1E-3

ファームウェア開発支援システム CHEFの広域コンパクト化方式

細谷泰一　磯田定宏　小林吉純　石田亨
串間和彦　(横須賀電気通信研究所)

1. 従来方式の問題点と本方式の特徴

最適なマイクロプログラム(μP)を生成するためには、マイクロオーダ(μO)の移動置換をμP全域にわたって行い、マイクロ命令(μI)数を削減する広域コンパクト化が必要である。今日までに提案された広域コンパクト化方式は、まずセグメント(注1)内のコンパクト化を行った後、空フィールドがある場合に隣接するセグメント間でのμOの移動を行うものが中心であった。しかしこの方式には、①移動対象となるμOがセグメント内コンパクト化結果により限定される、②移動距離が短いためコンパクト化効果が十分でないという欠点がある。

筆者らは以下の特徴を有する広域コンパクト化方式を考案し、ファームウェア開発支援システムCHEF(注2)の一機能として開発中である。

①セグメント内、間のコンパクト化処理を別フェーズとせず、まずプログラム全域にわたってμOの並列実行可能性を解析する。その後、制御が集中し実行頻度の高いセグメントのμI数が最小となるようμOのつめあわせを行う。このためコンパクト化効果が大きい。

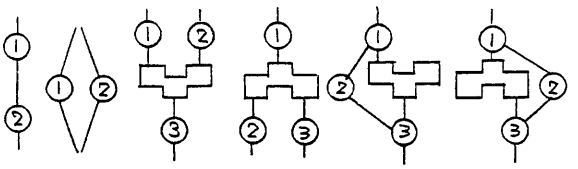
②プログラム全域より収集したフロー解析情報をもとに、隣接するμOだけに着目した微視的で単純な操作をプログラム全域にわたって収束するまで繰り返すアルゴリズムを採用している。このためあらゆる制御構造がコンパクト化対象となるなど汎用性が高い。

(注1) 先頭から順次実行され、必ず最後まで実行されるμIの列。

(注2) 3層の階層言語仕様と会話型のμP改良機能を特徴とする汎用ファームウェア開発支援システム。56年度信学会情報システム部門全国大会講演番号548参照。

2. 並列実行可能性の表現方法

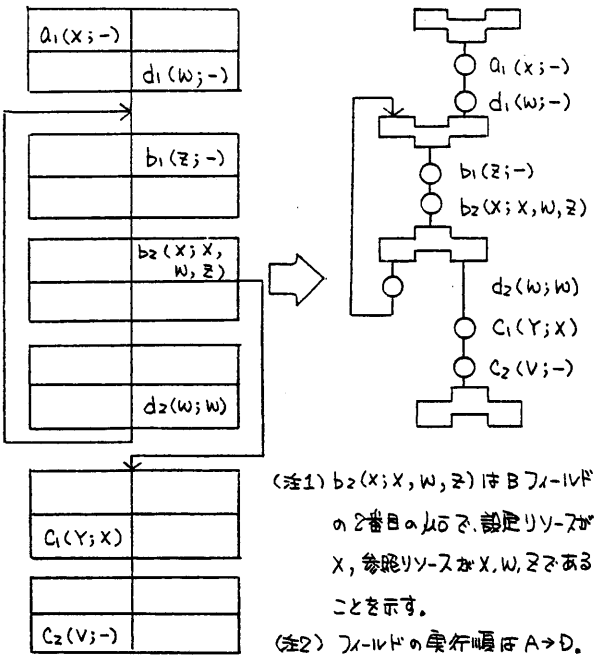
セグメント内の並列実行可能性の表現方法として、①μOを表わす節点、②μO間の実行順序を表わす有向枝を構成要素とするμPグラフ(図1(A)(B))が用いられてきた。筆者らはセグメン



(注) ⊤ は丁管、⊥ は人管を表わす。

(A) ①②が逐次実行されることを示す。
(B) ①②が並列実行されることを示す。
(C) ①又は②の実行後に③が実行されることを示す。
(D) ①の実行後に②又は③が実行されることを示す。
(E)(F) ②は①と③の間で実行されるが、①の属するセグメント及び③の属するセグメントのどちらに位置してもよいことを示す。

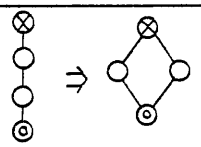
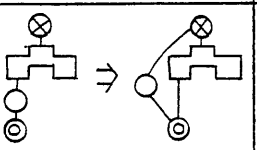
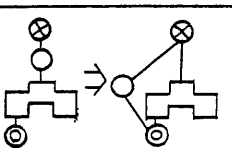
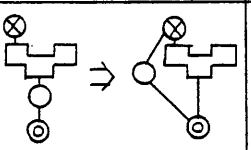
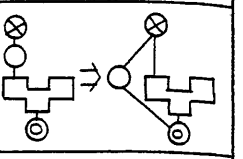
図1　μPグラフの基本的な表現の意味



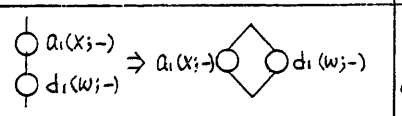
(注1) b2(X;X,W,Z)はBフィールドの2番目のμOで、設定リソースがX、参照リソースがX,W,Zであることを示す。
(注2) フィールドの実行順はA→D。

図2　入力μPと対応するμPグラフ

表1　グラフの変形規則とその適用例

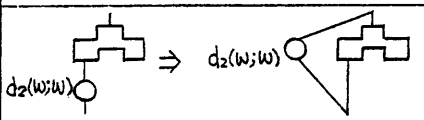
規則1	規則2	規則3	規則4	規則5
				

規則1の適用例



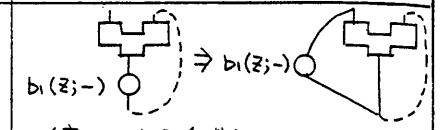
〈適用できる条件〉
① d_1の参照リソースがa_1の設定リソースでない。
② d_1の設定リソースがa_1の参照/設定リソースでない。

規則2の適用例



〈適用できる条件〉
① d_2の設定リソースが入管の他の出口でliveでない。

規則4の適用例



〈適用できる条件〉
① b_1の参照リソースがループの中で設定されていない。
② b_1の設定リソースがループの中でb_1以外で設定されていない。

ト間の並列実行可能性を表現するため③制御の集中分散を表わす制御管（Y管，I管）を導入し（図1(C)(D)）、さらにセグメントを越えるμ命の移動可能性を表現できるようμPグラフを拡張した（図1(E)(F)）。

3．コンパクト化処理手順

コンパクトルは、①入力μPからμPグラフの作成（図2）、②μPグラフの並列化、③μPグラフから出力μPの作成の3段階の処理に分れる。

(1) μPグラフの並列化

各μ命の設定参照リソースやμPグラフのフロー解析結果をもとに、μPグラフの全有向枝に表1に示す5個の変形規則を適用する。規則を適用できる有向枝がなくなれば、全μ命の並列実行可能性を表わすμPグラフが完成する。図2に変形規則を適用した結果を図3に示す。

(2) μPグラフから出力μPの作成

制御の最も集中するセグメントのμI数が最小になるようつめあわせを行い出力μPを作成する。セグメントの制御の集中の度合いは、両端の制御管の種別により、①Y-I、②Y-Y、I-I、③I-Yの順である。そこでこの順に各セグメントに必ず収容しなければならないμ命をまず収容し、その後μI数をふやさない範囲で収容できるμ命を収容する。図3に収容過程の一部を、図4に出力μPを示す。

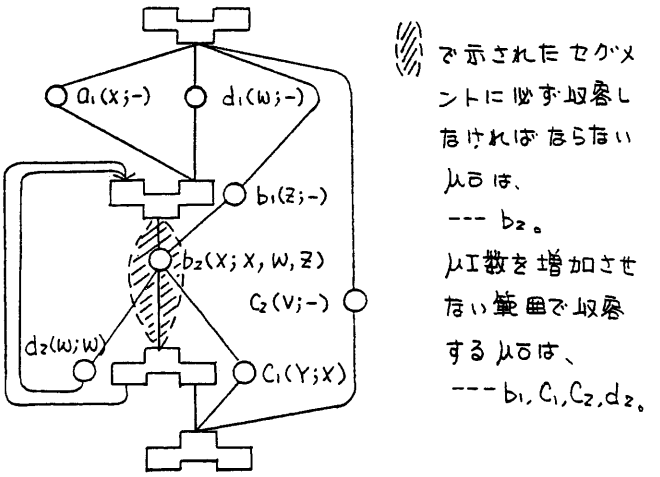


図3　並列化されたμPグラフ

で示されたセグメントに必ず収容しなければならないμ命は、
--- b_2。
μI数を増加させない範囲で収容するμ命は、
--- b_1, C_1, C_2, d_2。

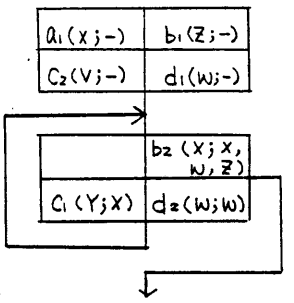


図4　出力μP

4．まとめ

CHEFコンパクト化機能は幅広いコンピュータに汎用的に適用可能とするため、①複数のμI形式や、②μ命間の複雑な配置条件にも対処できるよう設計されている。開発後の評価結果については別途報告の予定である。

並列からマルチエージェントへ

石田 亨
京都大学情報学研究科

マイクロプログラムは学生時代、萩原宏先生、富田真治先生のご指導で取り組んでいた。卒業論文では、研究室で開発されていた水平型マイクロプログラム制御の計算機 QA-1 のマイクロプログラムアセンブラを担当した。主記憶が 12KB しかない HITAC-10 を用いて、アセンブリ言語約 1 万行のソフトを開発した。当時は紙テープを用いていたので、プログラムは２巻となり、バグを修正するには相当の気力を奮い起こす必要があった。マイクロプログラムは、その当時から、ハードウェアとソフトウェアの境界として興味を感じていた。境界であるが故の設計の自由度と制約に、アーキテクトのバランス感覚が試されているように感じた。

電電公社横須賀通研に入所し、故花田収悦氏の研究室でソフト工学を学んだ。その後、学生時代の経験が買われたのか、ファームウェア記述言語のプロジェクトに配属された。当時電電公社では、計算機シリーズ DIPS が開発されていたが、NEC、日立、富士通 3 社の異なるアーキテクチャに基づいていたので、汎用的なマイクロプログラム記述言語を研究する理由があった。200Mのディスクを抱えて無人の大型計算機室を占拠し、マイクロプログラムを入れ替えて実験を繰り返したことを思い出す。この論文は、その中でも最も思い出深いもので、研究所への通勤時間に吊革に捕まりながら、頭の中で何度もマイクロオーダを並べ換えて生まれたアイデアである。参考文献も書いていないこの論文を見ると研究者として恥ずかしいが、当時は研究者という意識は全く希薄で、むしろ技術者として Aha!の瞬間を最初に経験したプロジェクトだった。この論文に記載されたアイデアは、その後、IEEE Transaction on Computers に掲載されている。

並列性への興味は、その後、プロダクションシステムと出会って、人工知能の中での並列処理に、さらに、分散人工知能、マルチエージェントシステムへと広がっていった。現在も社会情報学専攻で、仮想空間内での多数のエージェントのシナリオ記述言語を開発している。また、今後の研究として、メガスケール（100 万人、100 万台規模）のナビゲーションシステムを考えているが、そうした並列・分散の研究に物怖じしないで取り組めるのも、学生時代から並列処理の教育を受けたからではないかと思う。