

## Zur Simulation von PRAM durch TM

Nachfolgend sind einige wichtige Funktionen in Pseudocode aufgeführt, die naheliegendermaßen auf einer TM implementiert werden können, deren Platzbedarf polynomiell im Zeitbedarf der PRAM ist. Für weitere, nur benutzte aber nicht definierte, Funktionen drückt der Name hoffentlich deutlich genug aus, was sie jeweils tun.

Man beachte, dass bei den diversen indirekten rekursiven Aufrufen der allen gemeinsame Parameter „aktuell betrachteter Zeitpunkt“  $t$  spätestens bei jedem zweiten Aufruf um 1 erniedrigt wird.

Die folgende Funktion *check* liefert als Rückgabewert

- *true*, falls es richtig ist, dass Prozessor  $p$  zum Zeitpunkt *start* gestartet wird und zum Zeitpunkt  $t$  Zeile  $l$  des Programms abarbeitet und der Akkumulatorinhalt danach  $a$  ist;
- *false* sonst

```
function  
bool check (proc  $p$ , time  $start$ , time  $t$ , line  $l$ , nat  $a$ )  
begin  
  if ( $t < start$ ) then return (false); fi  
  if ( $t = start$ ) then  
    if ( $p = 0$  and  $start = 0$ ) then  
       $\langle$ der einfachste Fall: es muss  $l = 0$  sein, usw. $\rangle$   
      ...  
    elsif ( $p > 0$  and  $start > 0$ ) then  
      return (findStarter( $p$ ,  $start$ ,  $l$ ,  $a$ ));  
    else  $\langle$ andere Fälle sind unmöglich $\rangle$   
      return (false);  
    fi  
  fi  
  if ( $t > start$ ) then  
     $\langle$ Informationen über den vorangegangenen Schritt ermitteln $\rangle$   
    ( $found$ ,  $l'$ ,  $a'$ )  $\leftarrow$  findLineAccu( $p$ ,  $t - 1$ ,  $start$ )  
    if not  $found$  then return (false); fi  
     $\langle$ Konsistenz des vorangegangenen Schritt mit dem aktuellen überprüfen $\rangle$   
     $c_1 \leftarrow$  checkLine( $p$ ,  $t$ ,  $l'$ ,  $a'$ ,  $l$ );  
     $c_2 \leftarrow$  checkAccu( $p$ ,  $t$ ,  $a'$ ,  $l$ ,  $a$ );  
    return ( $c_1$  and  $c_2$ );  
  fi  
end
```

Die folgende Funktion *findStarter* liefert als Rückgabewert

- *true*, falls es richtig ist, dass Prozessor *p* zum Zeitpunkt *start* in Zeile *l* des Programms gestartet wird und der Akkumulatorinhalt danach *a* ist.
- *false* sonst

```
function  
bool findStarter (proc p, time start, line l, nat a)  
begin  
  for  $p' \leftarrow p - 1$  to 0 step -1 do  
    for  $s' \leftarrow start - 1$  to 0 step -1 do  
      for  $a' \leftarrow 0$  to  $n \cdot alphSize^t$  do  
        if (checkAccu(p, start, a', l, a)) then  
          for  $l' \leftarrow$  each startLineWithLabel(l) do  
            if (check(p', start - 1, s', l', a') then  
              return (true);  
            fi  
          od  
        fi  
      od  
    od  
  od  
  return (false);  
end
```

Die folgende Funktion *findLineAccu* liefert als Rückgabewert

- $(true, l, a)$ , falls es richtig ist, dass Prozessor  $p$  zum Zeitpunkt  $start$  gestartet wird und zum Zeitpunkt  $t$  aktiv ist. Es ist dann  $l$  die zum Zeitpunkt  $t$  aktuelle Zeilennummer und  $a$  der anschließende Akkumulatorinhalt.
- $(false, 1, 0)$  sonst

```
function  
(bool,line,nat) findLineAccu (proc  $p$ , time  $t$ , time  $start$ )  
begin  
   $found \leftarrow false$ ;  
  for  $a' \leftarrow 0$  to  $n \cdot alphSize^t$  do  
    for  $l' \leftarrow 1$  to  $maxLines$  do  
       $found \leftarrow check(p, t, start, l', a')$ ;  
      if ( $found$ ) then return ( $true, l', a'$ ) fi  
    od  
  od  
  return ( $false, 1, 0$ );  
end
```

Die folgende Funktion *checkLine* liefert als Rückgabewert

- *true*, falls es richtig ist, dass Prozessor *p* zum Zeitpunkt *t* aktiv war, sich in Zeile *l'* befand, der Akkumulatorinhalt *a'* ist und die Nummer der als nächstes auszuführenden Zeile *l* ist.
- *false* sonst

```
function  
bool checkLine (proc p, time t, line l', nat a', line l)  
begin  
   $c_1 \leftarrow (\text{instrInLine}(l') \neq \text{jump}) \text{ and } (\text{instrInLine}(l') \neq \text{jzero} \text{ or } a' \neq 0)$   
    and ( $l = l' + 1$ );  
   $c_2 \leftarrow ((\text{instrInLine}(l') = \text{jump}) \text{ or } (\text{instrInLine}(l') = \text{jzero} \text{ and } a' = 0))$   
    and ( $\text{jumpLabelInLine}(l') = l$ );  
  if ( $c_1 \text{ or } c_2$ ) then  
    return (true);  
  else  
    return (false);  
  fi  
end
```

Die folgende Funktion *checkAccu* liefert als Rückgabewert

- *true*, falls es richtig ist, dass Prozessor *p* zum Zeitpunkt *t* aktiv war, sich in Zeile *l* befand, vor Ausführung des Befehls der Akkumulatorinhalt *a'* ist und nach der Ausführung *a*.
- *false* sonst

```
function
bool checkAccu (proc p, time t, nat a', line l, nat a)
begin
  case instrInLine(l) of
    jump:
      return (a' = a);
    loadDirectGlobal:
      adr ← extractAdr(l);
      for t' ← t - 1 step -1 do
        for p' ← 0 to 2t' do
          if checkGlobalWriteOccurs(p', t', adr) then goto found fi
        od
      od
    found :
      ... <usw. wie Buch, Seite 78 unten>
      <die weiteren Kombinationen von Befehlen und Adressierungsarten>
      < werden hier nicht aufgeführt>
      ...
  esac
end
```