ICOT Technical Memorandum: TM-0239

TM-0239

時制命題論理を用いた部品からの並列プログラム合成

内平直志，本位田真一
(東芝)

November, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

時制命題論理を用いた
部品からの並列プログラム合成

内平直志　　本位田真一
（株式会社　東芝　システム・ソフトウェア技術推進部）

Ｐｒｏｌｏｇベースの並列オブジェクト指向言語ＭＥＮＤＥＬ／８６のオブジェクトはプログラム部品として良い性質を持っている。ＭＥＮＤＥＬ／８６におけるプログラム合成は、①部品の結合と②同期部の生成の２つのフェイズから構成されるが、本稿では②について述べる。部品間のメッセージの同期に関する条件を時制命題論理式で与えると、それを守るようにメッセージの流れを制御するコントローラが自動生成される。

[1] はじめに

近年、ソフトウェアの再利用あるいは部品化に対する期待は非常に高い。その期待に答えるべく、研究指向のものから現実志向のものまで、様々なレベルで様々な手法を用いた試みが成されてきた。その結果、我々の職場でもプログラマの意識の向上と相俟って、部品化・再利用は徐々に軌道に乗りつつある。またそれに伴い、プログラム部品の結合／合成に関する研究も数多く報告されている。しかしながら、これらは主に逐次プログラムを対象とするものであり、同期という独特な側面を持つ並列プログラムを対象としたものではなかった。

従来、並列プログラムに関しては、検証という面から論理学を利用した多くの仕事があった。その一つとして時制命題論理（Propositional Temporal Logic　ＰＴＬ）で並列プログラムの仕様を記述し、それを検証するものがある。ＰＴＬが検証にとって有利な点は、ＰＴＬは決定可能であり、その決定手続きもいくつか提案されていることである［FT85］。MannaとWolperは、この決定手続きを利用して時制命題論理及びその拡張論理（Extended Temporal Logic ＥＴＬ）で記述した仕様から、並列プログラムの同期部分を自動生成する手法を示した［MW84］［Wo82］。これは仕様の無矛盾性を証明する過程で生成されるモデルグラフを、プロセスの状態遷移図とみなすことによって、仕様にあった同期をとるためのプログラムコードを生成するものである。

我々は、自動プログラミングの現時点での最も現実的なアプローチは、プログラムを部品から合成する方法であると考えている。MannaとWolperの方法は、同期部分は生成するが、それ以外の部分は人間がプログラムしようというものであった。そこで我々は、『それ以外の部分』は部品群として予め用意しておき、部品群の中からいくつかを組み合わせ、結合し、最後に同期部分をMannaとWolperの手法によって生成することによって目的とする並列プログラムを合成する手法を提案する。既に部品の検索／結合については、Ｐｒｏｌｏｇベースの並列オブジェクト指向言語ＭＥＮＤＥＬにおける部品検索／結合として提案している［USKH86］。　本報告では、MannaとWolperの方法を応用し、既に検索／結合されたＭＥＮＤＥＬプログラムの部品の組に対して、ＰＴＬで与えた仕様を満たすような並列プログラムの同期部を生成するフェイズについて述べる。

[2] 並列プログラムにおける部品化と再利用

並列プログラムにおける部品化とは

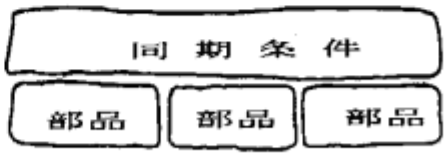
部品は、自分自身がどの様に使われるかを知らない。故に自己完結の度合いが高いほど良い部品といえる。並列プログラムの場合は、タスクやプロセスのように並列に動きうる単位プログラムが部品である。部品は他の部品と通信したり同期をとったりする。部品の自己完結度を高めるために、同期機構は、共有変数に基づくものよりメッセージ伝達に基づくものであるほうが良い。また同期をとる相手の部品は毎回異なるわけだから、通信チャンネルでメッセージ伝達を行うときは、間接指定のチャンネルでなければならない。ＭＥＮＤＥＬでは、

プロセス　＝　オブジェクト　＝　部品

であり、オブジェクト間の同期は１対１一方向非同期式通信パイプによるメッセージ伝達で行う。また、パイプは間接指定である。それゆえＭＥＮＤＥＬは部品化に適した言語仕様である。

同期部分の分離

同期は、相手がいるからこそできる。相手が不明である部品にとって同期は意味が無い。部品を使う側で同期の仕様を与えるのが本当である。そこで、同期に関する記述は部品の外側で与え、部品の中味の記述を同期の記述と切り離す。この考え方は順路式と同じである。本システムでは、同期部分の記述をゲートコントローラ仕様として表わす。



対象の限定 …… テキスト処理

部品化及び再利用が成功するための秘訣は、対象となる分野を限定することである。ここではテキスト処理ソフトウエアにおける部品化／再利用に限定する。

テキスト処理を選んだ理由を述べる。ＭＥＮＤＥＬのパイプとＵＮＩＸのパイプは同じ概念である。ＵＮＩＸは多くのテキスト処理プログラムを標準装備（sort cat cb spell wc ……）しており、シェルにおいてこれらのプログラムを組み合わせ、パイプでつなぐことにより、様々なテキスト処理が簡単にできた。これを部品結合と呼ぶならば、ＵＮＩＸのシェルはテキスト処理において、非常に単純かつ実用的な部品結合の手段を提供しているといえる。テキスト処理を対象としたとき、ＭＥＮＤＥＬはＵＮＩＸと同じように『実際に役立つ部品結合』を提供できるであろう。すなわち同じ並列プログラムでも『哲学者の食事』の哲学者やフォークを部品と考えるより、テキスト処理の各ルーチンのほうが部品として自然である。

［3］ＭＥＮＤＥＬ／８６

ＭＥＮＤＥＬ／８６はＰｒｏｌｏｇベースの並列オブジェクト指向プログラミング言語である。従来のＭＥＮＤＥＬ［HIOK88］との主な違いは、

メソッド ＝ プロダクションルール

になったことである。つまり各メソッドの起動条件としてＯＰＳ５相当の記述ができるようになり、記述能力は格段に向上した。新しいＭＥＮＤＥＬはオブジェクト指向言語であるとともに分散協調型プロダクションシステム記述言語でもある。ここでは、ＭＥＮＤＥＬ／８６のオブジェクトとパイプとメソッドについて簡単に説明する。

オブジェクト

ＭＥＮＤＥＬ／８６のオブジェクトは、各々並列に動く。オブジェクトは数個のメソッドと内部状態変数から構成される。

パイプ

オブジェクト間の通信はすべて１対１一方向非同期式通信パイプを経由して行われる。

図１ オブジェクトとパイプ

メソッド

メソッドはパイプからの入力データを取り込み、あるいは内部状態変数を参照し、その起動条件が満たされれば実行される。そして実行の最後にパイプにデータを送り出す、あるいは内部状態変数の内容を書き換える。メソッドの途中ではパイプや内部状態変数との入出力は一切できない。

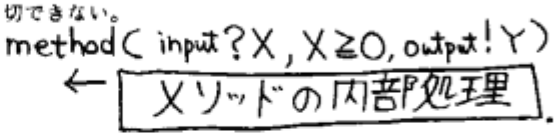


図２ メソッドの例

［4］ゲ── トとゲートコントローラ

本システムでは、ＭＥＮＤＥＬ／８６に『ゲート』と『ゲートコントローラ』という機構を追加する。このゲートとゲートコントローラにより同期の制御が行われる。

ゲート

ゲートは各パイプに１つ存在する。ゲートは開閉する。１回開閉する毎に１個のメッセージがゲートを通過できる。メッセージがパイプに溜まっていない場合は開閉できない。これはペトリネットのトランジションに相当する。

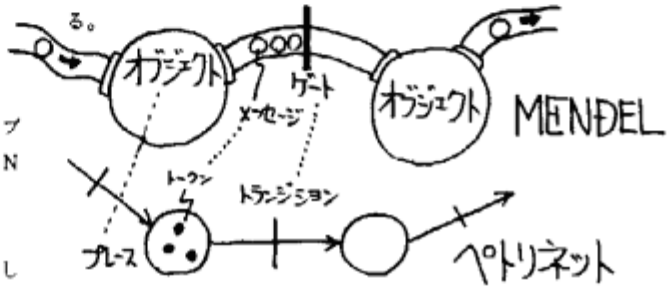


図３ ＭＥＮＤＥＬとペトリネット

ゲートコントローラ

全てのゲートの開閉を集中的に管理する。全てのオブジェクトがｗａｉｔ状態になるまで次のゲートを開けない。また、同時には１か所のゲートしか開けられない。

［5］ＰＴＬによるゲートコントローラの仕様記述

ＰＴＬ

ＰＴＬは命題論理（ここではand …&，or…#，not …￣，Imply …=>と表わす）に次の時制オペレータを追加した論理体系である。

[]ｆ (always)	ｆは将来の全ての状態で真
<>ｆ (eventually)	ｆは将来のある状態で真
@ｆ (next)	ｆは次の状態で真
f1$f2(until)	f2が真になる最初の状態までf1が真

ゲートの開閉が原子論理式

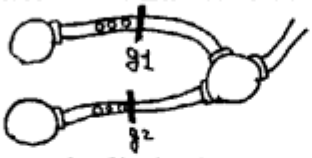
ゲートの開閉という事象をＰＴＬの原子論理式に対応させる。例えばゲートｇの開閉を原子論理式ｇで表わすと、ゲートｇが常に開閉し続けることは[]ｇ、少なくとも１回開閉することは<>ｇと表現できる。またゲートコントローラが毎回１箇所だけ開閉するという条件は、各状態において真である原子論理式はただ１つであること

に対応する。これをsingle event conditionと呼ぶ。

簡単な例

ｇ１とｇ２を交互に開閉するという仕様は次のようになる。

[]（ｇ１->@ｇ２）
[]（ｇ２->@ｇ１）



［6］ゲートコントローラの自動生成

ＰＴＬで記述された仕様からManna&Wolperの手法により状態遷移図を生成する。この手法はＰＴＬの決定手続きであるタブロー法に基づいている。その詳細はここでは省略する。とにかく状態遷移図ができる。状態遷移図の各アローに原子論理式であるゲート名が対応する。このときアローに対応するゲートを開閉しながら、この状態遷移図上を公平に移動しているかぎり、それはＰＴＬで記述された仕様を満たしている。前の例からは、次の状態遷移図が生成される。この状態遷移図上を移動すればｇ１とｇ２が交互に現われ、仕様は満たされる。

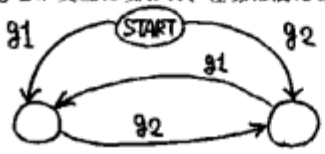


図４　状態遷移図

ゲートコントローラは、この状態遷移図に基づいてゲートの開閉を行う。公平さは、候補が複数個あったときに各状態ごとに、過去に開閉してから最も時間の経っているものもしくは１度も開閉していないものを優先するという戦略をとることで実現する。

［7］ゲートコントローラ仕様記述用マクロ

ゲートコントローラの仕様はＰＴＬで書かれるが、ＰＴＬは必ずしも直観的に解りやすいとは言えない。テキスト処理という分野限定により、仕様はある程度パターン化できる。そこで、そのパターンをマクロ化することによって、仕様記述の労力を削減し、読解性を高めることができる。現在用意しているマクロは次の４パターンである。

★ＰＲＥＣＥＤＥ（ｇ１，ｇ２）
　ｇ２が開く前に、少なくとも１回はｇ１が開いていなければならない。

★ＦＩＮＩＴＥ（ｇ）
　ｇを通過するメッセージは有限個である。ｇは有限回開きメッセージが終了するとｅｏｇ（ｅｎｄ　ｏｆ　ｇ）を発効する。

★ＨＡＬＴ
　いつかは停止する。

★ＯＰＥＮ（ｇ）
　ゲートｇは常に開放状態にしておく。

これらのマクロは、それぞれ次のようにＰＴＬに展開される。

PRECEDE(g1,g2)　…　￣（￣g1＄g2）＃[]（￣g2）
FINITE(g)　…
<>ｅｏｇ＆[]（ｅｏｇ->@（[]（￣ｅｏｇ＆￣ｇ））
HALT　…
<>ｈａｌｔ＆[]（ｈａｌｔ->@（[]ｈａｌｔ））
OPEN(g)　…　ｇはＰＴＬ記述に展開しない。

［8］例題：キーワードカウントプログラム

キーワードカウントプログラムを合成する。あらかじめ部品として次の３つのオブジェクト（部品）が用意されているとする。

●単語切り出しオブジェクト
　入力される文字ストリームから単語を切り出して出力する。

●キーワードチェックオブジェクト
　入力される単語がキーワードと合致すればそれを出力する。合致しなければ何もしない。

●サマリ作成オブジェクト
　入力された合致結果を集計してサマリレポートを作成し出力する。

この３つのオブジェクトが次の図のように部品結合されたとする。

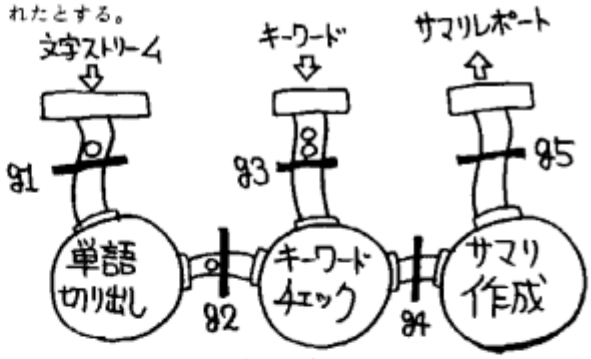


図５　部品結合の結果

このとき同期に関する仕様として以下のことを要求する。

①キーワードも文字ストリームも有限である。（ゲートｇ１，ｇ３を通過するメッセージは有限である。）
②キーワードチェックオブジェクトでは、キーワードの入力が全て終了してから単語切り出しオブジェクトからの入力を受け付ける。（ｇ３が打ち止め（ｅｏｇ３）となるまでｇ２を開かない。）
③ゲートｇ４は開放状態である。
④サマリレポートは必ず出力される。（ｇ５は必ず１回は開閉する。）
⑤最後にサマリーレポートを出力する。（ｇ５を開いた後はｈａｌｔする。）

⑧いつかは停止する。

これをＰＴＬ及びマクロで記述すると、以下になる。

①ＦＩＮＩＴＥ（ｇ１），ＦＩＮＩＴＥ（ｇ３）
②￣ｇ２＄ｅｏｇ３
③ＯＰＥＮ（ｇ４）
④<>ｇ５
⑤[]（ｇ５=>@ｈａｌｔ）
⑥ＨＡＬＴ

この仕様から次の状態遷移図が自動生成できる。

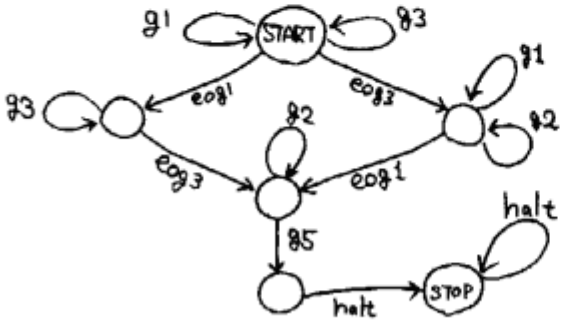


図６　生成された状態遷移図

ゲートコントローラは、生成された状態遷移図に従ってゲートの開閉を行う。このゲートコントローラと結合後の部品群を１セットとしてキーワードカウントプログラムの完成である。

[9] まとめ

我々は、部品の再利用をベースにした並列プログラムの合成システムを開発している。このプログラム合成システムは大きく２つのフェイズに分かれている。

①部品の検索／結合
②同期部分の生成

今回の報告の中心は、②であった。すなわちＰＴＬで記述された同期仕様を満たすようなＭＥＮＤＥＬのゲートコントローラの生成について述べた。

このプログラム合成システムのアプローチには、明らかに幾つかの限界（例えばＰＴＬの記述能力の限界）があるが、適用対象をテキスト処理に限定し、磨きをかければ実際に役立つものになりうると考えている。

時制論理で記述された仕様からプログラムの同期部を生成する方法の研究には、我々が基にしたManna&Wolperの他にClarke&Emerson［CE81］がある。また同様の原理をハードウェア設計における状態遷移図の生成［FTM84］や並列システムのスケジューリング則の生成［KI82］に適用した研究もある。また、同期部をその他の部分から切り離して記述する点で順路式［Ha76］がよく似ている。これらに対する我々の研究の意義は、同期部の生成をプログラム部品の再利用とリンクしたところにある。

今回報告したシステムには、まだ以下のような問題点が残っている。

①ゲートコントローラの生成に時間がかかる。
②合成されたプログラムが正しく動く保証はない。
③並列度が低い。

①は、タブロー法の効率的なインプリメンテーションの話である。②は、各部品固有の制約をゲートコントローラの生成に全く反映させていないことから生じる問題である。③は全てのプロセスがｗａｉｔするのを待って次のゲートを開くからである。今後これらに対して検討を加えていく。

謝辞

本研究は、第五世代コンピュータプロジェクトの一環として行われた。本研究の機会を与えて下さった新世代コンピュータ技術開発機構の各位に深く感謝する。

参考文献

［ＣＥ８１］E.M.Clarke E.A.Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic.Logics of Programs (Proceedings 1981). Lecture Notes in Computer Science 131.Springer-Verlag.1982.pp.52-71.

［ＦＴＭ８４］M.Fujita H.Tanaka T.Moto-oka. Specifying hardware in temporal logic & efficient synthesis of state-diagrams using Prolog.Proc. of FGCS'84.1984.

［ＦＴ８５］A.Pusaoka K.Takahashi.On QPTL and the refutation procedure on ω-graphs.ICOT Technical Report TR-132.1985.

［Ｈａ７６］A.N.Habermann. Introduction to operating system design.SRA.1976.

［ＨＵＯＫ８６］本位田　内平　大須賀　柏谷，推論型システム記述言語ＭＥＮＤＥＬ，情報処理学会論文誌 Vol.27 No.8.1986.

［ＫＩ８２］片井　岩井，非同期式同時進行システムに対する時制論理に基づくスケジューリング則の構成，計測自動制御学会論文誌Vol.18 No.12.1982.

［ＭＷ８４］Z.Manna P.Wolper. Synthesis of communicating processes from temporal logic specification.ACM Trans. on Lang. and Sys.,vol.6, No.1.1984.pp.68-93.

［ＵＳＫＨ８６］内平　関　柏谷　本位田，ＭＥＮＤＥＬにおける並列プログラムの部品結合，情報処理学会ソフトウエア工学研究会SE-46-8.1986.

［Ｗｏ８２］P.Wolper. Synthesis of communicating processes from temporal logic specification. STAN-CS-82-925.Stanford university.1982.