4T-4

通信ソフトウェア仕様の意味的検証法

西片 聡　　平原 富士則

ＮＴＴ通信網総合研究所

１．はじめに

通信サービスプログラムなどのシステム制御ソフトウェアの生産性向上を図るため、さまざまな形式的仕様記述言語が提案され、それらの言語で記述されたプログラム仕様を自動的に検証する方法が研究・開発されている[1-5]。

本稿では、状態遷移ベースの仕様記述言語で記述された通信サービスプログラムの仕様を意味的に検証する一方法を述べる。

２．意味的検証の意義

２．１　状態遷移ベースの仕様記述

通信システムなどを制御するソフトウェアの生産工程の効率化を図るため、システムの動作の仕様（プログラム仕様）を形式的に記述する仕様記述言語が提案されている。

仕様記述言語なかには、対象システムを状態遷移機械でモデル化する状態遷移ベースの言語がある[1,4]。この言語によって記述した仕様はシステムの動作そのものであるので、実行プログラムへの自動変換が行い易い。その点で特に、状態遷移ベースの仕様記述言語は通信システム制御ソフトウェアの仕様記述言語として適していると考えられる。

２．２　意味的検証の必要性

プログラム開発工程における手戻りを少なくし、作業の効率化を図るためには、実行プログラムを生成する前の仕様記述の段階で誤りを検出・訂正しておくことが重要である。

プログラム仕様に含まれる可能性のある様々な誤りのうち、シンタクスエラーや動作論理上の誤りを自動的に検出する手法に関しては、既に検討が行われている[5]。

ところで状態遷移ベースの仕様記述言語で記述するプログラム仕様は、サービスに対する要求を直接的に表現したものではなく、システムの動作として間接的に表現したものである。したがってその中には先に述べた動作論理上の誤りなどとは別に、意味的な誤りが含まれる可能性がある。

本稿では、それらの誤りを検出する（意味的検証）ひとつの方法を提案する。

本稿で述べる意味的検証法が対象とする検証事項は、次のとおりである。

(a)ユーザ要求を満足しているかどうか。

例：３者会議通話を記述したいとき、同一の会議トランクに３つの端末からの通話路を接続しているかどうか。

(b)サービスの常識を満足しているかどうか。

例：着信時に受話器をあげたら呼び出し音がとまるように記述しているかどうか。

(c)システムの動作規則（ハードウェアからの要請など）を満足しているかどうか。

例：トーンを送出する前に通話路設定をしているかどうか。

３．意味的検証の方法

３．１　対象とする仕様記述言語

本節以降では、状態遷移ベースの仕様記述言語であるＳＡＬ(Service Addition Language)[4,5]で記述された通信サービスプログラム仕様を対象に、意味的検証の方法を検討する。

通信サービスプログラム仕様をＳＡＬで記述する場合、通信システムの状態遷移を表現するために状態そのものを記述するのではなく、通話路、トランクなどの各リソースに対するアクションやメッセージの送受信などをシーケンシャルに記述する。

ＳＡＬでは、それらのアクションやメッセージの送受信などのシーケンスを「イベントシーケンス」と呼ぶ。

３．２　検証方法

状態遷移ベースの仕様記述言語ＳＡＬで記述されたプログラム仕様から、2.2節で述べた意味的な誤りを検出する方法として、以下の事項を検証する方法を提案する。

(1)イベントシーケンスにより不適切なリソースの状態の組合せが生じていないか。

(2)所望のリソース状態の組合せがイベントシーケン

Verifying Communication Software Specifications Using Event Sequence Analysis

Satoshi NISHIKATA and Fujinori HIRAHARA
NTT Telecommunication Networks Laboratories

スによって導かれているか。

たとえば、2.2節(b)の例にあげたことがらを検証をするためには、「受話器が上がっており、かつ呼び出し音が鳴っている」という状態がイベントシーケンスによって生じていないかを調べればよい。

すなわち、各リソースの状態をシミュレートし、それらの間の関係を検証することにより意味的な検証を行う。

3．3　リソース状態シミュレーション

3.1節で述べたように、ＳＡＬで記述したプログラム仕様はイベントのシーケンスであり、その中にリソースの状態は明示されていない。したがって、リソースの状態に関わる検証を行うためには、イベントシーケンスからリソースの状態を得る必要がある。そのために次のシミュレート作業を行う。

①状態変数の割り付け

リソースの状態を表現するために、リソース対応に変数を割り付ける。

②イベントと状態変数の関係づけ

イベントを実行したときにどの変数の値をどのように変化させるかを設定する。

③状態変数の操作

プログラム仕様（イベントシーケンス）をトレースし、イベントごとに、対応する状態変数の値を操作していく。状態変数の値がリソースの状態を表現する。

3．4　検証手順

意味的検証の手順の概要を示す（図１）。

①検証項目の設定

通過してはならない不適切なリソース状態および通過すべき所望のリソース状態を検証項目として記述する。

②プログラム仕様のトレース

イベントシーケンスで記述されたプログラム仕様をトレースし、リソース状態をシミュレートする。

③照合

①で設定した検証項目と②のシミュレート結果を照合し、一致する部分があればそれを検出・出力する。

4．まとめ

状態遷移ベースの仕様記述言語で記述された通信サービスプログラムの仕様を、対象通信システムのリソースの状態を調べることにより意味的に検証する方法を提案した。

参考文献

[1]"Functional specification and description language (SDL)",CCITT Recommendations Z.100-Z.104 (1984)

[2]"LOTOS(formal description technique based on the temporal ordering of observational behaviour)",ISO/DIS 8807 (1987)

[3]"Estelle(formal description technique based on an extended state transition model)", ISO/DIS 9074 (1987)

[4]柴崎他,”通信ソフトウェア設計・保守環境ＳＤＥにおける仕様記述言語ＳＡＬ,”昭62信学全大1780

[5]市川他,”通信ソフトウェア設計・保守環境ＳＤＥの基本構想,”昭62信学全大1781

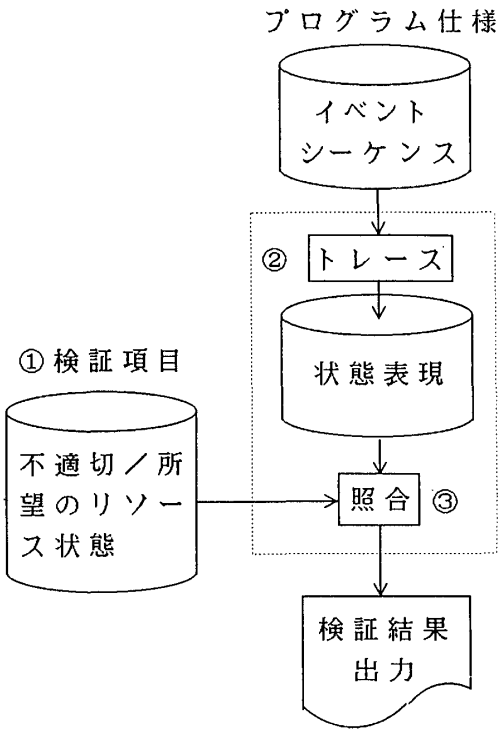

図１　意味的検証法の概要