ソフトウェア工学 117－4
（1997．11．21）

動的再構成システムの仕様化と検証

板橋吾一，加納稔久，高橋薫，加藤靖
仙台電波工業高等専門学校

〒989-31 仙台市青葉区上愛子字北原1番地
E-Mail : {s3105, a621}@ccedu.sendai-ct.ac.jp
kaoru@cc.sendai-ct.ac.jp
kato@info.sendai-ct.ac.jp
TEL : 022-392-4761
FAX : 022-392-4459

要旨 分散並行システムにおいては，リンクを経由してシステム構成要素間で通信が行われ，所望の分散アプリケーションが達成される．リンクが静的ではなく動的に構成・確立されるのであれば，システムは動的再構成システムとして複雑化を呈することになる．本稿では，システム構成要素を有限状態機械の概念でモデル化し，それらが動的にリンクを確立・解消しながら並行実行するような動的再構成システム仕様の挙動検証を可能とする方法について検討し，検証システムの実装を行う．

Specification and Validation of a Dynamically Reconfigurable System

Goichi ITABASHI, Toshihisa KANO, Kaoru TAKAHASHI and Yasushi KATO
Sendai National College of Technology

Kitahara 1, Kamiayashi, Aoba-ku, Sendai, 989-31, JAPAN
E-Mail : {s3105, a621}@ccedu.sendai-ct.ac.jp
kaoru@cc.sendai-ct.ac.jp
kato@info.sendai-ct.ac.jp
TEL : +81-22-392-4761
FAX : +81-22-392-4459

Abstract In a distributed concurrent system such as a computer communication network, the system components communicate with each other via communication links in order to accomplish a desired distributed application. If the links are dynamically established among the components, the system configuration as well as its behavior becomes complex. In this paper, we give formal specification of such a dynamically reconfigurable system in which the components are modeled by communicating finite state machines executed concurrently with the communication links which are dynamically established and released. We also present an algorithm to validate the safety and link-related properties in the specified behavior. Finally, we design and implement a validator that enables validation of the given specification.

1 まえがき

コンピュータネットワークなどのシステム，特に分散並行システムにおいては，システム構成要素間で，あるプロトコルに従った相互作用を経由し所望の分散アプリケーションが実行・達成される．リンクが固定的であればシステムの構造は変化しない．しかしながら，リンクが動的に確立・解消されるのであればシステムの構造も動的に変化する．結果的に，システム全体の挙動は動的再構成システム (dynamically reconfigurable system) として複雑化を呈することになる．

本稿では，システム構成要素間を通信型有限状態機械 (*CFSM: Communicating Finite State Machine*) の概念でモデル化し，それらが動的にリンクを確立・解消しながら並行実行するような動的再構成システムの形式仕様を与える．また，システム仕様の挙動を解析することによって，設計者に仕様内容の検証を可能とさせる挙動検証法を与え，インターネットブラウザを用いた検証システムの設計と実装を行う．

2 システムモデル

動的再構成システムを，互いにリンクを形成し通信しあういくつかの有限状態機械の集まりとしてモデル化する．

定義 1 動的再構成システム Sys を

$$Sys = \langle CFSM_1, \cdots, CFSM_k, \cdots, CFSM_n \rangle$$

で定義する．ここで，$CFSM_k (1 \leq k \leq n)$ は通信型有限状態機械であり次のように定義する：

$CFSM_k = \langle Q_k, E_k, \delta_k, q_{k0} \rangle$
- Q_k $CFSM_k$の状態の有限集合
- E_k $CFSM_k$のイベントの集合
  - $E_k = IP_k \times (\Sigma_k \cup \Delta_k \cup C_k)$
  - IP_k 作用点の集合
    - $IP_k = \{1, \cdots, n\} - \{k\}$
  - Σ_k 入力イベントの集合
  - Δ_k 出力イベントの集合
  - C_k 制御イベントの集合
- δ_k $CFSM_k$の状態遷移関数
  - $\delta_k : E_k \times Q_k \longrightarrow Q_k$
- q_{k0} $CFSM_k$の初期状態 ($q_{k0} \in Q_k$)

□

$(p, e) \in E_k$であることは，$CFSM_k$の入力または出力または制御イベント e が$CFSM_p$を対象として生起することを示す．$CFSM_k$における状態遷移はこれらのイベントの生起により可能となる．

制御イベントには *connect*(通信リンクの確立) と *disconnect* (通信リンクの切断) がある．入力イベント (メッセージ，作用の受信) または出力イベント (メッセージ，作用の送信) は一対の *CFSM* 間にリンクが確立して始めて可能になる．リンクの切断時は可能ではない．

定義 2 個々の *CFSM* に対する記法・記号を以下のように定義する．

$p?e$	作用点 p での受信イベント e ($CFSM_p$からの受信イベント e)
$p!e$	作用点 p での送信イベント e ($CFSM_p$への送信イベント e)
$p\#$	作用点 p での接続制御イベント ($CFSM_p$とのリンクの確立)
$p/$	作用点 p での切断制御イベント ($CFSM_p$とのリンクの切断)
$q \xrightarrow{e} q'$	状態 qから状態 q'へのイベント e による遷移
$q \xrightarrow{e}$	$q \xrightarrow{e} q'$なる q'が存在
$q \not\xrightarrow{e}$	$q \xrightarrow{e}$ でないとき
$q \longrightarrow$	$q \xrightarrow{e}$ なる e が存在
$q \not\longrightarrow$	$q \longrightarrow$ でないとき

□

3 システムの挙動と検証

3.1 システムの挙動

システム Sys の挙動を以下の考え方の下で特性化する：

1. システム全体の挙動開始時には，*CFSM* 間にリンクは一切確立されておらず，またそれぞれ初期状態にある．
2. 一対の *CFSM* が互いに *connect* を発して始めてリンクが確立される．リンクが確立することにより，対応する作用点同士を結びつける．このとき互いに異なる方向のチャネルが形成され，これにより双方向通信が可能である．*connect* による状態遷移は，対応する *CFSM* の *connect* による状態遷移と同期して起こる．
3. 確立されたリンクのチャネルの受信側には暗黙の *FIFO* (*First In First Out*)，完全 (*perfect*)，容量制限 (*bounded*) [1]の受信イベントキューが

[1]検証の立場から容量を制限している．

仮定される[2]．一方の $CFSM$ からの送信イベントは他方の $CFSM$ の受信キューに蓄えられる．イベントの受信は対応するキューの先頭にそのイベントが存在するときに限り可能となる．受信が起こると，そのイベントはキューから取り除かれる．なお、確立時のキュー内容は空である．

4. 確立されているリンクのチャネルは，一方の $CFSM$ が *disconnect* を発して切断される．このチャネルは，発側から相手側方向のチャネルである．切断された側のチャネルの送信イベントは以降無効となる．受信イベントキューは変化せず受信は可能である．逆方向のチャネルは，影響されない．両方向のチャネルが切断されて初めてリンクが切断される．

5. システム全体は個々の $CFSM$ の状態遷移によって挙動変化を起こしていく．

以上から，(a) 個々の $CFSM$ の状態，(b) $CFSM$ 対間に確立されたリンク端に仮定されたイベントキューの状態 (内容)，として表現されるシステム状態を考えることができ，システム Sys の挙動をそのようなシステム状態の遷移の系列として定義できる．このシステム状態遷移系列は一つのグラフ構造を形成し，このグラフ (システム状態グラフ) が Sys の挙動全体を表現する．

定義 3

$$Sys = \langle CFSM_1, \cdots, CFSM_k, \cdots, CFSM_n \rangle$$

のシステム状態 s を以下のように定義する：

$$s = \langle (q_1, \cdots, q_k, \cdots, q_n), \\ (c_{1,2}, \cdots, c_{i,j}, \cdots, c_{n,n-1}), \\ (l_{1,2}, \cdots, l_{i,j}, \cdots, l_{n,n-1}) \rangle$$

ここで，q_k $(1 \leq k \leq n)$ は $CFSM_k$ の状態である．$c_{i,j}$ $(1 \leq i \leq n\ ,\ 1 \leq j \leq n\ ,\ i \neq j)$ は $CFSM_i$ と $CFSM_j$ 間のリンクの $CFSM_i$ から $CFSM_j$ への方向のチャネルの $CFSM_j$ 側のイベントキューの状態 (内容) である．$l_{i,j}$ $(1 \leq i \leq n\ ,\ 1 \leq j \leq n\ ,\ i \neq j)$ は $CFSM_i$ と $CFSM_j$ 間のチャネルの状態を表し，*1* は確立状態，*0* は切断状態を表す． □

定義 4 システム状態グラフ G は次の 4 項組である：

$$G = \langle\ S, E, \delta, s_0\ \rangle$$

ここで，S はシステム状態の集合，E はイベントの集合，δ $(\delta : S \times E \rightarrow S)$ はシステム状態の遷移関数，s_0 は初期システム状態である．なお，E は，ある $CFSM$ から別の $CFSM$ への出力イベント，ある $CFSM$ の別の $CFSM$ からの入力イベント，$CFSM$ 間の接続制御イベント，$CFSM$ 間の切断制御イベントから成る． □

定義 5 システム状態に対する記法・記号を次のように定義する．

ϵ	空系列 (空のキュー内容を表す)
$\mathcal{C}$	チャネルの容量
$\mathcal{W}_i$	$CFSM_i$ の許容リンク本数
$(i,j)?e$	$CFSM_j$ の $CFSM_i$ からの入力イベント e
$(i,j)!e$	$CFSM_i$ から $CFSM_j$ への出力イベント e
$(i,j)\#$	$CFSM_i$ と $CFSM_j$ の間の接続制御イベント (リンクの確立)
$(i,j)/$	$CFSM_i$ から $CFSM_j$ への切断制御イベント (リンクの切断)
$s \xrightarrow{e} s'$	システム状態 s からシステム状態 s' へのイベント e による遷移
$s \xrightarrow{e}$	$s \xrightarrow{e} s'$ なる s' が存在
$s \not\xrightarrow{e}$	$s \xrightarrow{e}$ でないとき
$s \longrightarrow$	$s \xrightarrow{e}$ なる e が存在
$s \not\longrightarrow$	$s \longrightarrow$ でないとき
$append(e, c_{i,j})$	イベント e が最後尾に加えられたイベントキュー内容 $c_{i,j}$
$top(c_{i,j})$	イベントキュー内容 $c_{i,j}$ の先頭のイベント
$remove(c_{i,j})$	先頭のイベントが取り除かれたイベントキュー内容 $c_{i,j}$
$length(c_{i,j})$	イベントキュー内容 $c_{i,j}$ のイベント数
$state_i(s)$	システム状態 s における $CFSM_i$ の状態
$channel_{i,j}(s)$	システム状態 s におけるイベントキュー内容 $c_{i,j}$
$link_{i,j}(s)$	システム状態 s におけるチャネルの状態 $l_{i,j}$
$\#links_i(s)$	システム状態 s における $CFSM_i$ の接続リンク本数

□

定義 6

$$Sys = \langle CFSM_1, \cdots, CFSM_k, \cdots, CFSM_n \rangle$$

[2] つまり，通信は非同期である．

を動的再構成システムとしたとき，対応するシステム状態グラフ

$$G = \langle S, E, \delta, s_0 \rangle$$

を以下の規則の適用により推論される最小のものとして定義する：

(初期システム状態)

$s_0 = \langle (q_{10}, \cdots, q_{i0}, \cdots, q_{n0}), (\epsilon, , \cdots, \epsilon, \cdots, \epsilon), (0, \cdots, 0, \cdots, 0) \rangle \in S.$

(リンク確立)

$s = \langle (\cdots, q_i, \cdots, q_j, \cdots), (\cdots, c_{i,j}, \cdots, c_{j,i}, \cdots), (\cdots, l_{i,j}, \cdots, l_{j,i}, \cdots) \rangle \in S$,

$l_{i,j} = 0$, $l_{j,i} = 0$, $q_i \xrightarrow{j\#} q_i'$, $q_j \xrightarrow{i\#} q_j'$

ならば

$s' = \langle (\cdots, q_i', \cdots, q_j', \cdots), (\cdots, \epsilon, \cdots, \epsilon, \cdots), (\cdots, 1, \cdots, 1, \cdots) \rangle \in S$,

$s \xrightarrow{(i,j)\#} s'$.

(イベント送信)

$s = \langle (\cdots, q_i, \cdots), (\cdots, c_{i,j}, \cdots), (\cdots, l_{i,j}, \cdots) \rangle \in S$,

$l_{i,j} = 1$, $q_i \xrightarrow{j!e} q_i'$

ならば

$s' = \langle (\cdots, q_i', \cdots), (\cdots, append(e, c_{i,j}), \cdots), (\cdots, l_{i,j}, \cdots) \rangle \in S$,

$s \xrightarrow{(i,j)!e} s'$.

(イベント受信)

$s = \langle (\cdots, q_i, \cdots), (\cdots, c_{j,i}, \cdots), (\cdots, l_{j,i}, \cdots) \rangle \in S$

$top(c_{j,i}) = e$, $q_i \xrightarrow{j?e} q_i'$

ならば

$s' = \langle (\cdots, q_i', \cdots), (\cdots, remove(c_{j,i}), \cdots), (\cdots, l_{j,i}, \cdots) \rangle \in S$,

$s \xrightarrow{(j,i)?e} s'$.

(リンク切断)

$s = \langle (\cdots, q_i, \cdots, q_j, \cdots), (\cdots, c_{i,j}, \cdots, c_{j,i}, \cdots), (\cdots, l_{i,j}, \cdots, l_{j,i}, \cdots) \rangle \in S$,

$l_{i,j} = 1$, $q_i \xrightarrow{j/} q_i'$

ならば

$s' = \langle (\cdots, q_i', \cdots, q_j, \cdots), (\cdots, c_{i,j}, \cdots, c_{j,i}, \cdots), (\cdots, 0, \cdots, l_{j,i}) \rangle \in S$,

$s \xrightarrow{(i,j)/} s'$.

確立において $(i,j)\#$ と $(j,i)\#$ は同じものを指すが，どちらか一方を用いることにする． □

性質 1 一般に，システム Sys に対し，対応するシステム状態グラフ G は非有限である． □

ある容量 C に対し，C を超えるイベント送信を可能としないよう上記の規則を変えて得られるシステム状態グラフを**容量制限システム状態グラフ**と呼ぶとき，次の性質が成り立つ．

性質 2 任意のシステム Sys に対し，対応する容量制限システム状態グラフ G は有限である． □

3.2 挙動の検証

3.2.1 安全性

システム Sys に対応するシステム状態グラフ $G = \langle S, E, \delta, s_0 \rangle$ を用いて検証しようとすれば，G の生成過程で以下の状態を検証できる．

定義 7 $s \not\rightarrow$, $q_i \not\rightarrow (1 \leq \forall i \leq n)$, $c_{i,j} = \epsilon$ $(1 \leq \forall i, j \leq n, i \neq j)$, $l_{i,j} = 0$ $(1 \leq \forall i, j \leq n, i \neq j)$ であるようなシステム状態 $s = \langle (\cdots, q_i, \cdots), (\cdots, c_{i,j}, \cdots), (\cdots, l_{i,j}, \cdots) \rangle \in S$ を**終了状態**と呼ぶ． □

定義 8 $s \not\rightarrow$ であり，終了状態でないシステム状態 $s \in S$ を**デッドロック状態**と呼ぶ． □

定義 9 $C < length(c_{i,j})$ なるシステム状態 $s = \langle (q_1, \cdots, q_n), (\cdots, c_{i,j}, \cdots), (\cdots) \rangle \in S$ を**チャネルオーバフロー状態**と呼ぶ． □

定義 10 $s \not\rightarrow \wedge\ c_{i,j} \neq \epsilon \wedge q_j \overset{i?top(c_{i,j})}{\not\rightarrow}$ であるようなシステム状態 $s = \langle (\cdots, q_j, \cdots), (\cdots, c_{i,j}, \cdots), (\cdots) \rangle \in S$ を**未指定受信状態**と呼ぶ． □

性質 3 未指定受信状態はデッドロック状態である． □

定義 11 システム状態グラフ G がデッドロック状態，チャネルオーバフロー状態を含まないならば，システム Sys は**安全 (Safe)** であるという． □

3.2.2 リンク関連エラー

システム状態グラフ $G = \langle S, E, \delta, s_0 \rangle$ を用いながら安全性の検証を行なうと共に，以下に示すリンク関連のエラーも検出できる．

定義 12 $l_{i,j} = 0 \ \wedge \ q_i \xrightarrow{j!e}$ なるシステム状態 $s = \langle (\cdots, q_i, \cdots), (\cdots), (\cdots, l_{i,j}, \cdots) \rangle \in S$ の q_i を**未リンク送信状態**と呼ぶ． □

定義 13 リンクが 1 回も確立されることのない $CFSM$ 対を**未リンク $CFSM$ 対**と呼ぶ． □

定義 14 ($l_{i,j} = 1$ 又は $l_{j,i} = 1$) $\wedge$ ($q_i \xrightarrow{j\#}$ 又は $q_j \xrightarrow{i\#}$) なるシステム状態 $s = \langle(\cdots, q_i, \cdots, q_j, \cdots), (\cdots), (\cdots, l_{i,j}, \cdots, l_{j,i}, \cdots)\rangle \in S$ の q_i又は q_jを**重複確立トライ状態**と呼ぶ. □

定義 15 $l_{i,j} = 0 \wedge q_i \xrightarrow{j/}$ なるシステム状態 $s = \langle(\cdots, q_i, \cdots), (\cdots), (\cdots, l_{i,j}, \cdots)\rangle \in S$ の q_iを**重複切断トライ状態**と呼ぶ. □

定義 16 $\mathcal{W}_i < \#links_i(s)$ なるシステム状態 $s = \langle(\cdots, q_i, \cdots), (\cdots), (\cdots)\rangle \in S$ の q_iを**リンク過多状態**と呼ぶ. □

3.2.3 検証アルゴリズム

ここでは, システム状態グラフの生成を基本とした検証法を与える.

[検証アルゴリズム]
入力は 1. システム仕様

$$Sys = \langle CFSM_1, \cdots, CFSM_k, \cdots, CFSM_n\rangle$$

但し $CFSM_k = \langle Q_k, E_k, \delta_k, q_{k0}\rangle$
2. チャネル容量 $\mathcal{C}(\mathcal{C} \geq 1)$
3. 許容リンク本数 $\mathcal{W}_i(\mathcal{W}_i \geq 1(1 \leq i \leq n))$

```
begin
  initialize          (* 初期化 *)
  S_p := {s_0}
  (* 但し, s_0 = ⟨(q_10, ···, q_n0), (ε, ···, ε), (0, ···, 0)⟩ *)
  while (S_p ≠ ∅) do
    begin
      S_pの任意の要素を s とし以下の手続きの実行
        normal(s) (* システム状態グラフの生成 *)
        unsafe(s) (* 安全性の判定 *)
        link(s)   (* リンク関連エラーの判定 *)
        S_p := S_p − {s} (* s の処理終了 *)
        S := S ∪ {s}  (* s をシステム状態集合へ*)
    end
  decision1           (* 安全性に対する結果判定 *)
  decision2           (* リンクに対する結果判定 *)
end
```

[手続き]
以下は, 検証アルゴリズム内の手続きである.

```
procedure initialize
begin
  S := ∅     (* システム状態の集合 *)
  S_p := ∅   (* 処理システム状態集合 *)
  S_p := ∅   (* デッドロック状態集合 *)
  S_u := ∅   (* 未指定受信状態集合 *)
  S_co := ∅  (* チャネルオーバフロー状態集合 *)
  δ := ∅     (* システム状態遷移関数 *)
  L := {(i,j)|1 ≤ i,j ≤ n, i ≠ j}
             (* 未リンク CFSM対集合 *)
  L_e := ∅   (* 重複確立トライ状態集合 *)
  L_d := ∅   (* 重複切断トライ状態集合 *)
  L_s := ∅   (* 未リンク送信状態集合 *)
  L_f := ∅   (* リンク過多状態集合 *)
end

procedure normal(s)
begin
  以下の手続きを摘要し s の全ての次システム
  状態と状態遷移を求める :
  (* リンク確立 *)
    if (∃CFSM_i, ∃CFSM_j)(∃s')(s --(i,j)#--> s') then
      begin
        L := L − {(i,j),(j,i)}
        δ := δ ∪ {s --(i,j)#--> s'}
        if (s' ∉ S ∪ S_p) then S_p := S_p ∪ {s'}
      end
  (* イベント送信 *)
    if (∃CFSM_i, ∃CFSM_j)
       (∃e)(∃s')(s --(i,j)!e--> s') then
      begin
        δ := δ ∪ {s --(i,j)!e--> s'}
        if (s' ∉ S ∪ S_p) then S_p := S_p ∪ {s'}
      end
  (* イベント受信 *)
    if (∃CFSM_i, ∃CFSM_j)
       (∃e)(∃s')(s --(j,i)?e--> s') then
      begin
        δ := δ ∪ {s --(j,i)?e--> s'}
        if (s' ∉ S ∪ S_p) then S_p := S_p ∪ {s'}
      end
  (* リンク切断 *)
    if (∃CFSM_i, ∃CFSM_j)(∃s')(s --(i,j)/--> s') then
      begin
        δ := δ ∪ {s --(i,j)/--> s'}
```

```
      if (s' ∉ S ∪ S_p) then S_p := S_p ∪ {s'}
    end
end

procedure unsafe(s)
begin
  以下の手続きを摘要し s に関する安全性の欠如を
  検証する：
  (* デッドロック状態 *)
    if (s ↛) ∧
        ¬ ((1 ≤ ∀i, ∀j ≤ n, i ≠ j) ((state_i(s) ↛) ∧
            (c_{i,j} = ε) ∧ (l_{i,j} = 0)))
    then S_d := S_d ∪ {s}
  (* 未指定受信状態 *)
    if (s ↛) ∧
        ((∃CFSM_i, ∃CFSM_j)(channel_{i,j}(s) ≠ ε ∧
            state_j(s) --i?top(channel_{i,j}(s))--↛))
    then S_u := S_u ∪ {s}
  (* チャネルオーバフロー状態 *)
    if (∃CFSM_i, ∃CFSM_j)(∃e)(∃s')
        (s --(i,j)!e--> s' ∧ C < length(channel_{i,j}(s')))
    then
      begin
        S_co := S_co ∪ {s'}
        S_p := S_p − {s'}
      end
end

procedure link(s)
begin
  (* 未リンク送信状態 *)
    if (∃CFSM_i, ∃CFSM_j)
        (link_{i,j}(s) = 0 ∧ (∃e)(state_i(s) --j!e-->))
    then L_s := L_s ∪ {(i, state_i(s))}
  (* 重複確立トライ状態 *)
    if (∃CFSM_i, ∃CFSM_j)
        ((link_{i,j}(s) = 1 ∨ link_{j,i}(s) = 1) ∧
            (state_i(s) --j#--> ∨ state_j(s) --i#-->))
    then
      begin
        if (state_i(s) --j#-->)
        then L_e := L_e ∪ {(i, state_i(s))}
        if (state_j(s) --i#-->)
        then L_e := L_e ∪ {(j, state_j(s))}
      end
  (* 重複切断トライ状態 *)
    if (∃CFSM_i, ∃CFSM_j)
        (link_{i,j}(s) = 0 ∧ (state_i(s) --j/-->))
    then L_d := L_d ∪ {(i, state_i(s))}
  (* リンク過多状態 *)
    if (∃CFSM_i, ∃CFSM_j)(∃s')
        (s --(i,j)#--> s' ∧ (W_i < #links_i(s') ∨
            W_j < #links_j(s')))
    then
      begin
        if (W_i < #links_i(s'))
        then L_f := L_f ∪ {(i, state_i(s'))}
        if (W_j < #links_j(s'))
        then L_f := L_f ∪ {(j, state_j(s'))}
        S_p := S_p − {s'}
      end
end

procedure decision1
begin
  if (S_d ≠ ∅) then システム Sys は
          デッドロック状態を含む
  if (S_u ≠ ∅) then システム Sys は
          未指定受信状態を含む
  if (S_co ≠ ∅) then システム Sys は
          チャネルオーバフロー状態を含む
  if (S_d = S_co = ∅) then システム Sys は
          安全である
end

procedure decision2
begin
  if (L ≠ ∅) then システム Sys は
          未リンク CFSM対を含む
  if (L_s ≠ ∅) then システム Sys は
          未リンク送信状態を含む
  if (L_e ≠ ∅) then システム Sys は
          重複確立トライ状態を含む
  if (L_d ≠ ∅) then システム Sys は
          重複切断トライ状態を含む
  if (L_f ≠ ∅) then システム Sys は
          リンク過多状態を含む
end
```

[アルゴリズム終了]

性質 4 チャネル容量を制限しているため，システム状態グラフの生成は無限には行われず，このアルゴリズムは有限ステップで停止する．□

4 挙動検証システム

4.1 設計支援

動的再構成システムの設計において，(i) 仕様作成の支援，(ii) 検証の支援，(iii) 仕様化されたシステムの (半) 自動実装の支援は重要である．特に (ii) 検証の支援においては，仕様化されたシステムが所望の機能要件を満足しているか，また，機能的な矛盾を含んでいないかなどの支援が必要となる．このため，システムの非デッドロック性などの安全性の自動検証機能や，ユーザ主導のシステム挙動シミュレーション機能を提供しなければならない．

本稿では，安全性とリンク関連エラーの検証を行なう挙動検証システム (*Mild*) の設計と実現について述べる．

4.2 *Mild*：挙動検証システム

4.2.1 機能要件

以下に，挙動検証システムの具備すべき機能要件を整理する．

(a) 仕様入力 取り扱う動的再構成システムの仕様を，ユーザに容易に指定可能とさせるインタフェースの提供と，ユーザ自身が仕様を入力できる機能を提供する．また，仕様表示や検証に適した形の内部表現化を行う．

(b) 仕様表示 ユーザの要求に応じて，入力仕様のグラフ表示，あるいはテキスト形式での表示を行う．これには，*CFSM* それぞれの具体的定義の状態遷移図表示を含む．視認性の良い表示のため，表示ウインドウの縦横スクロール機能を具備する．

(c) 挙動検証 挙動検証システムのメイン機能である．基本的に，検証は上述したシステム状態グラフを生成することに対応する．その際，以下の検証をする：

- 非デッドロック性などの安全性の検証
- リンクに関連するエラーの検証
- 安全性，リンク関連エラーの両方を検証

また，以下の機能が有用である．

- 生成されたシステム状態グラフのグラフィカル表示
- 検証結果のテキスト表示とグラフィカル表示

4.2.2 実装

上記機能要件の下，本挙動検証システムを Web サーバ上に JAVA applet を用いて実装を行なっている．よってこの *Mild* は，Netscape Navigator や Microsoft Internet Explorer などのインターネットブラウザを用いることにより，ネットワークワイドで実行できる．

以下では簡単なモバイルシステム [2] を例にとり，本検証システムの実装内容と動作について記述する．

この例では，システム仕様は 2 つの *CFSM*(*car*,*base*) から成り，それぞれ自動車電話 (*car*) と基地局 (*base*) を表している．始め，両者は接続を行なおうとし，接続後，両者は信号を交わしていく．

仕様の指定・入力は，**図 1** のメイン画面によって行なえる．この挙動検証システムは，ブラウザの中に埋め込まれており，ユーザは検証するシステム仕様を選択 (この例では "PHS" を選択) し，*Display* ボタンを押すとシステム仕様の定義内容が表示され，*Validate* ボタンを押すと検証のセットアップが開始されるようになっている．*Edit* ボタンは，ユーザが自身がシステム仕様を入力できるエディタが表示されるようにしている．

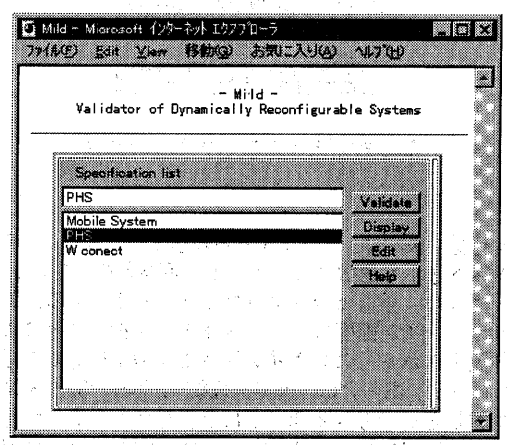

図 1: *Mild* メイン画面

図 2 は *car* の具体的定義内容を状態遷移図形式で表示している．図中，円で囲まれた数字は *CFSM* の状態番号を，枝は状態間遷移を，枝のラベルは遷移のトリガとなるイベントを示している．またシステ

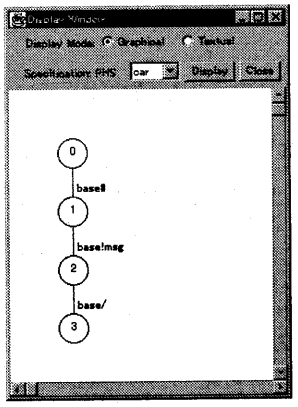

図 2: *CFSM* 表示ウインドウ (グラフィカル) - *car* -

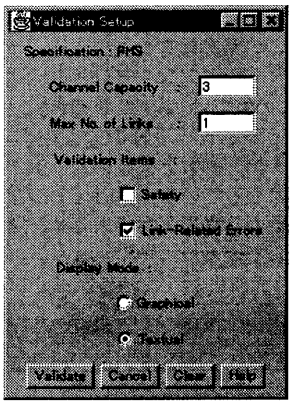

図 3: 検証セットアップウインドウ

ム仕様の表示は，グラフィカル表示だけではなく，テキスト表示も選択することにより可能となっている．

図 3 は，検証セットアップウインドウである．これは，メイン画面の *Validate* ボタンを押すことにより表示される．このウインドウでは検証に必要な項目の入力，あるいは，選択を行なう．ユーザはチャネル容量と許容リンク数 (これはまだプロトタイプ版であり許容リンク数は通常個々の *CFSM* によって異なるが，全ての *CFSM* が同じ許容数としている) を入力し，検証項目と表示モードの選択をすることで始めて検証が可能となる．図中では，チャネル容量 3，許容リンク数 1，検証項目はリンク関連エラー，表示モードはテキスト表示としている．

図 4 は，検証の結果表示ウインドウである．検証

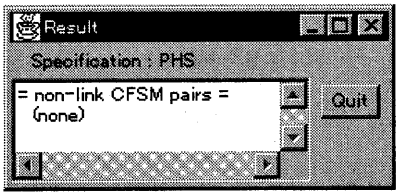

図 4: 結果表示ウインドウ

は，セットアップ画面で入力されたデータを参照して行なう．この "PHS" の例では検証の結果，未リンク *CFSM* 対は存在しなかったことを意味している．この結果表示ウインドウのグラフィカル表示機能，また，検証本体はなお開発中である．

5 むすび

本稿では，動的にその構造を変更可能なシステムのモデルを与え，挙動の検証を可能とする方法を述べ，それに対する挙動検証システムの作成を行った．この挙動検証システムは一部なお開発中であり，最終的な完成，広範な適用，および改良については今後の課題の一つである．

参考文献

[1] K. J. Turner, *Using Formal Description Techniques*, JOHN WILEY & SONS, 1993.

[2] R. Milner, *The Polyadic π-Calculus: a Tutorial*, Computer Science Dept., Univ. of Edinburgh, 1991.

[3] 佐藤明日香，関利美，渡部智広，高橋薫："動的再構成システムの仕様化とそのシミュレータ"，信学技報 SSE96-72, 1996.