5T-5

通信プロトコルにおける状態遷移表現の自動生成の一方式

中原 彰子　相田 仁　斉藤 忠夫　猪瀬 博
(東京大学工学部)

1．はじめに

通信プロトコルの複雑化に伴い、これをわかりやすく厳密に記述するための技術が必要となり、各種の形式記述法が提案されている。ところが、現実のプロトコル製品の実現においては、仕様は主としてイベントの系列として記述され、それをもとに担当者が、膨大な量の遷移表をじかに書き下すことが行なわれている。このような場合に計算機を用いて、状態遷移表の作成を支援することができれば、大いに有用であろう。そこで本稿では、イベントの順序による表現から状態遷移表現を得る方法について、現在検討している内容を報告する。

2．研究の概略

図1に、遷移表の作成支援システムの概略を示す。

○仕様の記述　システムの動作に関する記述には、時間論理を基本にした、イベント順序的な表現を用い、次第に具体化を行なう。割込処理や内部的な変数に関する処理は、主要部の記述とは別に記述し、詳細化の途中で追加できるようにする。

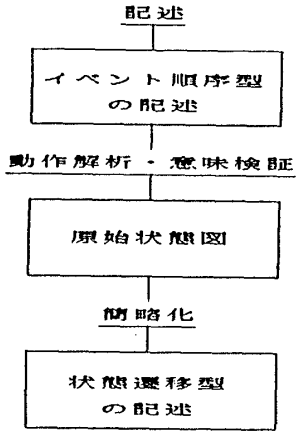

図1．作成支援システムの概略

○仕様の検証　検証には、信号の送受信処理に関する論理矛盾などをチェックする論理検証と、要求概念と仕様記述結果の相互矛盾などをチェックする意味検証の2種類がある。論理検証については、原始状態図作成後、別システムにより行なうものとし、ここでは、意味検証についてのみ考える。

○仕様の最適化　最後に、展開により得られた原始状態図を、遷移表の形に変換する。このとき、「外部入力待ち状態」を遷移表における「状態」に対応させる。こうして得られた遷移表は、インプリメントや製品検証のもとになるものである。

3．イベントの順序に基づく仕様の記述法

3．1　基本概念

「イベント」は、時間をかけて起きるできごとで、時区間の集合で表わされる。「ファクト」は、論理式で表わされ、その値が真であるようなステートの集合を表わす。「ステート」は、イベントに対応する時区間の始点・終点またはファクトの要素であり、内部状態変数のその時刻における値、そのステートで適用されるルールの集合といった情報を持つ。

3．2　イベントやファクトの詳細化

入力・出力イベント、内部状態変数に対する処理イベント、及びその値に関する論理式であるファクトのみが、プリミティブな構成要素であり、それ以外は、マクロなイベントやファクトである。これらのマクロなイベントやファクトは、最終的には、プリミティブな構成要素に置き換えなくてはならない。

3．3　内部状態変数に関する処理

内部イベントは、遷移の要因として表立って現われることはないこと、詳細化の途中で内部的な変数が設定されることがあること、などの理由により、内部状態変数に対する変更手続きとして、主要部の動作記述とは別に定義をする。

3．4　割込に関する処理

仕様の大きな部分を占めるのが、エラー発生等による割込処理に関する記述である。しかし、原始状態図上の全てのステートについて、このような割込による分岐を考えても仕方がない。そこで、割込についても、主要部の記述とは別に定義することにする。

Automatic Generation of a State-Transition Representation for Communication Protocols
Akiko NAKAHARA, Hitoshi AIDA, Tadao SAITO, Hiroshi INOSE
University of Tokyo

４．原始状態図への展開と意味検証

４．１　原始状態図への展開

イベント順序的な記述法による仕様は、基本的に、『ＡならばＢ』のルールの形をしている。この意味は、「あるステートにおいて、過去の履歴または現在の状態について論理式Ａが成立すれば、未来について論理式Ｂが成立する。」というものである。

原始状態図への展開は、初期条件から出発し、ルールを適用することによって行なわれる。ルールの適用は、一階の述語論理の考え方に従い、 Prolog と同様、マッチングやバックトラックを用いる。

４．２　ステートの同定とループの取り扱い方法

ルールの適用によって、新しくステートが設けられた場合には、新しく作られたステートについて、既存のステートとの同定を行なう。ステートの同定は、前提部(A) の成り立つルールの集合が等しいもの、即ち未来(B)に関して等しいものは、同じステートであるとみなすことによって実現する。

ステートの同定によりループが形成される。ループに対する終了条件の判断には、ループ上のステートについて漸化的に定義される内部変数の値、およびそれに関する論理式であるファクトを用いる。

(a) ルール

『再送を２回行なっても回復しない場合には失敗とする』

ルール(1) (Send ?message) <ならばただちに>
<どれか>1. [時間内に](Receive ACK)
2. [時間内に](Receive NAK)

ルール(2) (≦ ~retrans 2) <のとき> (Receive NAK)
<ならばただちに> (Send ?message)

ルール(3) (> ~retrans 2) <のとき> (Receive NAK)
<ならばただちに> ※失敗※

(b) 手順

[手順１] (初期条件) S0 から S1 にかけて、メッセージを送信した。
[手順２] ルール(1)により、ACK または NAK を受け取る。
[手順３] S3 において、ルール(2)により、メッセージの再送が生じる。
[手順４] ここで、もう１度ルール(1) が参照されるので、S4 と S1 が同定され、ループが形成される。
[手順５] ループを何度かたどると、~retrans が２となり、ルール(3) が適用される。

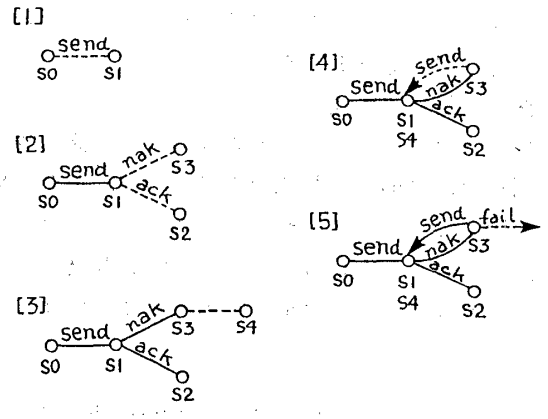

図２．原始状態図の作成手順

４．３　意味検証

意味検証は、具体的に記述された仕様が、もとのより抽象的に記述された仕様を満たしているかどうかを調べることにより行われる。意味検証に際し、具体的なレベルから抽象的なレベルを見る場合に、抽象レベルにおける論理式を、ある範囲で、より広い解釈を持った式に読み換えることを許す。これによって、具体化の場合に、例外的な事象を、論理的な不一致を引き起こすことなく、記述することができると考えられる。内部状態変数に関する処理や、割込処理についても、このように解釈する。

(a) 抽象的な記述　(b) 正しい具体化　(c) 誤った具体化

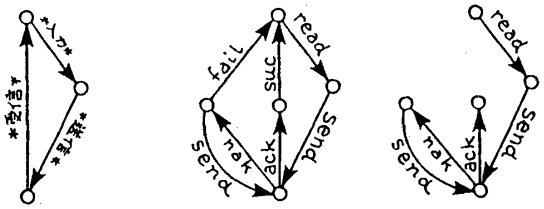

図３．意味検証

５．おわりに

以上、人間が親しみやすい、イベント順序的な記述から、計算機アクセス可能でインプリメントや検証の自動化に適した、状態遷移表による記述を作成するためには、どうすればよいか、という問題について、その実現方法を検討した。妥当な結果を得るためには、大まかな枠組から具体的な細部へと、段階的に仕様を記述することが有用であると考え、そのための記述技法について述べた。今後、さらに検討を進め、プログラム作成および評価を行なう予定である。

参考文献

[1]越田一郎："プロトコルの記述と変換を目的とした時間論理の応用",東京大学学位論文, Dec. 1984.