π 計算モデルを用いた Erlang における未監視プロセスの検出

1220337　佐々木 勝一　【 ソフトウェア検証・解析学研究室 】

1　はじめに

Erlang とは，並行指向の関数型プログラミング言語である．並行処理を実現するため，Erlang VM 上では複数の Erlang プロセスが動作する．Erlang の大きな特徴として，エラーハンドリングを主にプロセスの監視ツリーを用いて行う点が挙げられる．もしも想定外のエラーが発生した場合，エラーが発生したプロセスはそのままクラッシュする．その後，プロセスのクラッシュを検知した監視プロセスは，クラッシュしたプロセスを再起動する．Erlang ではこのようなエラーハンドリングを行うため，未監視プロセスはエラーハンドリングが行われない危険なプロセスであると捉えられる．全てのプロセスが監視されることを実行前に保証できることが望ましいが，プロセスの監視ツリーは動的に変化するため容易ではない．

そこで，本研究では，モデル検査による未監視プロセスの検出を目指し，Erlang からプロセスの監視ツリーが表現可能な π 計算への変換写像を定義する．この変換写像は，[1] で Noll らによって提案されている Erlang から π 計算への変換写像を拡張する形で定義した．本研究により，[1, 2] で解決されていなかった，ネストされたリストとタプルの意味を保った変換もまた解決された．

2　Core Erlang から π 計算への変換

Core Erlang とは，可読性と解析しやすさを両立するための Erlang の中間表現の一つである．また，π 計算とは，並行システムの振る舞いを形式的に表現するための記述体系である．

監視下にあるプロセスは「他のプロセスとリンクされているプロセス」であると定義する．よって，未監視プロセスを検査するためには，Erlang におけるリンクがモデルとして表現されなければならない．しかし，Noll らの変換写像から得られるモデルではリンクが表現されていない．そのため，変換写像の拡張に加えて，変換後の振る舞いを定義する Erlang プログラムの記述によってリンクを表現する．具体的には，リンクの生成と削除を行う組み込みの関数は，同じ意味を持つユーザ定義の関数として呼び出されるように変換する．

リンクを表現するための Erlang プログラムはネストされたリストとタプルを必要とするが，Noll らの変換写像はそれらの意味を保ったまま変換できない．[2] でそれらの意味を保った変換写像の定義が試みられているが，その定義はリストとタプルがネストされた場合に意味を保てないという問題がある．そのため，ネストされたリストとタプルの意味を保つ変換写像を新たに定義する．ここでは，リストとタプルを π 計算におけるプロセスとして扱う方針を採る．なお，提案する変換写像は，Core Erlang のシンタックスの一部でリストとタプルの出現を制限している．そのため，Core Erlang からその制限を満たす Core Erlang への補助的な変換写像もまた定義する．

[1] は Core Erlang のサブセットのみ扱っており，本研究で必要とする Erlang のいくつかの要素に対する変換が定義されていないため，それらに対する変換写像を追加で定義した．具体的には，ブール演算子，比較演算子，シーケンス，そしてパターンマッチングにおけるエイリアスパターンの変換写像を新たに定義した．

3　π 計算モデルに対するモデル検査

モデル検査器として，Mobility Workbench [3] に備えられた検査器を使用する．未監視プロセスを検査するため，未監視プロセスが存在しないことを調べる Erlang プログラムを用意し，任意のタイミングで実行されるように解析対象プログラムに組み込む．それをモデルに変換し，「未監視プロセスが見つかった」という状態に到達するかどうかを検証する．

例として作成した Erlang プログラムを実際に検証したところ，157 時間以内には検証が終わらなかった．実験環境として，OS は macOS Big Sur，CPU は 1.4 GHz Quad-Core Intel Core i5，メモリは 8 GB，そして MWB は MWB's 99 バージョン 4.137 を使用した．状態爆発を防ぎ，現実的な時間で検証できるようにすることは今後の課題である．

4　おわりに

本研究では，モデル検査による未監視プロセスの検出を目指し，Erlang をプロセスの監視ツリーが表現可能な π 計算への変換写像を定義した．この変換写像は，Noll らの Erlang から π 計算への変換写像を拡張する形で定義した．本研究により，[1, 2] で解決されていなかった，ネストされたリストとタプルの意味を保った変換もまた解決された．今後の課題として，モデルへの変換法を工夫し，現実的な時間で未監視プロセスの検出を行えるようにすることが挙げられる．

参考文献

[1] T. Noll and C. K. Roy. Modeling Erlang in the Pi-Calculus. In *Proceedings of the 2005 ACM SIGPLAN workshop on Erlang*, pages 72–77, 2005.

[2] C. K. Roy and T. Noll. Modelling Programming Languages for Concurrent and Distributed Systems in Specification Languages. RWTH Aachen University, Germany, 2004.

[3] Björn Victor. The Mobility Workbench. `https://www.it.uu.se/research/group/concurrency/mwb`. 2022-02-05 参照.