メモリモデルを考慮してモデル検査をするための SPIN 用ライブラリと反例の可視化

1215097　松元 稿如　【 プログラミング言語研究室 】

A Library for Model Checking under Weak Memory Models with SPIN and Counterexample Visualization

1215097　Kosuke MATSUMOTO　【 Programming Languages and Systems Lab. 】

1　はじめに

現代のプロセッサは，高速化のためにプログラムに記述されている順序とは異なる順序で，メモリアクセス命令の列をメモリに反映する場合がある．その結果，メモリを共有して動作する並行プログラムは，メモリアクセス命令がメモリに反映される順序（**メモリ順序**）によっては，プログラム通りの順序（**プログラム順序**）では起こり得ない実行結果になる場合がある．図 1 は，メモリ順序によっては実行結果が異なるプログラムである．図 1 のプログラム中の `Store(x, 1)` はメモリアドレス x に 1 を書き込むストア命令，`Load(y, eax)` はメモリアドレス y の値を `eax` レジスタに読み出すロード命令である．図 1 のプログラムをプログラム順序で実行した場合は `eax = 0`，`ebx = 0` の結果になることはない．しかし，スレッド A の `Load` が実行された後にスレッド B の実行が始まり，2 行目の `Load` を実行するまでに，スレッド A の 1 行目の `Store` の結果がメモリに反映されなかった場合には，スレッド B が 2 行目で x から 0 を読み出し，`eax = 0`，`ebx = 0` の結果になる．

メモリ順序は各プロセッサのメモリモデルで定義されている．そのため，プログラムを検査する際には実行するプロセッサのメモリモデルを考慮する必要がある．しかし，モデル検査器 SPIN は自動的にはメモリモデルを考慮したモデル検査を行わない．SPIN でメモリモデルを考慮したモデル検査を行うためには，ユーザがモデル中にメモリモデルに従った動作を作り込む必要がある．しかし，そのようなモデルは煩雑になりやすく，誤りを含みやすい．

そこで本研究では，メモリモデルを考慮して並行プログラムのモデル検査を行うための SPIN 用ライブラリ **MMLib** を開発した [1, 2]．本ライブラリは，複数のスレッドからアクセスされる変数を共有変数として管理し，**共有変数アクセス命令**というメモリアクセス命令の代わりのマクロを提供する．MMLib を使えば，SPIN に与えるモデル中の共有変数に対するメモリアクセスを共有変数アクセス命令を使用するように書き換える程度の手間で，ユーザは SPIN でメモリモデルを考慮したモデル検査ができる．

初期値 `x = 0; y = 0;`

スレッド A	スレッド B
`Store(x, 1);`	`Store(y, 1);`
`Load(y, eax);`	`Load(x, ebx);`

図 1: メモリ順序によっては実行結果が異なるプログラム

```
#include "pso.h"
proctype A() {
 WRITE(x, 1);
 int eax = READ(y);
}
```

```
proctype B() {
 WRITE(y, 1);
 int ebx = READ(x);
}
```

（右へ続く）

図 2: MMLib を使った図 1 のモデル

MMLib を使用した場合でも，ユーザは SPIN が出力する反例を元にデバッグを行う．しかし，反例には展開された共有変数アクセス命令や MMLib が内部で使用しているデータ構造が，プログラムのモデルと入り混じって出力される．そのため，ユーザが反例を読み解いてバグを理解するのは難しい．

そこで本研究では，MMLib を使用したモデルの反例を可視化するソフトウェアも開発した．本反例可視化ソフトウェアは，メモリ順序が直感的に理解できるように工夫している．

2　MMLib

MMLib は，シーケンシャルコンシステンシ（SC）とトータルストアオーダリング（TSO），パーシャルストアオーダリング（PSO）の 3 つのメモリモデルについて，メモリモデルを考慮して並行プログラムをモデル検査するための SPIN 用ライブラリである．ユーザは，検査に使いたいメモリモデルのライブラリを，モデル中

にインクルードして利用する．これにより，共有変数アクセス命令を利用できるようになる．

本ライブラリを使用した図 1 のプログラムのモデルを図 2 に示す．1 行目で本ライブラリをインクルードしている．proctype は SPIN のスレッドを定義するキーワードである．スレッド A とスレッド B からアクセスされる x と y は共有変数として扱うため，x に 1 を書き込みたい場合は，3 行目のように WRITE(x, 1) と記述する．また，y から値を読み出したい場合は，4 行目のように READ(y) と記述する．

3 MMLib の実装

MMLib では，共有変数アクセス命令がプログラム順序と異なる順序でメモリに反映される場合があるように，スレッドの書き込みを溜めておくストアバッファや，ストアバッファの中身を取り出してメモリに反映するメモリプロセスを導入している．各スレッドは，書き込みの際に，直接共有変数に値を書き込むのではなく，一旦ストアバッファに書き込む対象の共有変数と書き込む値の組を格納する．その後に，メモリプロセスが，非決定的にストアバッファの中身を取り出して共有変数に反映する．その結果，メモリプロセスの動作によっては，共有変数アクセス命令のプログラム順序とメモリ順序が異なる場合が再現できる．また，プログラム順序と異なるメモリ順序を許すメモリモデルのプロセッサ上でも，書き込みを実行したスレッドからは，書き込みの結果がメモリに反映される前でも自身が実行した書き込みの結果が読み出せる必要があるため，スレッド毎のメモリのコピーを導入して自身が実行した書き込みの結果を読み出せるようにしている．

4 反例可視化ソフトウェア

本研究で開発した反例可視化ソフトウェアは，反例に沿ったステップ実行の様子をモデルの上に提示する．図 3 のように，ステップ実行に合わせて緑色の矢印で各スレッドの実行位置を表示する．メモリに反映されていない共有変数アクセス命令は黄色でハイライトすることで，直感的に表現する．さらに，メモリに反映されていない共有変数アクセス命令の中でも，プログラム順序と異なる順序でメモリに反映されることが確定したり，その順序が他のスレッドから観測されたものについては，メモリモデルに関するバグの原因になりやすいため赤色でハイライトすることで強調する．また，プログラム順序と異なるメモリ順序を許すメモリモデルのプロセッサ上では，同じタイミングでもスレッドによって共有変数から読み込める値が異なるので，各スレッドがそのステップで読み込める共有変数の値の表を表示する．

5 おわりに

本研究では，メモリモデルを考慮して並行プログラムのモデル検査をするための SPIN 用ライブラリ MMLib を開発した．本ライブラリを使用することで，ユーザ

```
proctype T1() {
  WRITE(want1, 1);
  WRITE(turn, 0);
  if
  ::(READ(turn) == 1)  -> skip;
  ::(READ(want0) == 0) -> skip;
  fi;
CS: skip; /*Critical Section*/
  WRITE(want1, 0);
}
```

図 3: 反例可視化ソフトウェア

は SPIN でメモリモデルを考慮してモデル検査できる．MMLib を使って，x86-TSO リトマステスト [3] やモデル検査の教科書 [4]，文献 [5] で使われている並行プログラムのモデルを MMLib を使って書き直して実験を行った結果，MMLib が提供する共有変数アクセス命令の動作が正しいことを確認した．さらに本研究では，本ライブラリを使用したモデルの反例を可視化するソフトウェアも開発した．MMLib と本反例可視化ソフトウェアを使うことで，ユーザはプログラム順序と異なるメモリ順序を許すメモリモデルのプロセッサ上での並行プログラムの実行を容易にモデル検査してデバッグできるようになる．本反例可視化ソフトウェアを実装し，ピーターソンのアルゴリズムによる相互排除プログラムから作成したモデルの反例を可視化した結果，反例を読み解くことが容易になり，相互排除が成り立つようにプログラムをデバッグできた．

参考文献

[1] 松元 稿如, 鵜川 始陽, and 安部 達也. メモリモデルを考慮したメモリアクセス命令を提供する SPIN 用ライブラリ. ソフトウェア工学の基礎, 23:63–72, 2016.

[2] Kosuke Matsumoto, Tomoharu Ugawa, and Tatsuya Abe. Improvement of a library for model checking under weakly ordered memory model with SPIN. *Journal of Information Processing*, 26:314–326, 2018.

[3] Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-TSO (extended version). Technical Report UCAM-CL-TR-745, University of Cambridge, Computer Laboratory, 2009.

[4] 産業技術総合研究所システム検証研究センター. モデル検査 上級編 －実践のための三つの技法－. 株式会社近代科学社, 2010.

[5] Alexander Linden. *On the Verification of Programs on Relaxed Memory Models*. PhD thesis, Universite de Liege, 2013.