様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２８ 年　 ６ 月 １３ 日現在

機関番号：２２６０４
研究種目：基盤研究(C)（一般）
研究期間：2013～2015
課題番号：２５３３００８７

研究課題名（和文）Timed CSPを用いた実時間並列システム検証プログラムの開発

研究課題名（英文）Development of a validity check program for real time parallel systems using timed CSP

研究代表者
福永　力（Fukunaga, Chikara）

首都大学東京・理工学研究科・教授

研究者番号：００１８９９６１

交付決定額（研究期間全体）：（直接経費）　　3,500,000円

研究成果の概要（和文）：本研究は並列計算処理プログラムの安定動作のための設計技法（形式言語）に関するものである．複数のユニットが並列に動作すると予測できない不具合により並列計算が立ち往生してしまう．それを避けるべく並列処理部分のシーケンスをユニットごとに設計図としてまとめておき、複数のユニットが並列動作した時の全体の並列動作の流れのみを形式的に確認する方法が研究されている．本研究はシーケンスをただイベントとして記述するばかりでなく時間の概念も導入して並列処理が安定に動作するか検証しようというものである．並列処理で生じると予測されるいくつかの不具合は設計段階で予知できる技法の開発を進めることができた．

研究成果の概要（英文）：The study is concerned with a validity check program for a parallel processing system. If some units are run in parallel, the system will be dead-locked with unexpected sequence failures. We have a tool to find them hidden in a system at the system design level. Our study has tried to refine the validity checker to discover invalidities caused by timing conflict beside simple event sequence. We have confirmed that some of timing invalidities possibly hidden in a system could be found with this newly refined checker.

研究分野：情報科学

キーワード：並列処理　形式手法　CSP理論　失敗・発散検査

１．研究開始当初の背景

2013年に本研究が科研費・基盤Cに採択され開始された．約40年前にC．A．R．Hoareにより提唱された並列システム構成記述の基盤となるCSP（Communicating Sequential Processes）理論[2]はその後、後継者達により並列システムのうちに内在するデッドロックやライブロックを見出す検査システム（失敗・発散詳細化検証；Failure Divergence Refinement; FDR）[3]の完成へと発展・進化した．CSP理論による並列システムの検証はそのFDRをもって一定の成果を上げたものであるが、並列に進行する複数のプログラムユニットの同期はそれぞれが生成するイベント群の特定のものを通してなされるものであった．そこに時間の概念は組み込まれてはいなかった．2000年以降、CSP理論は理論としてさらに発展し、時間の概念も組み込んだtimed-CSP理論[4]が登場した．FDRもその概念を組み込んだ新しいものになるべきであったが、その動きは表立って見られなかった．時間の概念を本格的に組み込んだ検証システムを完成させないと真の並列システムに内在する不具合の研究はできないと考えた本研究代表者達は2011年に「Development of an ML-based Verification Tool for Timed CSP Processes」なる研究をCPA（Communication Parallel Architecture）2011（リムリック、アイルランド）で公表した[1]．これはFDR とは異なる独自の検証システムでTimed-CSPを全面的に取り込んだものであった．その後このFDRとは一線を画すTimed-CSP Explorerの本格的発展を目指し本研究が開始された．

２．研究の目的

本研究は前節で述べたように並列システム（例えばマルチコアやネットワーク結合されて複数PCによる協調計算システム）のモデル検証ツールの構築をFDRとは別のものとして行うものである．特に実時間システムを意識した並列システムのモデル検証を行うもので、Timed CSPのもとで表現されているモデルの検証と詳細化（refinement）を行うツールである．

CSP はプロセスをそれが引き起こすイベントの並びとして表す．したがって 2 つのプロセスのイベントの起きた系列(トレース)を調べれば両者の並列システム内での整合性が確認できる．またあるトレース以降に決して起き得ないイベントの集合を通してプロセスのデッドロックを検知することができるし、ライブロックに陥るトレースを特定しプロセスがそのトレースをとる可能性があるかどうかの検証も可能である．これが CSP で並列システムを検証する場合のシナリオである．問題はイベントの起こる時間の記述が CSP では行えないことである．イベントの並びは緩く時間の 流れを表しているがイベントとイベントの間隔に時間を導入できない．　例えばコンピュータマウスのダブルクリックと 2 回の時間をおいたクリックの区別がつかないことになる．　また本質的に実時間システムである組み込みシステムの検証にいわゆるFDR2（FDR ver.2） を利用できないという問題が指摘されていた．　しかし 2000 年以降時間概念を CSP 記述に組み込んだ Timed CSP 理論が体系化され研究者の間で広 く認知されるまにでなってきた．理論そのものは線形時相論理を基にしている．記述も CSP との親和 性が高い．我々はこの Timed CSP をもとに昨年（2011）にその検証を行うツールを開発しその概要 を公表した

並列システムの時間概念が組み込まれていないプロセス代数 CSP のもとでの 検証・詳細化ツールはすでに存在することを前節で述べた（FDR）．が時間概念を加えた Timed CSP での最終的な形でのツールは 未だ存在しない．申請者がすでに途中段階の時点で公

表した Timed CSP Explorerの完成を目指そうというものである.

３．研究の方法

第1節に述べたように我々は2011年にTimed CSP Explorerを未完成ではあるがそこまでの出来具合を公表した．未完成ではあるのに公表に踏み切った背景にはさらに何年か完成までにかかってしまうが、多分さまざまな研究者がtimed-CSPベースの検証システムの構築に取り組み初めているかもしれない．そこで我々もその方向で研究を開始していると言っておいたほうがよいのではという判断といくつかの並列システムで起こる不具合のうちその時点で1つは完全に検証できたということで発表に足る意義があると判断したからである．そこで我々はCSP Explorerの未公開版をもとにこのプログラムの本体、ユーザインターフェース（UI） の両者の充実をはかりアプリケーションとして公開することかを研究の最終目的と定めた． 本体部分の充実という面では現在の版では時間情報を入れた（Timed）トレース（TT）のみでの並列システム安全性の検証が成功しているが、より充実した並列システムの検証には失敗および発散トレース集合の構成のアルゴリズムを開発しそれを組み込むことが必須であり、その開発を行うことを考えた. 従来の CSPでは豊富に検証例題が用意されていた（あるいは様々な問題を考案してきている）が Timed CSP ではまだごくわずかである． 時間情報を含めないと有効な検証がなし得ない例題をソフトウェアのみでなくハードウェア設計事例も含めて考案することも同時に進めることを目指した． UI、デバッグ環境の整備を経て最終的に公開していくことを目標とした．したがってその研究は以下の段階を踏んでいくとした.

1. Timed CSP で導入されたプロセス代数演算子のセマンティックモデルの構築、
2. 操作意味論表現の演算子のML関数としての実装、
3. Timed CSP にあわせた CSP_M の拡張、
4. 検証アルゴリズムの開発、
5. MLによる検証部本体、検証デバッグ部実装、
6. 検証例題の制作・収集による検証結果の信頼性の向上
7. MLあるいはより適切な言語によるヒューマンインターフェースの開発

４．研究成果

時間概念を取り込んだ CSP による並列処理システムの失敗-発散-詳細化解析プログラムの作成が本研究の具体的課題である．1 年目は CSP と同様な並列処理記述理論であるπ-calculus の動向を詳細に調べ、CSP に欠けているπ-calculus のプロセス記述についてその CSP への拡張可能性について調べた.

2 年目は CSP 理論がハードウェア、ソフトウェアとして実現されている例を詳細に調べた． ハードウェアの例では XMOS であり、ソフトウェアでは occam-pi、またそれを発展させる Guppy 計画、 XMOS 上での Sire コンパイラである．XMOS は現在 CSP 理論を実装化した FPGA のようなハードウェアチップでその回路記述は CSP 理論をもと行われる．また ST20 というハードウェアもその実現は 20 世紀の終わりではあるが、CSP 理論をもとにリアルタイムカーネルを実装するなど、興味深い CSP の実現例である．我々はその詳細を検討し、組込みシステムやリアルタイム OS 上での安全な並列処理システムの実現にはやはり時間概念を組込んだ timed-CSP の必要性を再確認した.

ここでさまざまな例を収集したのであるが意外と並列システムの設計において時間概念が重要となる例を見出すことに時間がか

かってしまった．この段階でとても有効であったのは CSP 研究会という研究会号であった．CSP コンソーシアムという団体（代表：松井和人氏）が主催している研究会でだいたい定例会が年に 2 回ほど開催される．参加人数は 30〜50 人程度で参加者も大学、研究所研究者、IT 企業形式手法プログラム開発者、並列システム設計のハードウェアエンジニア、プログラマである．この団体、および研究会を通じていろいろな有意義な助言や情報を得ることができた．また報告者もこの会で積極的に自分の研究成果や調査を公表することにより大きく CSP 理論のもつ多面性を認識できた．この研究を常に緊張感をもって継続できたのもこの研究会の存在が大きいといってよいであろう．

3 年目にその 2 年間で収集した例をもとにどのような形で失敗-発散-詳細化解析を行うえばもっとも並列システムの安全性（失敗がなく（プロセスの軌跡が追えて）かつ発散のない（デッドタイム、ライブタイムのない）プロセス並列処理）の検討に入った．つまり前節で述べた研究計画のステップのうち、1 から 3 まではなんとか到達し、続いて 4 に向かうところである．研究計画では 4 は 2 年目の早い時期に到達してしかるべきであったが、例題の収集やフレームワークとして採用するコンパイラ技法の不手際（最初は ML を従来通り使用しようと思ったが、コンパクトなコンパイル技法であるナノパスコンパイラ技術を利用すれば CSP_M で書かれたシステムをそのままコンパイラに取り組めると考えてしばらくこの技法の習熟に集中していたのだが、その後この技法に多くの問題があることがわかりそこでまた ML に戻るか Haskell を採用しようかと大きく逡巡してしまった）で、検証アルゴリズムの開発に至ったのが 3 年目に入ってからになってしまった．

並列システムの検証には前述したようにトレース、失敗集合、発散トレース集合を構成しその解析が必要である．我々のツールはだトレース解析の検証のみが行えるものであった． 一般的にトレース解析ではシステムの安全性が検証されるのみで、それ以上の（デッドロック、ライブロックの有無の）検証・詳細化には失敗や発散トレース集合の構成が必要であった．このアルゴリズムの開発中に CSP 研究会からの情報で Timed-CSP を組み込んだ検証システムの完成の情報を聞き及んだ．tock-CSP である[5]．この tock-CSP は参考文献[5]を基に時間を離散化させ（離散化度は検証者による）その離散時間の中で Timed-CSP 検証を行うものであった．離散時間ではあったがこのプログラムを検討するとかなりきめ細やかに Timed-CSP でモデル表現が行えることがわかった．

同種のプログラムでオックスフォード大学の FDR3 は時間概念を取り入れた、しかしあくまで no-timed CSP ベースでありながら、時間概念をイベントとして表現し時間概念をある程度含むプロセス記述機能の拡張（しかし）に成功し、シンガポール大学の PAT3 は一部時間概念を取り入れた CSP の検証を完成させた．そういった状況で現在 CSP そのものの発展と拡張の可能性を確認したところで本研究はそれらの詳細を調べ、どのように我々の timed CSP を発展させるか新たな曲面に入ったところである．

引用文献：

[1] Yamakawa, T, T. Ohashi and C. Fukunaga, “Development of an ML-based Verification Tool for Timed CSP Processes”, Proceedings of CPA (Communication Parallel Architecture) 2011, Limerick, Ireland, 19-22 June 2011, pp. 363-376.

[2] Hoare, C. A. R. , ”Communicating Sequential Processes”, Prentice Hall, 1985

[3] Formal Systems (Europe) Limited, “Failures-divergence refinement: FDR2,” http://www. fsel. com

[4] Schneider, S. , Concurrent and Real-time Systems, The CSP Approach, Wiley, 2000

[5] Armstrong, P., G. Lowe, J. Ouaknine, A.W. Roscoe, “Model checking Timed CSP“, Oxford Computer Center Preprint, 2011.

５．主な発表論文等

〔雑誌論文〕（計　0　件）

〔学会・研究会発表〕（計　5　件）

[1] 第 16 回 CSP 研究会（2015 年 11 月 21 日@東洋大学、東京都文京区）
福永　力「Native Python の並列処理」

[2] 第 15 回 CSP 研究会（2015 年 6 月 6 日@東洋大学、東京都文京区）
福永　力「ST20 プロセッサ -Real Time CSP 実現に向けた試み」

[3] 第 14 回 CSP 研究会（2014 年 11 月 29 日）福永　力「CSP ベースの並列処理記述言語の動向 」

[4] 第 13 回 CSP 研究会（2014 年 5 月 24 日@東洋大学、東京都文京区）
福永　力「XMOS のコンテキストスイッチング」

[5] 第 11 回 CSP 研究会発表資料（2013 年 5 月 25 日@東洋大学、東京都文京区）
福永　力「π-calculus と occam-π」

〔図書〕（計　0　件）

〔産業財産権〕

○出願状況（計　0　件）

６．研究組織

(1)研究代表者

福永　力　　（Chikara Fukunaga）

首都大学東京・大学院理工学研究科・教授

研究者番号：00189961

(2)研究分担者（0）