる．

2　動機づけ

作業手順の例として血液検査の一部をなす採血の手順 (図.1) をあげる．実際には分岐を含んだ複雑なもの

になっている．ここでは，そのうちのひとつの作業の流れを抽出したものを示している．この手順は，患者 (patient) が採血室に到着し受付で id カードを提示する所から始まっている．受付係 (clerk) は id カードを受け取りそれを受付システム (receipt system) に読み込ませる．受付システムは血液検査のオーダのデータベースを検索し，該当するオーダがあれば受付表 (receipt sheet) 2 枚とラベル (label) を発行する．受付表には受付番号と患者氏名と id が印刷されている．受付係は 1 枚を患者に渡し，もう 1 枚とラベル (label) を所定の位置に受付順に置く．看護士 (nurse) は採血管 (tube) を準備する．準備が済んだらラベルを受付システムに読み込ませ，ラベルを採血管に貼る．受付システムは対応する受付番号を呼び出す．看護士はラベル上の患者氏名と受付表の患者氏名を画面の名前で確認する．呼び出して椅子にかけた患者から名前を名乗ってもらい，それと受付係から受け取った受付表の氏名を比較する．それが一致していたならば，患者から血を採取する．採取した採血管と受付表を束ねてそれを検査室に看護士が持って行く．

図.1 の手順において重要なタスクは 9 と 11 である．2 回同じものを確認すれば人為的欠陥を回避できる可能性があるとして導入している．採血の前にこの 2 つのタスクのいずれかで取り違いが発見されるように設計されている．患者の取り違いを発見したときのリカバリの手順が存在するはずであるが，この手順では規定していないのでリカバリを表す単純なタスクを仮定して 6 節で述べる振舞モデルを作成する．

複数の患者の採血をひとりの看護士が同時に行っていたとして，何らかの理由により採血管や受付表やラベルが入れ替わってしまっている状況を考える．そのような状況でいずれかのタスクを看護士が怠ってしまったとしても発見できるかどうか検証する．看護士がいずれかのタスクで起こす人為的欠陥について耐性があれば，高い信頼性を保証することになる．

3 準備

この節では本稿で利用する理論体系について述べる．期待される手順をあらわすモデルを記述するためと人為的欠陥の意味を定義するために Milner の CCS のサブセットを利用する．また，作業手順の目的を記述するために有限列についての線形時相論理を利用する．

3.1 プロセス代数

CCS はプロセス代数のひとつの体系である．シーケンシャルな振舞をする複数のプロセスの同期通信を元にした体系である．同期通信は入力アクションと出力アクションという 2 種類のアクションの実行で生じる．入力アクションは適当な名前がつけられる．名前の集合を A とする．$a \in A$ と同期する出力アクションを $\overline{a}$ で与える．$\overline{A}$ は A と互いに素であるとする．$A \cup \overline{A}$ の要素を $\alpha, \beta, \cdots$ であらわすことにする．

プロセス式の構文を以下のように与える．E, F はプロセス，a は名前とする．

- 出力アクション $(\overline{a}(e_1, \cdots, e_n) : E)$ はプロセスとする
- 入力アクション $(a(x_1, \cdots, x_n) : E)$ はプロセスとする
- E と F の選択合成 E+F はプロセスとする
- E と F の同期合成,$E|F$ はプロセスとする
- プロセス定数 $P(x_1, \cdots, x_n)$ もプロセスで，$P(x_1, \cdots, x_n) \stackrel{\text{def}}{=} E$ によって与えられる
- 条件式 `if` (b) (E) (F) もプロセスとする

ここで，$x_1, \cdots$ は `V` 上の変数，$e_1, \cdots$ は `V` の式，b は `V` 上の論理式とする．$STOP$ という特別なプロセスを導入する．†2．何もしないと云う特別な意味を持つ．各意味は CCS と同じであるのでここでは省略する．

3.2 有限列上の LTL

作業手順の目的をあらわすために線形時相論理 LTL をもちいる．作業手順は初期状態をもち，有限列 $\sigma = s_0\, s_1 \cdots s_n$ であると仮定する．有限列 σ の長さを $|\sigma|$ と書き，$|\sigma|$ は $n+1$ とする．有限列 σ の i で始まるサフィックスを $\sigma^{i\cdots} = s_i \cdots s_n$ として，i 番目の状態を σ^i と書くことにする．

作業手順の目的を以下ように再帰的に定義される

†2 CCS では $ZERO$ としているが，本稿では $STOP$ とする．

論理式で与えることにする．ϕ, ψ は時相論理式とする．そのとき，

- 状態論理式 p は時相論理式
- 時相論理式の否定 $\neg\varphi$ は時相論理式
- $\varphi \vee \psi$ と $\varphi \wedge \psi$ は時相論理式
- $\Box\varphi, \Diamond\varphi, \circ\varphi, \varphi\,\mathcal{U}\,\psi$ は時相論理式

である．$\mathtt{V}$ 上の変数 $x, y, z, \cdots$ とする．状態論理式 p は $\mathtt{V}$ 上で構成される．もし p が s で真となるとき $s[\![p]\!] = \mathtt{tt}$ となる.. また， $\mathtt{tt}$ と $\mathtt{ff}$ はそれぞれ真と偽を意味する．

各時相論理式の有限列上の意味を定義する．もし有限列 σ が φ を充たしている時， $\sigma \models \varphi$ と書くことにする．

$$
\begin{array}{lcl}
\sigma \models p & \Leftrightarrow & |\sigma| \neq 0 \text{ and } \sigma^0[\![p]\!] = \mathtt{tt}, \\
\sigma \models \neg\varphi & \Leftrightarrow & |\sigma| \neq 0 \text{ and } \sigma \not\models \varphi, \\
\sigma \models \varphi \wedge \psi & \Leftrightarrow & \sigma \models \varphi \wedge \sigma \models \psi, \\
\sigma \models \varphi \vee \psi & \Leftrightarrow & \sigma \models \varphi \vee \sigma \models \psi, \\
\sigma \models \mathsf{X}\varphi & \Leftrightarrow & \sigma^{1\cdot\cdot} \models \varphi, \\
\sigma \models \varphi \mathsf{U} \psi & \Leftrightarrow & \exists\, k \text{ s.t. } \sigma^{k\cdot\cdot} \models \psi \text{ and} \\
 & & \text{for every } j < k,\ \sigma^{j\cdot\cdot} \models \varphi.
\end{array}
$$

有限列の集合 T と時相論理式 φ が与えられたとして，全ての $\sigma \in T$ について $\sigma \models \varphi$ であるとき φ は T で妥当であると云う．

3.3 プロセスと時相論理式の関係

プロセスは $a(e)$ と $\bar{a}(x)$ によって値を送受信することができる．このとき e の評価値を x へ束縛する単一の代入と見なすことにする．この代入はプロセスであらわされたモデルの状態を変更する．これを $s[v/x]$ であらわすことにする．ここで，v は e の s における評価値とする．同期の列は代入の列を生成し，さらに状態の遷移を生成すると考えることにする．たとえば，p を $(x = y)$ とすると $s[v/x][\![x = y]\!]$ は $v = s[v/x][\![y]\!]$ となる．

4 人為的欠陥の意味

人為的欠陥の意味をプロセス代数で定義する．この形式化は人為的欠陥が起こるかも知れないタスクに対応するプロセスを生成する．

4.1 作業手順の構造

人為的欠陥の形式的な意味を定義する前に作業手順の構造について考える．作業手順はタスクの列の集合と見なす．タスクはそれを行う主体と対象を持つ．タスクが実行された時には主体から対象にメッセージが送信される．メッセージによりある種の情報が伝達される．メッセージはその送信と受信によって生起される．また，タスクは互いに依存関係が存在し，その依存関係はタスク間の順序としてあらわされる．ここで云うタスクはアトミックであると仮定する．

作業手順は 3 つ組 $(\mathcal{C}, \mathcal{T}, \mathcal{O})$ によって特徴づけられる．$\mathcal{C}$ は タスクの主体と対象の集合． $\mathcal{T}$ はタスクの集合で，タスクの依存関係は $\mathcal{O}(\subseteq \mathcal{T} \times \mathcal{T})$ によってあらわされる．たとえば， 図.1 の採血手順の例では，

- $\mathcal{C}$ ={patient, nurse, label, receipt, clerk, tube, laboratory},
- $\mathcal{T}$ ={”give a clerk id-card”, ”make system read id-card”, ”give clerk two sheets”, ”give receipt to nurse”,$\cdots$},
- $\mathcal{O}$ ={(”give a clerk id-card”, ”make system read id-card”), (”make system read id-card”, ”give clerk two sheets”),$\cdots$}.

となる．

本稿では，耐性の解析をする時には実際の意味は考えていない．手順に現れるタスクは単なる記号であり，その実際の意味は状況によって異なる．作業手順の耐性を検証して得られた結果は記号としてのタスクからある状況で解釈が与えられたとして，意味を考えなければならない．

4.2 人為的欠陥の埋め込み

人為的欠陥の意味を期待される作業手順のモデルから人為的欠陥が埋め込まれた作業手順のモデルへの変換を示す．本稿では， 3 節で述べた体系の制約から omission fault と selection fault と sequence fault を扱うことにする．作業手順の振舞モデル $W = (\mathcal{C}, \mathcal{T}, \mathcal{O})$ における欠陥タスクの集合 H が与えられたとする．

最初に omission fault について述べる．$t \in H$, $H \subseteq \mathcal{T}$ として，$(t_1, t), (t, t_2) \in \mathcal{O}$ である各 $t_1, t_2 \in \mathcal{T}$

について，$\{(t_1, t_2)\}$ を $\mathcal{O}$ に追加する．つまり $\mathcal{O}' = \mathcal{O} \cup \{(t_1, t_2)\}$ とする．$\mathcal{O}'$ により新たに $\mathcal{W}_H = (\mathcal{C}, \mathcal{T}, \mathcal{O}')$ を得る．

$\mathcal{T} = A \cup \overline{A}$ として，この変換を 3 節のプロセス代数であらわす．選択合成をもちいて

$$(\alpha(e) : STOP) + (STOP))$$

ここで $\alpha \in A \cup \overline{A}$.

つぎに， selection fault について述べる．人為的欠陥を起こすタスクの集合 $H \subseteq \mathcal{T}$ と期待されないタスクの集合 $H' \nsubseteq \mathcal{T}$ が与えられたとする．$t \in H$, $t' \in H'$ として，もし $(t_1, t), (t, t_2) \in \mathcal{O}$ であるならば，t を t' で置き換えた (t_1, t') と (t', t_2) を得る．$\{(t_1, t'), (t', t_2)\}$ を $\mathcal{O}$ に追加して $\mathcal{O}' = \{(t_1, t'), (t', t_2)\} \cup \mathcal{O}$ と $\mathcal{T}' = H' \cup \mathcal{T}$ より，$W_{H,H'} = (\mathcal{C}, \mathcal{T}', \mathcal{O}')$ を得るとしてこの変換を選択合成をもちいて

$$\alpha(e_1) : STOP + \alpha'(e') : STOP$$

とあらわす．ここで $\mathcal{T}' = A \cup \overline{A}$ として，$\alpha(e_1), \alpha'(e') \in A \cup \overline{A}$. Omission fault と selection fault をあわせて

$$(\alpha(e_1) : STOP) + (\alpha'(e') : STOP) + STOP)$$

とすることができる．

最後に sequence fault について述べる．$t_1, t_2 \in H$ として，$\mathcal{O}$ は $\mathcal{O}' = \{(t_1, t_2), (t_2, t_1)\} \cup \mathcal{O}$ となる．この $\mathcal{O}'$ から $\mathcal{W}_H = (\mathcal{C}, \mathcal{T}, \mathcal{O}')$ を得ることができる．各 $t_1, t_2 \in H$ について 2 つのプロセスを定義する．以下のようになる．ここではプロセス名をそれぞれ $Task\_t1$, $Task\_t2$ している．

$$\begin{aligned} Task\_\alpha1 &\stackrel{\text{def}}{=} (\alpha_1(e_1) : STOP), \\ Task\_\alpha2 &\stackrel{\text{def}}{=} (\alpha_2(e_2) : STOP) \end{aligned}$$

ここで，$\mathcal{T}' = A \cup \overline{A}$ として，$\alpha_1(e_1), \alpha_2(e_2) \in A \cup \overline{A}$ とする．同期合成をもちいて以下のようなプロセスであらわせる．

$$Task\_\alpha1 \mid Task\_\alpha2$$

omission fault, selection fault，sequence fault のいずれかが起こるとして，

$$\begin{aligned} Task\_\alpha1 &\stackrel{\text{def}}{=} ((\alpha_1(e_1) : STOP \\ &\quad + \alpha_1'(e_1') : STOP) + STOP) \\ Task\_\alpha2 &\stackrel{\text{def}}{=} ((\alpha_2(e_2) : STOP \\ &\quad + \alpha_2'(e_2') : STOP) + STOP) \end{aligned}$$

と $Task\_\alpha1 \mid Task\_\alpha2$ であらわせる．この形式化をもちいて図.1 に埋め込んだモデルを図.2 に示す．$H = \{$"confirm label", "compare receipt"$\}$ と仮定している．

5 検証の枠組

この節では欠陥を埋め込んだうえで，作業手順が目的を達成できるかどうか解析する枠組を与える．ロバストネスは外的要因に依存した属性である．作業手順のロバストネスを以下のように定義する．

定義 1 (Robustness) e をエラー状態をあらわす論理式， $\hat{e}$ 番目の状態はエラー， φ は目的をあらわす時相論理式とする．もし $\hat{e}$ が σ 上に存在し $\sigma^{\hat{e}\cdot\cdot} \models \Diamond\varphi$ ならば σ は e でロバストであるとして，σ は (e, φ)-*robust* であるという．もし φ が W から得られるすべての状態列について妥当であるならば，W は (e, φ)-*robust* である．

この定義は直観的には状態列 σ がエラー状態から復帰して φ を充たすゴールに至ることができることを意味している．

ロバストネスを保証する技術について考える．このテクニックはエラーの発見とリカバリを含んでいる．リカバリはエラーハンドリングとフォルトハンドリングからなる．エラーハンドリングはエラーを一時的に解消することであり，この場合再びエラーが出現するかもしれない．フォルトハンドリングはファルトを解消し，再びそのフォルトは起こらないようにする．本稿ではエラーハンドリングについて議論することにする．人為的欠陥は完全に解消することは難しいと考えられるからである．

ここで，リカバリを定義する．リカバリは直観的にはエラーになった後でも期待される状態に復帰することとする．

定義 2 (Recovery) σ を状態列， e をエラー状態をあらわす論理式，r をリカバリ状態をあらわす論理式とする．もし $\sigma \models \Box(\neg e) \vee \Diamond(e \wedge \Diamond r)$ ならば σ はリカバリ可能であるという．

いま， $\Box(\neg e) \vee \Diamond(e \wedge \Diamond r)$ を $e \rhd r$ と書くことにする．

補題 1 () σ が与えられたとして，e_i をエラー状態を

$Task\_confirm\_label \stackrel{\text{def}}{=}$
$((confirm\_label(name\_on\_display\_TaskConfLbl):$
$(if(name\_on\_label\_TaskConfLbl$
$= name\_on\_display\_TaskConfLbl)$
$(STOP)$
$(\overline{recovery\_label}($
$name\_on\_display\_TaskConfLbl):$
$recovery\_label(name\_on\_label\_Nurse):$
$STOP)))$
$+STOP)$
$Task\_compare\_receipt \stackrel{\text{def}}{=}$
$((compare\_receipt($
$name\_on\_display\_TaskCmpRecpt):$
$(if(name\_on\_receipt\_of\_patient\_TaskCmpRecpt$
$= name\_on\_display\_TaskCmpRecpt)$
$(STOP)$
$(\overline{recovery\_receipt}($
$name\_on\_display\_TaskCmpRecpt):$
$recovery\_receipt($
$name\_on\_receipt\_of\_patient\_Nrs2):$
$STOP)))$
$+STOP)$

$Nurse2 \stackrel{\text{def}}{=}$
$(receive\_receipt($
$name\_on\_receipt\_for\_patient\_Nrs2):$
$(Task\_compare\_receipt($
$name\_on\_receipt\_for\_patient\_Nrs2)$
$|Nurse1))$
$Nurse1 \stackrel{\text{def}}{=}$
$(\overline{extract\_blood}:$
$\overline{take\_sample\_to\_lab}($
$name\_on\_receipt\_for\_patient\_Nrs2,$
$name\_on\_label\_Nurse):$
$Nurse)$
$Nurse \stackrel{\text{def}}{=}$
$(give\_receipt\_to\_nurse(receipt\_id\_Nrs,$
$name\_on\_receipt\_Nurse):$
$give\_label(receipt\_id\_on\_label\_Nrs,$
$name\_on\_label\_Nurse):$
$\overline{prepare\_tube}:$
$\overline{make\_system\_read\_label}:$
$\overline{put\_label\_on\_tube}(name\_on\_label\_Nrs):$
$(Task\_confirm\_label(name\_on\_label\_Nrs)$
$|Nurse2))$

図 2　人為的欠陥を埋め込んだ振舞モデル

あらわす論理式，r_i はリカバリされた状態をあらわす論理式とする．このとき，もし $\sigma^{\hat{r_i}\cdots} \models (\bigwedge_i r_i) \wedge \Diamond\varphi$ かつ $\sigma \models \bigwedge_i (e_i \rhd r_i)$ ならば $\sigma \models (\bigwedge_i (e_i \rhd r_i)) \Rightarrow \Diamond\varphi$ となる．

証明 1 e_i がエラー状態論理式，r_i がリカバリ状態論理式とする．$\sigma \models (e_i \rhd r_i)$ が σ 上で成り立つと仮定する．

$$\sigma \models (e_i \rhd r_i)$$
$$\textit{iff} \quad \sigma \models \Box(\neg e_i) \vee \Diamond(e_i \wedge \Diamond r_i)$$
$$\textit{iff} \quad \sigma \models \Box(\neg e_i) \ \textit{or}\ \sigma \models \Diamond(e_i \wedge \Diamond r_i)$$

$\Diamond(e_i \wedge \Diamond r_i)$ が成り立つ場合，e_i と $\Diamond r_i$ が成り立つ s_i が存在する．すべての r_i が s_j で成り立つとして，$\sigma^{s_j\cdots} \models (\bigwedge_i r_i) \wedge \Diamond\varphi$ を仮定する．ここで $j \geq i$ ．そのとき φ が真となる状態 s_k が存在する．そこで，$\sigma \models (e_i \rhd r_i) \Rightarrow \Diamond\varphi$ となる．もし全てのエラー状態 e_i で $\sigma \models (e_i \rhd r_i) \Rightarrow \Diamond\varphi$ になるとすれば，$\sigma \models \bigwedge_i (e_i \rhd r_i) \Rightarrow \Diamond\varphi$ となる．

もう一方の $\neg e_i$ が常に成り立つ場合について考える．$\Diamond\varphi$ は明らかに成り立つ．なぜならフォルトを埋め込む前の作業手順のモデルはすでに何らかの方法で検証が済んでおり，必ず全ての列でゴールに至ると仮定しているからである． ■

補題 1 より，もし $\sigma \models \neg\Diamond\varphi$ となる σ が存在しているとと $\sigma \models \neg(e_i \rhd r_i)$ となる e_i が存在することになる．これはリカバリできないエラー状態が存在することを云っている．ゆえに，ゴールをあらわす論理式が与えらるとして，その否定を与えることでモデル検証の手法をもちいてリカバリ可能性が検証できる．

6 Model Checking

モデル検証ツール [6] [7] を使って図.1 の作業手順に人為的欠陥を埋め込んで耐性を検査する．図.1 の to confirm label と to compare name は誤りを発見するタスクである．この場合，患者を取り違えないための重要なタスクである．そこで，この 2 つのタスクに 4 節で示した人為的欠陥を埋め込むことにする． $H = \{$"confirm label", "compare name"$\}$ とする．図.2 はそれぞれのタスクに人為的欠陥を埋め込んだモデルを示している．この他に患者が受付番号を間違えて採血の場所に来てしまう人為的欠陥を起こすとする．その振舞をあらわすプロセス *Patient* が定義されている．

モデル検証するために作業手順の目的を LTL で与えなければならない．図.1 での目的は患者を取り違

$$Lab \stackrel{\text{def}}{=} (take\_sample\_to\_lab(name\_on\_receipt\_Lab, name\_on\_label\_Lab) : Lab)$$

図 3　検査室の作業モデル

$$\begin{aligned}&(Patient(\text{"}Alice\text{"})\\ &|(Patient(\text{"}Bob\text{"})\\ &|Receipt(1, 100)\\ &|Clerk(1)\\ &|Tube(1)\\ &|Lab(1)\\ &|Nurse(1))\end{aligned}$$

図 4　採血の状況

えることなく検査室に検体を持っていくと云うことである．これを看護士が検査室に検体を持って行った時点でラベルの名前と受付表の名前が一致していることであると読み変える．いま，検査室をあらわすモデルを図.3 のように与えることにする． このとき，目的を (1) の時相論理式であらわす．

$$\Box\Diamond(name\_on\_receipt\_Lab = name\_on\_label\_Lab) \tag{1}$$

リカバリ可能性をモデル検証器 [7] で検証する．このツールは emptyness を検証するので (1) の否定を与えなければならない．よって，与える論理式は

$$\Diamond\Box\neg(name\_on\_receipt\_Lab = name\_on\_label\_Lab) \tag{2}$$

となる．また，この検証器はモデルのインスタンスを与えなければならない．インスタンスはある状況を再現したのものになる．ここでは 2 人の患者を 1 人の看護士が同時に採血する状況を再現する．例えば図.4 のように与える．Nurse と Lab 以外の詳細はここでは省略する．

最初に，患者は人為的欠陥を生じるが看護士は人為的欠陥を生じないとした場合について検証を行った．患者が受付番号を間違えて採血にくるという人為的欠陥を考えている．この場合 (2) を充たす列は発見できなかった．看護士が期待された振舞をしている時には間違いを発見してリカバリできるであろう．しかし，看護士にも ”confirm label” や ”compare names” を行わないと云った人為的欠陥が生じるとして検証した場合は反例を観測できる．よって看護士に人為的欠陥を生じた場合は患者の間違いを発見しリカバリできない可能性があると云える．

表 1　モデル検証の結果

患者数	作業領域	患者取り違い事故
1	—	起こらない
2	分けていない	起こる
2	分けている	起こらない

つぎに，患者には人為的欠陥は生じないとして，看護士は ”confirm label” や ”compare names” を行わないと云った人為的欠陥が生じる検証を行う．患者が 1 人だけの場合，患者が 2 人で作業スペースが別れていない場合，患者が 2 人で作業スペースが別れている場合である．いずれの場合も看護士は 1 人としている．それぞれで反例を得られるかどうかの結果を表. 1 に示す．また，作業スペースが別れていない場合 ”confirm label” と ”compare names” では十分でないことが反例から云える．このいずれかを怠ったときには患者を取り違えてしまう反例が見つかっている．

7　おわりに

作業手順の人為的欠陥に対する耐性をモデル検証をもちいて解析する方法を提案した．人間の思考の出力としての行動に着目して人為的欠陥を分類し，その形式的な意味をプロセス代数で定義している．期待される手順のモデルにこの意味をもとに人為的欠陥を埋め込んでいる．作業手順の耐性を検証するときには，人為的欠陥が埋め込まれたモデルに対する作業手順のリカバリーを定義した．手順の目的をあらわす時相論理式が与えられたとして，その反例が見つかる時にはリカバリーができていないことを示した．この定理をもとにモデル検証をもちいて耐性を検査する方法を提案している．また，採血の手順に適用して有用性を示した．

我々の解析はタスクを表すプロセス代数のアクションとしての記号の列だけを扱っている． 4.1 節で述べたように記号の意味を考慮してはいない． FMEA

や FTA のようにタスクの意味を考慮しながら解析することはしていない．作業手順の耐性を向上するにはタスクの意味を理解し他の手法も併用して利用するべきである．また，反例の解析ではタスクの意味を十分理解したうえで解析する必要がある．

本稿は作業手順についてのべている．しかし，システムの操作に人為的欠陥を埋め込んだときのシステムの耐性を検証することにも利用できると考えている．

参 考 文 献

[1] Avižienis, A., Laprie, J.-C., Randell, B., and Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing, *IEEE Transactions on Dependable and Secure Computing*, Vol. 1(2004), pp. 11–33.

[2] Bernardeschi, C., Fantechi, A., and Gnesi, S.: Model checking fault tolerant systems, *Software Testing, Verification and Reliability*, Vol. 12, No. 4(2002), pp. 251–275.

[3] Gnesi, S., Lenzini, G., and Martinelli, F.: Logical Specification and Analysis of Fault Tolerant Systems through Partial Model Checking, *Proceedings of the International Workshop on Software Verification and Validation (SVV 2003), Mumbai, India*, Etalle, S., Mukhopadhyay, S., and Roychoudhury, A.(eds.), Electronic Notes in Theoretical Computer Science, Vol. 118, Amsterdam, Elsevier, December 2003, pp. 57–70.

[4] Krishnan, P.: A semantic characterisation for faults in replicated systems, *Theoretical Computer Science*, Vol. 128, No. 1-2(1994), pp. 159 – 177.

[5] Milner, R.: *Communication and concurrency*, Printice-Hall, 1989.

[6] 永藤直行: GDB とシステムモデルを用いたソースコード検証器の開発, 情報処理学会論文誌 プログラミング, Vol. 47, No. SIG 11(PRO30)(2006), pp. 1–12.

[7] PRESYSTEMS Inc.: A Model Checker: nhk. `http://www4.ocn.ne.jp/~presys/index_en.html`.

[8] Reason, J.: *Human Error*, Cambridge University Press, 1990.

[9] Rushby, J.: Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems, *In Second International Symposium on Formal Techniques in Real Time and Fault Tolerant Systems*, Springer Verlag, 1991, pp. 237–257.

[10] Swain, A. D. and Guttmann, H. E.: HANDBOOK OF HUMAN RELIABILITY ANALYSIS WITH EMPHASIS ON NUCLEAR POWER PLANT APPLICATIONS, Draft Report NUREG/CR-1278, U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research, Washington, DC, May 1982.