6J-07

計数セマフォを含むプログラムから論理制約付き項書換え系への変換

小嶋 美咲 † 西田 直樹 ‡ 酒井 正彦 §
名古屋大学 † 名古屋大学 ‡ 名古屋大学 §

1 はじめに

近年，プログラムの検証に関数型や命令型プログラミング言語の計算モデルである論理制約付き項書換え系（LCTRS）[2] を用いる研究が盛んになっている．特に，LCTRS を用いた等価性の検証はプログラムの正当性を保証する手法としてその有用性が期待されている．

先行研究 [4, 1] では、バイナリセマフォによる排他制御を含むプログラムを対象とし，そのプログラムの動作を表現する LCTRS を生成する変換を提案した．なお，対象とするプログラムを並行実行するプロセスの数は固定されている．

本稿では，先行研究の変換を計数セマフォに拡張する．変換を計数セマフォにそのまま用いた場合，セマフォを待つ順序を守らないプロセスの発生を許してしまう．そこで，そのようなプロセスが発生しないように生成する一部の書換え規則に変更を加える．

2 バイナリセマフォのモデル化

本節では，バイナリセマフォを用いて排他制御を行うプログラムを LCTRS で表現する方法 [4, 1] を説明する．

セマフォ s に対する以下の操作を扱う [3].

- `down(&s)`：資源にアクセスするときの操作で，s が保持する値が 0 でなければ 1 だけ減算する．0 の場合は，呼び出したプロセスをセマフォ s に対する待ち状態にする．
- `up(&s)`：資源を解放するときの操作で，セマフォ s に対する待ち状態のプロセスがあれば，s が保持する値はそのままで，最も早く待ち状態になったプロセスを実行可能状態にする．待ち状態のプロセスがない場合は s が保持する値を 1 だけ加算する．
- `down` と `up` はアトミックな操作であり，一連の操作を実行している間に他の操作で割り込むことはできない．

3 つ以上のプロセスが実行される場合は待ち行列を管理する必要がある．そこで，銀行などで使われるような番号札・発券機・表示板を使った順番待ちシステムを書換え規則で表し，セマフォに対する待機列を表現する．

1 つの共有資源 src（ここでは大域変数）に対するセマフォ s（`int` 型）と，m 個のプロセスがあるプログラムの環境は以下のように表せる．

$$\mathsf{cnfg}(\mathsf{p}(st_1, n_1), \ldots, \mathsf{p}(st_m, n_m), \mathsf{sem}(v_\mathtt{s}, v_d, v_t), v_{src})$$

i 番目の引数 $\mathsf{p}(st_i, n_i)$ は i 番目のプロセスを表す．st_i は i 番目のプロセスの関数の状態であり，直観的にはプログラムカウンタ値を表す．n_i は i 番目のプロセスが保持する番号札の番号で，0 の場合は待機していない状態，すなわち実行可能状態であることを表す．$m+1$ 番目の引数 $\mathsf{sem}(v_\mathtt{s}, v_d, v_t)$ はセマフォ s を表す項である．$m+2$ 番目の引数 v_{src} は src に格納されている値である．$v_\mathtt{s}$ はセマフォの値で，0 または 1 をとる．v_d（表示板）は呼び出し中の番号で，v_t（発券機）は次に発券される番号札の番号である．

セマフォが複数ある場合はセマフォごとに表示板と発券機を用意し，セマフォごとの番号札を保持できるように p の引数を増やす．

プログラム中の `down(&s);` の操作は図 1 の (a)，(b) の形式の書換え規則に変換する．`up(&s);` も同様に (c)–(e) の形式に変換する．各規則の動作は下記の通りである．

(a) セマフォ s の値が 0 でないとき，s の値を 1 だけ減算し，実行状態を次に移行する．
(b) セマフォ s の値が 0 のとき，セマフォを取得したいプロセスは番号札 t を取得し，待ち状態になる．発券機 t は次の番号札（$t+2$）を用意する．

Transforming Programs with Counting Semaphores into Logically Constrained Term Rewrite Systems
† Misaki Kojima, Nagoya University
‡ Naoki Nishida Nagoya University
§ Masahiko Sakai, Nagoya University

(a) $\mathsf{cnfg}(\ldots,\mathsf{p}(st',0),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(\ st'',0),\ldots,\mathsf{sem}(s-1,\ d,\ \ t\ \ ),src)\ [s \neq 0]$
(b) $\mathsf{cnfg}(\ldots,\mathsf{p}(st',n),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(\ st',\ t),\ldots,\mathsf{sem}(\ \ s,\ \ d,\ t+2),src)\ [s = 0 \wedge n = 0]$
(c) $\mathsf{cnfg}(\ldots,\mathsf{p}(st',n),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(\ st'',0),\ldots,\mathsf{sem}(\ \ s,\ \ d,\ \ t\ \ ),src)\ [n = d \wedge n \neq 0]$
(d) $\mathsf{cnfg}(\ldots,\mathsf{p}(st''',0),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(st'''',0),\ldots,\mathsf{sem}(\ \ s,\ d+2,\ t\ \ ),src)\ [t \neq d+2]$
(e) $\mathsf{cnfg}(\ldots,\mathsf{p}(st''',0),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(st'''',0),\ldots,\mathsf{sem}(s+1,\ d,\ \ t\ \ ),src)\ [t = d+2]$

図 1　バイナリセマフォに対する down と up の動作を表現する書換え規則

(c) $\mathsf{cnfg}(\ldots,\mathsf{p}(st',n),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(\ st'',0),\ldots,\mathsf{sem}(s,d+1,t\,),src)\ \ [n = d+1 \wedge n \neq 0]$
(d) $\mathsf{cnfg}(\ldots,\mathsf{p}(st''',0),\ldots,\mathsf{sem}(s,d,t),src) \to \mathsf{cnfg}(\ldots,\mathsf{p}(st'''',0),\ldots,\mathsf{sem}(s,d+1,t\,),src)\ \ [t \neq d+2 \wedge odd(d)\,]$

図 2　計数セマフォに対する down と up の動作を表現する書換え規則

(c) 取得している番号と呼び出し中の番号 c が一致したら実行状態（p の第 2 引数が 0）になり，実行状態を次に移行する．
(d) セマフォ s に対する待ち状態のプロセスがないとき，s の値を 1 だけ加算し，実行状態を次に移行する．
(e) セマフォ s に対する待ち状態のプロセスがあるとき，s の値を更新せずに次の番号を呼び出し（d の値に 2 を加算し），実行状態を次に移行する．

down と up の操作はアトミックに実行される操作であるため，複数個の規則で段階的に項を書換えるのではなく，一つの規則で表している．

$t = d + 2$ のとき，最後に発券した番号が呼び出し中になっているため，どのプロセスも待機していないことになる．よって，$t = d + 2$ が成り立つかどうかで待機しているプロセスがあるかどうかを判定できる．

3　計数セマフォへの拡張

本節では，計数セマフォを用いて排他制御を行うプログラムを LCTRS で表現する方法を提案する．バイナリセマフォでは，セマフォの値 s の初期値が 1 としたが，計数セマフォでは s の初期値を任意の正整数にできる．しかし，s の初期値の変更だけでは計数セマフォを LCTRS で正しく表現することはできない．

P1 から P4 まで 4 つのプロセスがあり，P1 と P2 がセマフォを取得し，P3 と P4 が待機している状況を考える．P1 がセマフォを返却すると，待機列の先頭にある P3 が実行可能状態に変わる．このとき，P3 がセマフォを取得する前に P2 がセマフォを返却すると，P4 が実行可能状態になり，P3 がセマフォを取得することができなくなる．したがって，既存の変換を計数セマフォに拡張するためには，セマフォを取得できる状態にあるプロセスがセマフォを取得し終える前に他のプロセスがセマフォを返却できないようにするような工夫が必要になる．

そこで，図 1 の規則 (d) で d の値に 2 を加算していたのを，規則 (c) と (d) で 1 ずつ加算するように変更する．d の値が偶数であるとき，規則 (d) が適用された後に規則 (c) が適用されていないことになる．つまり，セマフォを取得できる状態にあるプロセスがセマフォを取得し終える前であることを表す．この状態で再度規則 (d) が適用されないために，規則 (d) の条件として d の値が奇数であることを追加する．図 1 から変更した規則 (c)，(d) は図 2 のように表せる．

4　今後の課題

今後の課題の一つは変換の正当性を証明することである．また，割込み処理を LCTRS で表現することも検討していく．

参考文献

[1] M. Kojima, N. Nishida, and Y. Matsubara. Transforming concurrent programs with semaphores into logically constrained term rewrite systems. In *Informal Proc. WPTE 2020*, pp. 1–12, 2020.
[2] C. Kop and N. Nishida. Term rewriting with logical constraints. In *Proc. FroCoS 2013*, Vol. 8152 of *LNCS*, pp. 343–358. Springer, 2013.
[3] A. S. Tanenbaum and A. S. Woodhull. *Operating systems — design and implementation.* Pearson Education, 3 edition, 2006.
[4] 小嶋, 西田, 松原, 酒井. 排他制御を含むプログラムから論理制約付き項書換え系への変換. 信学技法 SS2019-46, 電子情報通信学会, 2020. Vol. 119, No. 451, pp. 31–36.