Concurrent Computation of Binomial Coefficient

Dimitrios Yfantidis (3938)

Aristotle University of Thessaloniki,
Faculty of Sciences, Department of Informatics

January 2024

Abstract

This presentation accounts for an academic assignment on the lesson **Concurrent Programming & Software Safety**.

Abstract

This presentation accounts for an academic assignment on the lesson **Concurrent Programming & Software Safety**.

The assignment in question mainly demands the implementation of a concurrent program, as formulated by Manna & Pnueli, to compute the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} = \frac{n \cdot (n-2) \cdot \ldots \cdot (n-k+1)}{1 \cdot 2 \cdot \ldots \cdot k}$$

Assignment Prompt

- **Hypothesis:** Suppose that one procedure computes the numerator while another procedure computes the denominator.

Assignment Prompt

- **Hypothesis:** Suppose that one procedure computes the numerator while another procedure computes the denominator.
- **Tip:** Prove that $i!$ is a divisor of $j \cdot (j+1) \cdot \ldots \cdot (j+i-1)$. This way the numerator's procedure can fetch partially computed results from the denominator's procedure and performs the division immediately, so that its partially computed results don't grow too big. E.g: $1 \cdot 2$ divides $10 \cdot 9$, $1 \cdot 2 \cdot 3$ divides $10 \cdot 9 \cdot 8$, etc.

Mathematical Preliminary Work (1st Proof)

1. Consider the following numeric expression:

$$Q = \prod_{k=0}^{i-1} (j + k) = j \cdot (i + 1) \cdot \ldots \cdot (j + i - 2) \cdot (j + i - 1)$$

2. Assign $u := j + i - 1 \Rightarrow j = u - i + 1$, thus:

$$Q = (u - i + 1) \cdot (u - i + 2) \cdot \ldots \cdot (u - 1) \cdot u = \frac{u!}{(u - i)!}$$

3. Consequently:

$$\frac{1}{i!} \prod_{k=0}^{i-1} (j + k) = \frac{Q}{i!} = \frac{u!}{(u - i)!i!} = \binom{u}{i} = \binom{j + i - 1}{i} \in \mathbb{Z}$$

   Proving that $i! \mid j \cdot (j + 1) \cdot \ldots \cdot (j + i - 1)$

Mathematical Preliminary Work (2nd Proof)

1. Argument $\mathcal{I}(i, j)$:

$$i! \mid j \cdot (j+1) \cdot \ldots \cdot (j+i-1)$$

2. Suppose $f : \mathbb{N}^2 \to \mathbb{N}$ and

$$f(m, n) = m \cdot (m+1) \cdot \ldots \cdot (m+n-1) = \prod_{k=0}^{n-1} (m+k)$$

3. Argument $\mathcal{I}(i, j)$ can be written as:

$$i! \mid f(j, i)$$

Mathematical Preliminary Work (2nd Proof)

Induction by i

- Trivial case, $\mathcal{I}(1, j)$: $1! | f(j, 1) \Leftrightarrow 1 | j$ (true $\forall j \in \mathbb{N}$)
- Induction hypothesis: $i! \mid f(j, i)$
- Induction step: $i \to i + 1$

Mathematical Preliminary Work (2nd Proof)

Induction by i

- Trivial case, $\mathcal{I}(1, j)$: $1!|f(j,1) \Leftrightarrow 1|j$ (true $\forall j \in \mathbb{N}$)
- Induction hypothesis: $i! \mid f(j,i)$
- Induction step: $i \rightarrow i+1$

Induction by j (within the induction step of i)

- Trivial case, $\mathcal{I}(i+1, 0)$:

$$f(0, i+1) = 0 \cdot (0+1) \cdot \ldots \cdot (0+i) = 0, \forall i \in \mathbb{N}$$

  Thus, the argument $\mathcal{I}(i+1, 0)$ holds true as $(i+1)! \mid 0, \forall i$
- Induction hypothesis: $(i+1)! \mid f(j, i+1)$
- Induction step: $j \rightarrow j+1$

$$\begin{aligned} f(j+1, i+1) &= (j+1) \cdot \left[(j+1)+1\right] \cdot \ldots \cdot \left[(j+1)+(i+1)-1\right] \\ &= (j+1) \cdot (j+2) \cdot \ldots \cdot (j+i) \cdot (j+i+1) \\ &= (i+1) \cdot (j+1) \cdot \ldots \cdot (j+i) + j \cdot (j+1) \cdot \ldots \cdot (j+i) \\ &= (i+1) \cdot f(j+1, i) + f(j, i+1) \end{aligned}$$

Mathematical Preliminary Work (2nd Proof)

- The first term is divisible by $(i+1)!$ because of the induction hypothesis for i, thus:

$$(i+1)! \mid (i+1) \cdot f(j+1, i)$$

Mathematical Preliminary Work (2nd Proof)

- The first term is divisible by $(i+1)!$ because of the induction hypothesis for i, thus:

$$(i+1)! \mid (i+1) \cdot f(j+1, i)$$

- The second term is divisible by $(i+1)!$ because of the induction hypothesis for j, thus:

$$(i+1)! \mid f(j, i+1)$$

Mathematical Preliminary Work (2nd Proof)

- The first term is divisible by $(i+1)!$ because of the induction hypothesis for i, thus:

$$(i+1)! \mid (i+1) \cdot f(j+1, i)$$

- The second term is divisible by $(i+1)!$ because of the induction hypothesis for j, thus:

$$(i+1)! \mid f(j, i+1)$$

- As a consequence:

$$(i+1)! \mid (i+1) \cdot f(j+1, i) + f(j, i+1) = f(j+1, i+1)$$

and so the argument $\mathcal{I}(i+1, j+1)$ holds true.

Moving Forward

Lemma: We have proved that the product of n consecutive integers is divisible by $n!$.

Moving Forward

Lemma: We have proved that the product of n consecutive integers is divisible by $n!$.

Next steps (citing Manna & Pnueli):

- As mentioned in the beginning, process P_1 computes the numerator of the formula by successively multiplying into an integer variable, b, the factors n, $n-1$, ..., $n-k+1$. These factors are successively computed in variable y_1.

Moving Forward

Lemma: We have proved that the product of n consecutive integers is divisible by $n!$.

Next steps (citing Manna & Pnueli):

- As mentioned in the beginning, process P_1 computes the numerator of the formula by successively multiplying into an integer variable, b, the factors n, $n-1$, ..., $n-k+1$. These factors are successively computed in variable y_1.
- Process P_2, responsible for the denominator, successively divides b by the factors 1, 2, ..., k, using integer division. These factors are successively computed in variable y_2.

Algorithm Correctness

Citing Manna & Pnueli:

"For the algorithm to be correct, it is necessary that whenever integer division is applied it yields no remainder. We rely here on a general property of integers by which a product of m consecutive integers is evenly divisible by $m!$."

Algorithm Correctness

Citing Manna & Pnueli:

"For the algorithm to be correct, it is necessary that whenever integer division is applied it yields no remainder. We rely here on a general property of integers by which a product of m consecutive integers is evenly divisible by $m!$."

"Thus, b should be divided by y_2, which completes the stage of dividing b by $y_2!$, only when at least y_2 factors have already been multiplied into b by P_1. Since P_1 multiplies b by n, $n-1$, etc., and y_1 is greater than or equal to the value of the next factor to be multiplier, the number of factors that have been multiplied into b is at least $n - y_1$."

Algorithm Correctness

"Therefore, y_2 divides b as soon as $y_2 \leq n - y_1$, or equivalently, $y_1 + y_2 \leq n$. This condition, tested at statement m_1, ensures that b is divided by y_2 only when it is safe to do so."

Algorithm Correctness

"Therefore, y_2 divides b as soon as $y_2 \leq n - y_1$, or equivalently, $y_1 + y_2 \leq n$. This condition, tested at statement m_1, ensures that b is divided by y_2 only when it is safe to do so."

"The semaphore statements at l_1 and m_2 protect the regions $l_{2,3}$ and $m_{3,4}$ from interference. They guarantee that the value of b is not modified between its retrieval at l_2 and m_3 and its updating at l_3 and m_4."

Algorithm Formulation

Input: $k, n \in \mathbb{N}$, $0 \le k \le n$
Output: $b \in \mathbb{N}$
Locals: $y_1, y_2 \in \mathbb{N}$, mutex: r
Initialization: $y_1 := n$, $y_2 := 1$, $b := 1$

p1	p2
local t1: **integer**	**local** t2: **integer**
l_0 : **while** $y_1 > (n-k)$ **do**:	m_0 : **while** $y_2 \le k$ **do**:
l_1 : **request** r	m_1 : **await** $y_1 + y_2 \le n$
l_2 : $t_1 := b \cdot y_1$	m_2 : **request** r
l_3 : $b := t_1$	m_3 : $t_2 := b \; div \; y_2$
l_4 : **release** r	m_4 : $b := t_2$
l_5 : $y_1 := y_1 - 1$	m_5 : **release** r
l_6 : **end-while**	m_6 : $y_2 := y_2 + 1$
	m_7 : **end-while**

Challenges

One major issue is that for small values of n, P_1 would finish execution even before P_2 had managed to start.

```
(n, k) = (40, 12)
[REAL] > bincoef(40, 12) = 5586853480, bits: 33, NUM bits: 62, DNM bits: 29
[COMP] >
p1: b=40
p1: b=1560
p1: b=59280
p1: b=2193360
p1: b=78960960
p1: b=2763633600
p1: b=93963542400
p1: b=3100796899200
p1: b=99225500774400
p1: b=3075990524006400
p1: b=92279715720192000
p1: b=2676111755885568000
p2: b=2676111755885568000
p2: b=1338055877942784000
p2: b=446018625980928000
p2: b=111504656495232000
p2: b=22300931299046400
p2: b=3716821883174400
p2: b=530974554739200
p2: b=66371819342400
p2: b=7374646593600
p2: b=737464659360
p2: b=67042241760
p2: b=5586853480
bincoef(40, 12) = 5586853480
```

However it is not necessary for n to be big enough to cause overflow problems.

Challenges

Consequently, a binary semaphore was introduced to the program so that P_1 performs `wait()`, giving priority to P_2 and P_2 performs `signal()` if it was to yield a remainder by performing *div*. As a result, P_1 and P_2 seem to alternate.

```
(n, k) = (40, 12)
[REAL] > bincoef(40, 12) = 5586853480, bits: 33, NUM bits: 62, DNM bits: 29
[COMP] > p1: b=40
p2: b=40
p1: b=1560
p2: b=780
p1: b=29640
p2: b=9880
p1: b=365560
p2: b=91390
p1: b=3290040
p2: b=658008
p1: b=23030280
p2: b=3838380
p1: b=130504920
p2: b=18643560
p1: b=615237480
p2: b=76904685
p1: b=2460949920
p2: b=273438880
p1: b=8476605280
p2: b=847660528
p1: b=25429815840
p2: b=2311801440
p1: b=67042241760
p2: b=5586853480
bincoef(40, 12) = 5586853480
```

Algorithm Formulation v.2

Input: $k, n \in \mathbb{N}$, $0 \leq k \leq n$
Output: $b \in \mathbb{N}$
Locals: $y_1, y_2 \in \mathbb{N}$, mutex: r, semaphore: S
Initialization: $y_1 := n$, $y_2 := 1$, $b := 1$, S.value $:= 1$

p1	p2
local t1: **integer**	**local** t2: **integer**
l_0 : **while** $y_1 > (n-k)$ **do**:	m_0 : **while** $y_2 \leq k$ **do**:
l_1 : S.wait()	m_1 : **await** $y_1 + y_2 \leq n$, **meanwhile**:
l_2 : **request** r	m_2 : S.signal() if p_1 is blocked
l_3 : $t_1 := b \cdot y_1$	m_3 : **request** r
l_4 : $b := t_1$	m_4 : $t_2 := b \ div \ y_2$
l_5 : **release** r	m_5 : $b := t_2$
l_6 : $y_1 := y_1 - 1$	m_6 : **release** r
l_7 : **end-while**	m_7 : $y_2 := y_2 + 1$
	m_8 : **end-while**

Success

- In some cases where the numerator, or even the denominator overflows, the binomial coefficient doesn't necessarily overflow.
- The brute force algorithm would always overflow in these cases but this implemntation manages to correctly compute the binomial coefficient.

Examples

Correct result (OF in numerator):

```
(n, k) = (65, 16)
[REAL] > bincoef(65, 16) = 648045936942300, ...
   ... bits: 50, NUM bits: 94, DNM bits: 45
[COMP] > bincoef(65, 16) = 648045936942300
```

Correct result (OF in numerator & denominator):

```
(n, k) = (65, 24)
[REAL] > bincoef(65, 24) = 397370533061665800, ...
   ... bits: 59, NUM bits: 138, DNM bits: 80
[COMP] > bincoef(65, 24) = 397370533061665800
```

Examples

Incorrect result (OF in numerator & denominator):

```
(n, k) = (65, 30)
[REAL] > bincoef(65, 30) = 3009106305270645216, ...
   ... bits: 62, NUM bits: 170, DNM bits: 108
[COMP] > bincoef(65, 30) = 11639028057633424
```

Correct result (OF in numerator & denominator):

```
(n, k) = (70, 21)
[REAL] > bincoef(70, 21) = 385439532530137800, ...
   ... bits: 59, NUM bits: 124, DNM bits: 66
[COMP] > bincoef(70, 21) = 385439532530137800
```

References

1. J. M. ain't a mathematician; Nurdin Takenov
   "The product of n consecutive integers is divisible by n factorial"
2. Zohar Manna; Amir Pnueli (1995)
   "Temporal Verification of Reactive Systems: Safety"