Division Strategies - Using Commutative Reasoning

Compiled by: Theodore M. Savich

March 31, 2025

Strategy descriptions and examples adapted from Hackenberg (2025).

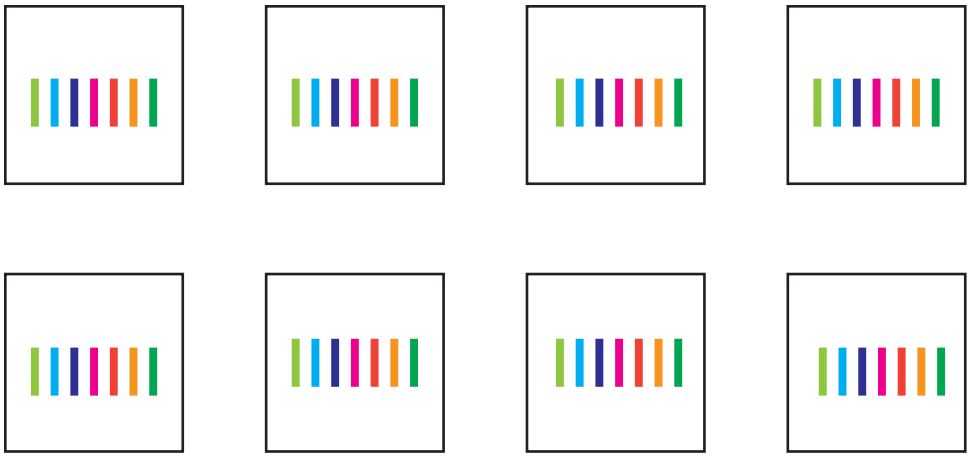
This is a strategy for transforming the context of a sharing division problem (where the number of items in each group is unknown) into one where measurement division strategies can be used. Measurement division strategies are generally easier to use because students can count something.

$$\boxed{\text{Number of groups}} \times \boxed{\text{Unknown Number of items in each group}} = \boxed{\text{Total number of items}}$$

The idea of **Using Commutative Reasoning** is to reframe N —- the number of groups - — as the act of placing one item into each group simultaneously. So, when you count one N, you're putting one item into every group; counting three Ns means each group receives three items. This new interpretation of N enables us to apply measurement division strategies, since our goal becomes finding how many times N fits into the total number of items. This method is incredibly useful—but first, we need to clarify this shift in how we view N! You might create a chart that illustrates distributing items one round at a time across the groups as you count by N. When you're learning this strategy, using such visual representations can be very beneficial. The problem remains a sharing division problem, but we can now effectively apply measurement division strategies to solve it.

Example: There are 56 cupcakes and 8 boxes. If we are going to put an equal number of cupcakes in each box, how many will go in each box?

The original meaning of 8 in the problem is # of boxes, or # of groups. The meaning Victoria gave to 8 when she wrote down eight 8s (see above) was that 8 meant the # of items in a group. Neither of these meanings for 8 would allow her to count by repeatedly by 8 until she reaches 56, and then to know she has solved the problem. In other words, neither of these meanings for 8 will allow her to count seven 8s as a meaningful solution to the problem. WHY?



Number of cupcakes given out	Number of cupcakes in each box
One 8 = 8	1
Two 8s = 16	2
Three 8s = 24	3
Four 8s = 32	4
Five 8s = 40	5
Six 8s = 48	6
Seven 8s = 56	7

Using Commutative Reasoning

Strategy Overview

Using Commutative Reasoning leverages the commutative property of multiplication to facilitate division. By repackaging the number of groups and the number of items in each group, this strategy simplifies the division process and aligns it with multiplication reasoning.

Automaton Design

We design a **Transducing Automaton** that converts the division problem into its commuted form. This automaton:

1. Reads the original group count and element count.
2. Repackages these values to form the commuted counts.
3. Performs the calculation based on the commuted structure.
4. Outputs the quotient.

Automaton Tuple

We define the automaton as the 7-tuple

$$M = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_{0/accept},\ \#,\ F)$$

where:

- $Q = \{q_{0/accept},\ q_{\text{commute}},\ q_{\text{calculate}},\ q_{\text{output}}\}$ is the set of states. (Here, $q_{0/accept}$ serves as both the start and accepting state.)
- $\Sigma = \{G, E\}$ is the input alphabet, where G represents the original group count and E represents the number of elements.
- $\Gamma = \{\#, G, E, G', E'\}$ is the stack alphabet:
  - $\#$ is the bottom-of-stack marker.
  - G and E are the original values.
  - G' and E' are the commuted values.
- $q_{0/accept}$ is the start (and accept) state.
- $\#$ is the initial stack symbol.
- $F = \{q_{0/accept}\}$ is the set of accepting states.

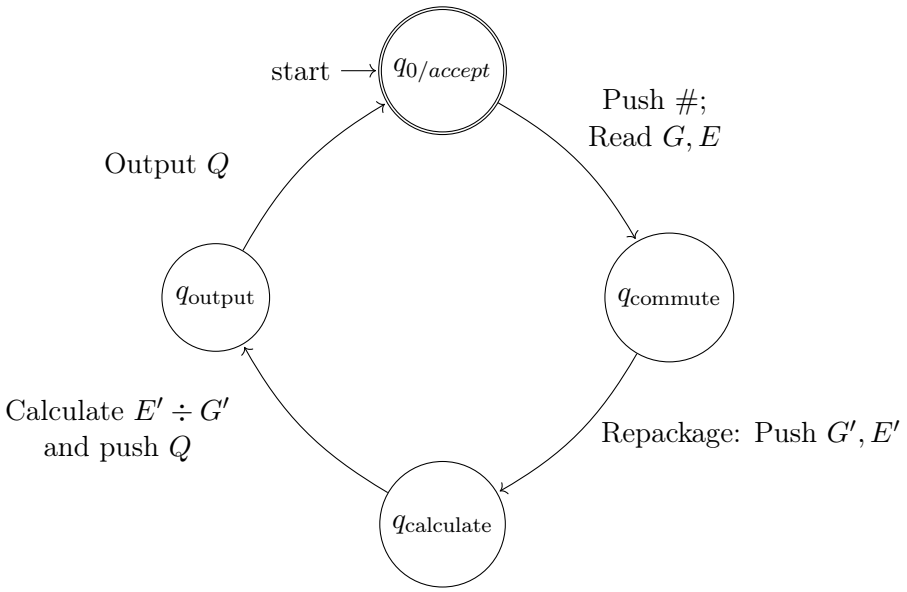
State Transition Table

Current State	Input Symbol	Stack Top	Next State	Stack Operation	Description
$q_{0/accept}$	ε	—	q_{commute}	Push #	Initialize
q_{commute}	G, E	any	q_{commute}	Repackage: Push G' and E'	Commute $G \leftrightarrow E$
q_{commute}	ε	—	$q_{\text{calculate}}$	No stack change	End commutation
$q_{\text{calculate}}$	ε	—	$q_{\text{calculate}}$	Compute quotient: $E' \div G'$	Perform calculation
$q_{\text{calculate}}$	ε	—	q_{output}	Push Q (quotient)	When calculation complete
q_{output}	ε	—	$q_{0/accept}$	Output Q	Output result and accept

Automaton Behavior

1. **Initialization:** Start in $q_{0/accept}$, push # onto the stack, then transition to q_{commute}.
2. **Commuting Groups and Elements:** In q_{commute}, read the input values G and E, repackage them to produce G' and E' (i.e. $G' = E$ and $E' = G$), and push these onto the stack.
3. **Calculating Division:** Transition to $q_{\text{calculate}}$ to compute the quotient from the commuted values (i.e. $E' \div G'$).
4. **Outputting the Result:** In q_{output}, output the computed quotient Q and then return to the merged start/accept state $q_{0/accept}$.

Circular Automaton Diagram



Example Execution

Problem: Divide 56 items into groups of 8 using commutative reasoning.

1. **Start:**
   - Input: $G = 8$ (group size) and $E = 56$ (total items).
   - Initial Stack: #
2. **Commutation:**
   - In q_{commute}, repackage G and E to obtain $G' = 56$ and $E' = 8$.
3. **Calculation:**
   - In $q_{\text{calculate}}$, compute $E' \div G' = 8 \div 56$ if interpreted as division, or (more typically) convert the division problem: since $56 \div 8 = 7$, the commuted form helps to see that there are 7 groups.
4. **Output:**
   - The quotient $Q = 7$ is output in q_{output}, meaning 7 groups of 8.

Flexible Handling of Commutation

The automaton flexibly repackages the number of groups and elements, aligning the division process with multiplication reasoning to simplify the calculation.

Conclusion

This transducing automaton models the use of commutative reasoning in division by transforming the input division problem into its commuted form. The circular diagram with the merged start/accept state reflects the iterative nature of the process, ensuring that the quotient is correctly computed and output.

HTML Implementation

References

Hackenberg, A. (2025). *Course notes* [Unpublished course notes].