形式仕様のやわらかい獲得支援環境の構築

6L－1

福沢 尚司　臼井 伸幸　宋 国煥　富樫 敦　白鳥 則郎
東北大学電気通信研究所 / 情報科学研究科

1　はじめに

システムをあいまい性なく，完全に記述するものとして形式仕様がある．形式仕様の１つであり，また，他の形式仕様記述言語であるSDL, Estelle, LOTOS等の基礎ともなっているものが状態遷移システムである．状態遷移システムはその目的から一般にはユーザがそれを直接記述するのは困難である．

そこで本稿ではユーザの作成した要求仕様を形式仕様に変換する機能を中心に形式仕様のやわらかい(Flexible)獲得支援環境を提唱し，その一部を実現したシステムを試作する．

2　準備

2.1　要求仕様と形式仕様

本稿で扱う要求仕様，形式仕様とはそれぞれ以下のようなものである．

要求仕様：$\mathcal{R} =< R, \gamma_0 >$
R は機能要求の集合
γ_0 は初期条件
(機能要求：$\rho =< id, a, f_{in}, o, f_{out} >$
id は機能名
a は入力
f_{in} は実行前条件
o は出力
f_{out} は実行後条件
機能要求は $\rho =< a, f_{in}, f_{out} >$ のように省略した形で考えることがあり，その場合 $\rho : f_{in} \stackrel{a}{\Rightarrow} f_{out}$ のように表記する.)
例：$\mathcal{R}_1 =< \rho_1 : aa \stackrel{a}{\Rightarrow} \neg aa, \rho_2 : bb \stackrel{b}{\Rightarrow} aa, aa \wedge bb >$
機能の独立性を生かして，各機能毎に追加，修正，消去が容易である．

形式仕様：$M =< Q, \Sigma, \rightarrow, q_0 >$
Q は状態の集合
Σ は入力の集合
$\rightarrow$ は遷移関係で $\rightarrow \subset Q \times \Sigma \times Q$
q_0 は初期状態
例：

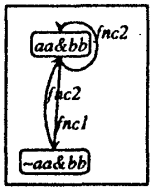

図１形式仕様（状態遷移システム）の例

2.2　要求仕様から形式仕様への変換アルゴリズム

詳細は文献[1]に依り，概略のみを述べる．まず初期条件の成り立っている状態を作る．これを初期状態とする．ある状態である機能の f_{in} が成り立っているとき，その機能の f_{out} と，それとは独立なその状態で成り立っている条件が成り立つような状態への遷移を作る．それを遷移が作れなくなるまで行なう．

2.3　状態に対する制限

誤った状態が作られるのを防ぐためにあらかじめ状態に対して制限を付けることができる．制限にはいくつかの方式があるが[1]，そのうちの１つである *inhibit constraints* は，$\neg(l_1 \wedge ... \wedge l_n)$ のような形をしていて同時に成り立ってはならない条件を書くものである．

3　試作システム

3.1　システム構成

図にシステムの構成概略を示す．

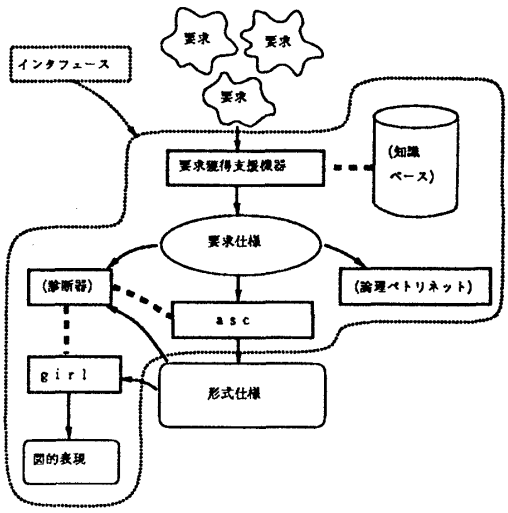

図２試作システム概略図

A Flexible Support Environment for Formal Specifications
Shoji Fukuzawa, Nobuyuki Usui, Kukhwan Song, Atsushi Togashi and Norio Shiratori
{Reseach Institute of Electrical Communication, Graduate School of Information Sciences} Tohoku University

システムは要求仕様から形式仕様への変換を行なう asc, 要求仕様の獲得や修正等を支援し, プログラム間のデータやファイルとの橋渡しをするインタフェース, 状態遷移システムのグラフィカルな表示を行なう girl から成り立っている. それ以外にも過去の蓄積から目的の仕様に類似したものを利用できるデータベースや, 要求仕様を論理ペトリネットの形に変換し, その性質を利用して検証を行なったりそれをそのまま形式仕様とすることを目的とする機能 [2], また girl の機能と連動して形式仕様の正当性の診断を行なえるシステムなども考えられている.

3.2 変換プログラム asc

asc はプログラム言語 Prolog で書かれた要求仕様から形式仕様への変換器である.Prolog を用いたのはこの言語がリストとそのメンバを処理するのに適していたためで,asc では入力として Prolog のリスト形式で書かれたファイルを用いる. 入力ファイルを受けると, アルゴリズム [1] に従って処理を行ない, girl の入力となる lts 形式のファイルを出力する.

3.3 インタフェース

ユーザが直接 Prolog のリストを作成せねばならない不便を解消するために, また修正, 変更を容易にし, データのやりとりをスムーズに行なうためのもの.

3.4 girl

lts 形式で書かれたファイルを入力として X-Window 上に状態遷移システムをグラフィカルに表示するツール. PostScript 形式での出力も可能.

4 実行例

簡単な CATV の形式仕様を得る.

要求仕様

$initial\_condition : \tilde{}mute, \tilde{}force, \tilde{}pw, \tilde{}bz$
$function :$
$pw\_off : pw, \tilde{}force, \tilde{}bz == pwkey => \tilde{}pw$
$pw\_on : \tilde{}pw == pwkey => \tilde{}mute, pw$
$ch\_up : pw, \tilde{}force, \tilde{}bz$
$== chupkey/[printCH] => pw$
$ch\_dw : pw, \tilde{}force, \tilde{}bz$
$== chdwkey/[printCH] => pw$
$ch\_ch : pw, \tilde{}force, \tilde{}bz$
$== tenkey/[printCH] => pw$
$mute\_on : \tilde{}mute, \tilde{}bz, \tilde{}force, pw$
$== mutekey/[printMUTE] => mute$
$mute\_off : mute, \tilde{}bz, \tilde{}force, pw$
$== mutekey/[printVOL] => \tilde{}mute$
$force\_tune : \tilde{}pw$
$== ftune/[printCH] => force, pw$
$force\_tune : pw == ftune/[printCH] => force$
$force\_cancel : force, \tilde{}bz, pw$
$== pwkey/[printCH] => \tilde{}force$
$buzzer\_on : \tilde{}bz == buzzer/[buzzeron] => bz$
$buzzer\_off : bz == anykey/[buzzeroff] => \tilde{}bz$
$inhibit\_constraints :$
$\tilde{}(\tilde{}pw, force)$
$\tilde{}(\tilde{}pw, bz)$
$\tilde{}(\tilde{}pw, mute)$

形式仕様

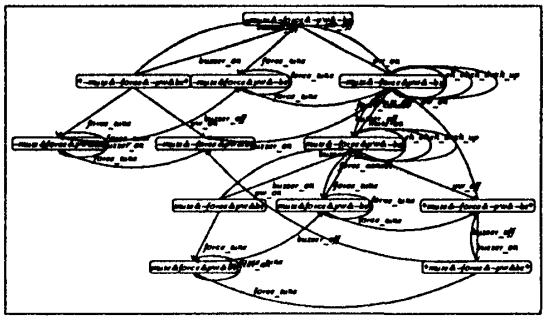

図 3 CATV の形式仕様

5 まとめ

形式仕様の獲得のための支援環境を示し, システムを試作した. 今後の課題としては, より Flexible な開発が行なえるよう改良を加えること, 先にあげたようなペトリネットやデータベースへの対応や診断器の開発, 果てはグラフィカルなエディティングを可能にするなど幅広い展開が考えられている.

参考文献

[1] Atsushi Togashi,Nobuyuki Usui,Kukhwan Song and Norio Shiratori,"A Derivation of System Specifications based on a Partial Logical Petri Net"(to be submitted).

[2] Song,K.,Togashi,A.,Shiratori,N.,An Automatic Generation of Formal Specifications and its Verification using Lgical Petri Net,Tech.Rep. of IEICE,CST-94-29,1994.