A-019

分散相互排除アルゴリズムの仕様記述言語 LNT によるモデル化と検証

Modeling and Verification of Mutual Exclusion in Distributed System with Specification Description Language LNT

橋爪由道 †　　和﨑 克己 ††
Yuto Hashizume　Katsumi Wasaki

1　はじめに

本研究では，ノードの要求に論理時計値など優先度を表す値を付け，それが競合した際に優先度に従って動作し，ライブロックを防ぐといった性質がある，優先度を用いる分散相互排除アルゴリズムの作成・検証を行う [1]．仕様記述には LNT 言語を用い [2]，記述したアルゴリズムが優先度通りに動作しているか，どのノードも一度は利用権を要求しているかという 2 点について CADP toolbox[3] で検証する．

2　優先度を用いる分散相互排除アルゴリズム

ある共有リソースを利用するリクエストが複数存在する場合にそのうちの 1 つを選ぶ問題を相互排除問題といい，特に分散システム上で相互排除問題を扱う問題を分散相互排除問題という [1]．

以下は相互排除アルゴリズムが満足すべき条件である

- 複数のプロセスが同時に利用権を持つことがないこと
- 永遠に利用権を持たないプロセスが存在しないこと

これらの性質を満足する優先度を用いる分散相互排除アルゴリズムについて以下で述べる．

2.1　優先度を用いる分散相互排除アルゴリズム

優先度を用いる分散相互排除アルゴリズムは，リクエストに優先度を示す値を付け，競合した場合に優先度の高い方を選ぶ方法である [1]．優先度の比較について全プロセスで共通のルールを設定することでライブロックを防ぎ，条件を満たす．優先度の付け方にはいくつか方法があるが，今回は論理時計とノード毎に固有に持たせた優先度を用いる．基本的には古いリクエスト (論理時計値が小さいもの) を優先するものとし，論理時計値が同じときにノード固有の優先度を比較する．その結果，要求の優先度はいずれ最大になり，利用権を得ることができる．なお，本研究では検証にあたって論理時計を実装するモデルと実装しないモデルを作成する．

2.2　ノードの挙動

資源を利用したいノードは自分以外の全ノードに対して，利用権を要求する request メッセージを送信する．request メッセージを受信したノードが利用権を要求していなかった場合，利用を許可する OK メッセージを返す．利用権を要求していた場合，そのノードが保持する優先度と受信した request メッセージに付いている優先度を比べ，そのノードが保持する優先度の方が高ければ利用権を待つリストに受信したメッセージのアドレスを保存し，OK メッセージが返ってくるのを待つ．受信したメッセージの優先度が高ければ，送信元ノードに OK メッセージを返し，資源の利用が終わるのを待つ．自分以外の全ノードから OK メッセージがを受け取ったノードは資源を利用する．資源の利用を終えたノードは，利用権を待つリストにある全ノードに OK メッセージを送信する (図 1)．

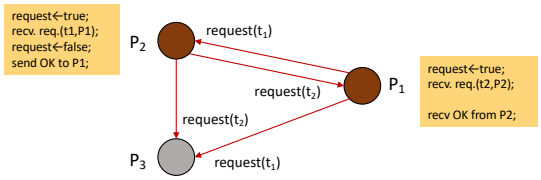

図 1　メッセージ通信による分散相互排除 (P_1 が P_2 より優先度が高いケース)

3　LNT 記述

3.1　ノード間の通信

図 2 のようにノード間を中継するネットワークをプロセスとして作成する [4]．ノードとネットワークはアクション send と recv で同期しており，それぞれノードからネットワーク，ネットワークからノードにメッセージを送信する際に用いる．メッセージにはその種類と送信元と宛先のアドレス，優先度という情報が入っている必要がある．そこでそれらの情報を一つにまとめる“message”というデータタイプを設定した．送信元ノードからネットワークにメッセージを送り，ネットワークでメッセージから送信先アドレスを取り出すことで指定のノードへメッセージを届けることができる．これを応用すると，通信路上で考えられる様々なエラーについての表現も可能となり，多くのモデルが作成できると考えている．ソースコード 1 に LNT の記述の抜粋を記載する．行番号 2 から send と recv というアクションでノードのプロセスの集合とネットワークのプロセスが並列合成されていることがわかる．このプログラムを実行することで各プロセスが呼び出され，LTS 図 (プロセスグラフ) が作成される．

† 信州大学大学院総合理工学研究科, Graduate School of Science and Technology, Shinshu University
†† 信州大学工学部電子情報システム工学科, Department of Electrical and Computer Engineering, Faculty of Engineering, Shinshu University

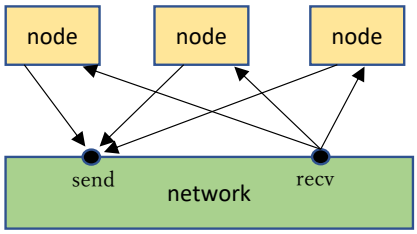

図 2　ネットワークとプロセス

ソースコード 1　LNT 記述 (抜粋)

```
process MAIN [want_access,get_access,send,recv
     ,GET:any,raise BAD_MESSAGE:none]is
  par send,recv in
    par
      node[want_access,get_access,send,recv,
           BAD_MESSAGE](No1 of addr,1 of value )
    ||
      node[want_access,get_access,send,recv,
           BAD_MESSAGE](No2 of addr,2 of value )
    ||
      node[want_access,get_access,send,recv,
           BAD_MESSAGE](No3 of addr,3 of value )
    end par
  ||
    network[GET,send,recv, BAD_MESSAGE]
  end par
end process
```

3.2　observer を用いた制限

デッドロックの検知などに比べより細かい検証を行う際，プロトコルに制限を設けるため observer を設けるという方法を実践した．例えば今回作成した優先度を用いる分散相互排除アルゴリズムの場合，論理時計値の上限を考えなければ各ノードが何度でも利用権を求め続けることができるため，公平性が無い状態遷移が存在する．そこで observer と各ノードを parallel composition を用いて同期することでプロセスの挙動を制御し，検証を行い易くした．なお，observer によって制限したモデルとしていないモデルについて分岐比較を用いて検証を行い，前者が後者に包含されていることを確認した．

4　検証

モデル検証には CADP toolbox により実装されている LTS 図の情報を出力するコマンド，観測値挙動の等価性を調べるコマンド，SEQ という状態遷移順序を表現する形式を用いる．なお，検証対象は 3 ノードのモデルであり，検証内容に応じて論理時計を実装しているモデルといないモデルを用いる．

4.1　デッドロックの有無

このモデルにおける前提として，論理時計の上限を設けない限りデッドロックが存在してはならない．そこで“bcg_info”というコマンドを用い，論理時計を実装していないモデルと実装しているモデルについて，それぞれ有無を確認した．結果として前者にはデッドロックが存在せず，後者には一つのデッドロックが存在した．これは全てのノードの論理時計値が上限に達したことによると考えられる．

4.2　全てのノードが利用権を求める

論理時計を実装するプログラムの場合，論理時計値に上限が設定されているため，ノードが一度も利用権を求めないパターンが存在する．そのため検証の対象は論理時計を実装していないプログラムに限る．さらに，有界公平でないパターンを排除するため，observer を用いて各ノードがそれぞれ一度だけ利用権を求めるよう制限したモデルを対象とする．そのモデルに対し図 3 に記載する LTS 図を用い，コマンドを用いて観測値挙動で等価であることを確認し，全てのノードが利用権を求めることが分かった．なお，“want_access”は利用権を要求するアクションであり，後に続く数はノードのアドレスである．

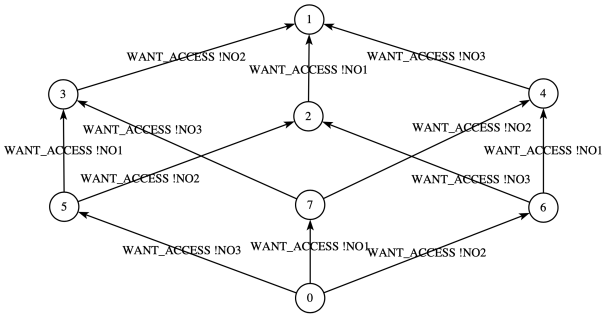

図 3　サービス仕様のプロセスグラフ

4.3　優先度の働き

状態遷移順序を表現する SEQ 記法 [3] を用いて検証したいノード番号の順序を記述することにより，生成した LTS 内にその順序があるかを調べることができる．observer で挙動を制限したモデルを対象する．論理時計を実装していないモデルについてはあらかじめノードが持っている優先度通りに動作しているか検査し，論理時計を実装しているモデルについては論理時計値通りに動作しているか検査する．今回のモデルの場合，ノード数が 3 であるため利用権を得る順序は 6 通りであり，それらを網羅的に記述した．結果としてどちらのモデルについても想定した優先度順に利用権を得ており，優先度に従わない状態遷移順序については存在していないことが分かった．

5　まとめと今後の課題

本研究では優先度を用いる分散相互排除アルゴリズムについて LNT によって記述し，検証を行った．今後の課題として，検証方法の改良が挙げられる．特により複雑なモデルを考えると，その挙動を網羅的に記述することに限界がある．そのためノード数が変化してもスケーラブルな検証方法を確立する必要がある．そこで高い記述性を有する高機能な MCL(Model Checking Language)[3] などの利用を考えていく．

参考文献

[1] 真鍋義文“情報工学レクチャーシリーズ　分散処理システム”森北出版株式会社,2013.
[2] INRIA/VASY-INRIA/CONVECS, Reference Manual of the LNT to LOTOS Translator(Version 7.2).2023. https://cadp.inria.fr/ftp/publications/cadp/Champelovier-Clerc-Garavel-et-al-10.pdf
[3] CADP http://cadp.inria.fr/
[4] Hugues Evrard “Formal Modeling of Distributed Systems”2022.http://cadp.inria.fr/ftp/presentations/Evrard-MARS-22.pdf