情報処理学会第37回(昭和63年後期)全国大会

プロトコルの段階的詳細化を支援するプロトタイピング・ツール

4M-1

中原彰子，相田仁，斉藤忠夫
（東京大学工学部）

1．はじめに

通信プロトコルの実現のためには，実行可能仕様を用いたプロトタイピングが有効である．本稿では，現在開発中のプロトコル記述／検証支援ツール上での，プロトタイピングの実現方法について述べる．各モジュールの実現度が異なる場合には，プロトタイピングにより得られた情報を，仕様の詳細化に役立てることができる．本稿ではまた，このような，仕様が抽象的な記述を含む場合のプロトタイピングについて考える．

2．背景と目的

2．1　論理検証とプロトタイピング

ＦＳＭや時間論理に基づくプロトコルの論理検証では，そのモデルの範囲内で完全な検証を実現することが可能である．しかし一般に論理検証では，検証項目が限られており，複雑な要求を含むプロトコルに対して，完全な正当性の証明を行うことは困難である．このため，プロトタイピングを用いて動作の確認を行うことが重要となる．

2．2　抽象的な記述の扱い

仕様を直接実行することにより，仕様作成の初期の段階において，容易にプロトタイピングを実現することができる．ここで，抽象的な記述を扱う必要が生じる．動作確認の目的のためには，仕様が抽象的な記述を含む場合にも，仕様の実行を続けることが望ましい．この場合には，プロトタイピングの結果得られる情報は，誤りではなく，抽象記述に対する制約としての役割を果たす（図１）．

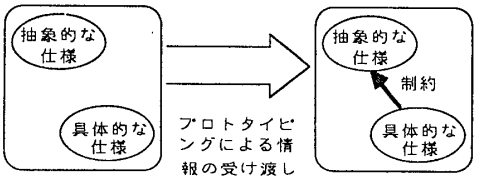

図１．基本概念

3．システムの概略

本節では，我々がすでに提案したプロトコル形式記述言語であるイベント順序記述[1,2]を，インタラクティブに実行するシステムの概略を述べる．

3．1　イベント順序記述によるモジュールの表現

プロトコルを構成する各モジュールの形式的表現として，「イベント順序記述」を用いる．イベント順序記述は，変数処理部を持つ拡張有限状態遷移機械に基づく記述法である．この記述法では，入出力等のイベント並びのうち，記述対象の動作順序として許されるものを，ルールの形で宣言的に規定することにより仕様を表現する（例-a）．

3．2　モジュールとインタラクション点の宣言

複数の仕様モジュールの間で模擬的な通信を実行し，プロトタイピングを実現する．まず，仕様モジュール間に通信路を設定するために，モジュールとインタラクション点の宣言を行う（例-b）．

例.

(a) イベント順序記述による
モジュール仕様の記述

module_name:
Ab_sender

module_name:
Ab_receiver

(receive ACK ?seq)
(T (not (= ?seq Xs)))
-> (send ?data Xs)

ルール

(b) モジュールとインタラクション点の宣言

```
(declare_module
        sender1 Ab_sender)
(declare_module
        receiver1 Ab_receiver)
(declare_interact inter1
        (sender1 receive)
        (receiver1 send))
```

sender1　receiver1

inter1

モジュールの宣言により，時間を追って変化する内部状態を持つ，モジュールのインスタンス.（プロセスに対応，以下単にモジュールと呼ぶ）を生成する．このモジュールは，イベント順序記述で定められた動作規則にしたがって動作する．

インタラクション点の宣言により，各モジュール内で定義される入力／出力イベントを，特定のインタラクション点に対応付ける．このインタラクショ

A Prototyping Tool for Protocols to Support Stepwise Refinement

Akiko NAKAHARA, Hitoshi AIDA, Tadao SAITO
University of Tokyo

ン点を介してデータを交換することにより，モジュール間の通信を実現する．

3．3　インタラクション点におけるデータの交換

データの交換では，必要に応じてモジュールを起動し，結果として生じる入出力動作を観察する．具体的な交換の手順を以下に示す．

①起動された各モジュールは，過去のイベント列を参照し，与えられたルールにしたがって，次に起こすべき入出力動作を決定する．

②入出力動作が決まると各モジュールは，対応するインタラクション点にデータ交換の要求を出したのち，待ち状態にはいる．

③交換要求を受け取ったインタラクション点は，データの交換を待っている仕様モジュールを捜し，もし待ちがあれば交換を成立させる（いわゆるランデブーモデル）．次節で述べる抽象処理は，この交換データの照合時に行う．

④交換の待ちがない場合には，データ交換を行う相手モジュールを捜して実行させる．それでも交換が成立しない場合には，失敗とみなす．

3．4　非決定性分岐の提示と選択

実行可能なイベントの候補が複数ある場合には，ユーザにそれを提示／選択することにより，インタラクティブに処理を進める．また，使い勝手を考慮して，バックトラックの機能をつける．

4．抽象処理

本節では，仕様が抽象的な記述を含む場合に，結果を制約として伝える方式について検討する．

4．1　抽象的な記述

従来より，プロトコルの論理検証について，動作に関する部分とシーケンス番号など変数に関する部分は別々に記述/検証すべきだという議論が行われてきた[3]．プロトタイピングにより詳細化を支援する場合にも，上記の枠組みを適用することが有用であると考えられる．そこで，イベント順序記述上の抽象的な表現として，次の２つを考える．

(a)動作に関する記述である「マクロイベント」

(b)変数に関する記述である「条件付き変数」

以下，それぞれついて簡単に述べる．

4．2　マクロイベントと部品合成による詳細化

入力／出力イベントは，分割不可能な基本単位であり，仕様内で唯一に定まるイベント名と，受渡しされるデータの組で表現される．これに対してマクロイベントは，「コネクション設定」のように，ある期間内に生じる一連のイベントを表す．記述中で用いられるマクロイベントは，最終的には入出力イベントの列に展開される．このときマクロイベントは，展開された入出力イベント列の到達可能性を示す．

仕様の実行において，相手先モジュールの動作仕様が具体的に定まっていない場合には，交換が無条件に成立すると考える．実行結果は，要求としてインタラクション点に蓄えられる．この情報より，プロトコル部品合成の手法を用いて，相手先モジュールの未定義マクロイベントの詳細化を行う（図２）．

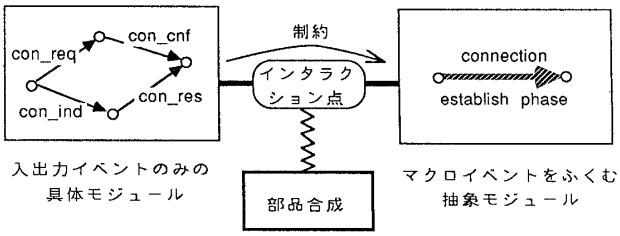

図２．動作に関する抽象処理

4．3　条件付き変数と意味単化

イベントの引数として，値が具体的に定まらない非束縛変数を用いる場合がある．これらの型付けされた非束縛変数には，必要に応じて適当な条件式を付与することができる．これを条件付き変数と呼ぶ．

このような条件付き変数については，データ交換時にインタラクション点において，条件式の意味を考慮したマッチングを行う．これにより，各モジュールで変数に付与された条件を相手に伝えることができる（図３）．

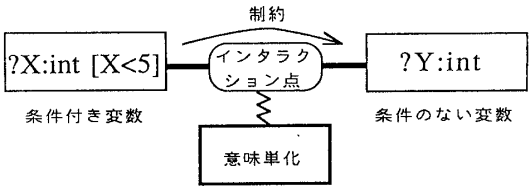

図３．変数に関する抽象処理

5．おわりに

以上，プロトタイピングにより段階的詳細化を支援する方式の概略を述べた．本方式では，入出力動作と内部変数という２種類の抽象記述を扱い，インタラクション点を介して，抽象記述の内容を制約として相手モジュールに伝える．現在，試作システムを作成中であり，今後さらに検討を進め，具体的な評価を行う予定である．

参考文献

[1]中原,相田,斉藤,猪瀬："プロトコルのイベント順序表現から状態遷移表現への自動変換，" 情報処理学会第34回全国大会．

[2]中原,相田,斉藤："イベント順序に基づくプロトコルの形式的記述とその処理系，" 情報処理学会第35回全国大会．

[3]G.v.Bochmann: "A General Transition Model for Protocols Communication Services," IEEE Trans., Vol. COM-28, No.4, Apr.1980.