Subtraction Strategies: Rounding and Adjusting

Compiled by Theodore M. Savich

March 11, 2025

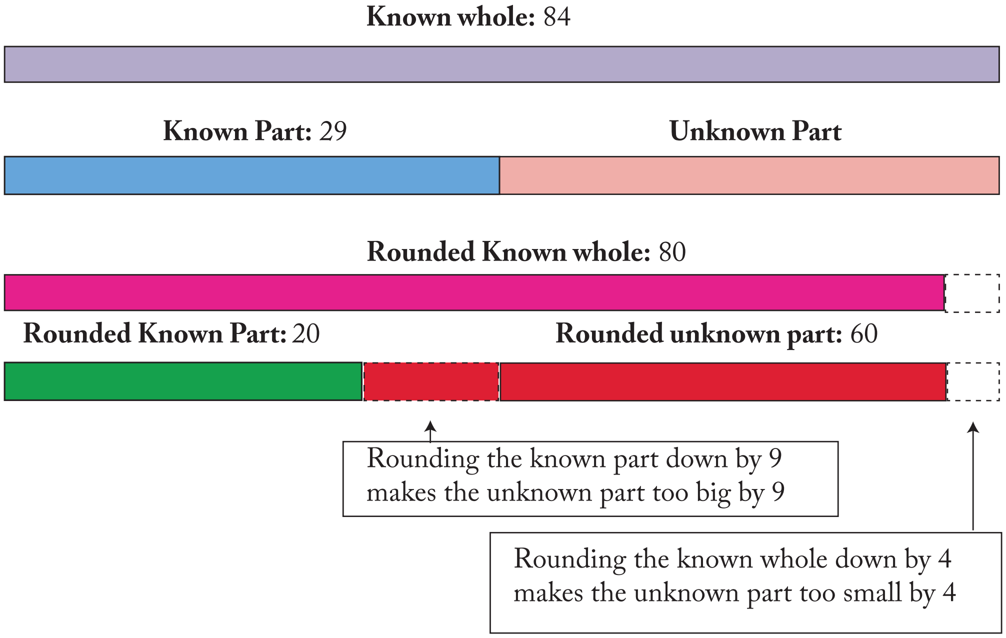
Rounding and Adjusting

Transcript

Video from Carpenter et al. (1999). Strategy descriptions and curation by Amy Hackenberg.

- **Teacher:** Kevin had 84 gumdrops. During the week, he ate 29 gumdrops. How many gumdrops does he have left?
- **Kevin:** 55.
- **Teacher:** How'd you get 55?
- **Kevin:** I knew if I had 80 gumdrops and I ate 20, I knew I would have 60 gumdrops. But I had to add 4 more because it was 84 minus 20, so that would be 64. And I took away 4 more, and that would be 60. But I had to take away 5 more and that would be 55.

Picture


Known whole: 84
Known Part: 29
Unknown Part
Rounded Known whole: 80
Rounded Known Part: 20
Rounded unknown part: 60
Rounding the known part down by 9 makes the unknown part too big by 9
Rounding the known whole down by 4 makes the unknown part too small by 4

Notation

Rounding

$$
\begin{align}
84 - 29 &= \square \tag{1}\\
84 - 4 &= 80 \tag{2}\\
29 - 9 &= 20 \tag{3}\\
80 - 20 &= 60 \tag{4}
\end{align}
$$

Adjusting

$$
\begin{align}
60 + 4 &= 64 \tag{5}\\
64 - 4 &= 60 \tag{6}\\
60 - 5 &= 55 \tag{7}
\end{align}
$$

Explaining the Adjusting

- Kevin knew that if he had 80 gumdrops and ate 20, he would have 60 gumdrops left.
- Rounding the known whole down by 4 makes the unknown part too small by 4.
- So, adjust the difference by adding 4 gumdrops back to get 64.
- Rounding the known part down by 9 makes the unknown part too big by 9
- So, adjust the difference by subtracting 9. Kevin does this by chunking back by 4 (to get 60) and then by 5 (to get 55).

Description of Strategy

Change either the known part or the known whole to a "good" number—usually the nearest base—to make the subtraction easier. Then subtract and adjust your answer. This extra adjusting step can be a bit trickier than rounding when you add!

- If you round the known whole up, you pretend you had more than you really did, so the unknown part seems too big.
- If you round the known whole down, you act like you had less, and you'll need to add back what you subtracted at the end.
- Similarly, if you round the known part down, you're not subtracting enough and must add back in.
- If you round the known part up, you subtract too much and need to add some back to fix it.

Automaton Type

Pushdown Automaton (PDA): Needed to remember the amount of adjustment required.

Formal Description of the Automaton

We define the PDA as the 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_{0/accept}, Z_0, F)$$

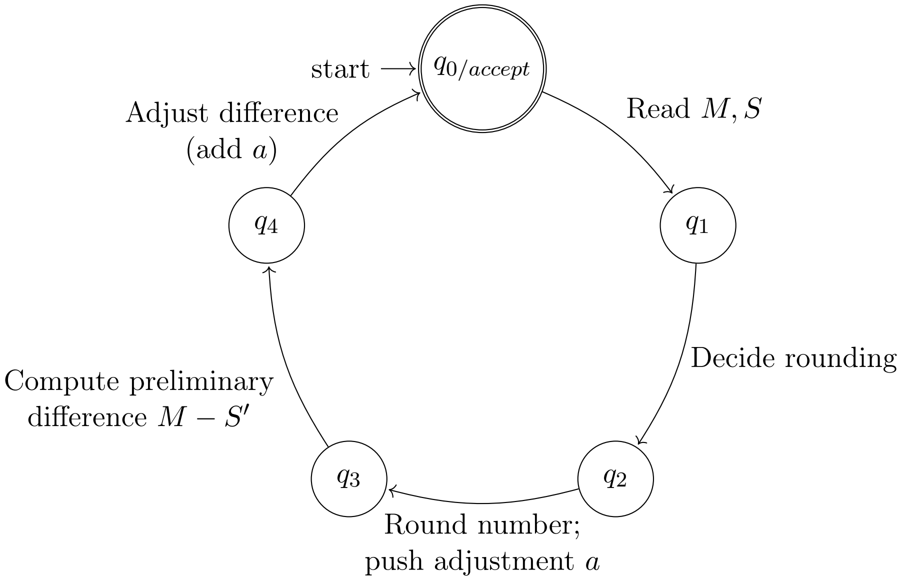
where:

- $Q = \{q_{0/accept},\, q_1,\, q_2,\, q_3,\, q_4\}$ is the set of states.
- $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is the input alphabet (representing the digits of the minuend M and subtrahend S).
- $\Gamma = \{Z_0\} \cup \{x \mid x \in \mathbb{Z}\}$ is the stack alphabet, where Z_0 is the initial stack symbol and x represents the adjustment value.
- $q_{0/accept}$ is the start state, which is also the accept state.
- $F = \{q_{0/accept}\}$ is the set of accepting states.

The transition function δ is defined by:

1. $\delta(q_{0/accept},\, \text{“}M, S\text{”},\, Z_0) = \{(q_1,\, Z_0)\}$
   (Read the minuend M and subtrahend S.)
2. $\delta(q_1,\, \varepsilon,\, Z_0) = \{(q_2,\, Z_0)\}$
   (Decide which number to round and determine the rounding strategy.)
3. $\delta(q_2,\, \varepsilon,\, Z_0) = \{(q_3,\, a\, Z_0)\}$
   (Perform the rounding. Let a be the adjustment amount where, for example, if rounding the subtrahend, $a = S' - S$.)
4. $\delta(q_3,\, \varepsilon,\, a) = \{(q_4,\, a)\}$
   (Compute the preliminary difference using the rounded value; that is, compute $M - S'$.)
5. $\delta(q_4,\, \varepsilon,\, a) = \{(q_{0/accept},\, Z_0)\}$
   (Adjust the preliminary difference by incorporating a (i.e., final difference $= (M - S') + a$) and output the result.)

Automaton Diagram for Rounding and Adjusting



JavaScript

```
document.addEventListener(’DOMContentLoaded’, function() {
    const outputElement = document.getElementById(’subRoundingOutput’);
    const minuendInput = document.getElementById(’roundSubMinuend’);
    const subtrahendInput = document.getElementById(’roundSubSubtrahend’);

    if (!outputElement) {
        console.warn("Element␣subRoundingOutput␣not␣found");
        return;
    }

    window.runSubtractionRoundingAutomaton = function() {
        try {
            const minuend = parseInt(minuendInput.value);
            const subtrahend = parseInt(subtrahendInput.value);

            if (isNaN(minuend) || isNaN(subtrahend)) {
                outputElement.textContent = "Please␣enter␣valid␣numbers␣for␣minuend␣and␣
                    subtrahend.";
                return;
            }

            let output = ’’;
            output += ‘<h2>Rounding and Adjusting Subtraction</h2>‘;
            output += ‘<p><strong>Original Problem:</strong> ${minuend} - ${subtrahend}</p
                >‘;

            // Determine rounding strategy (round subtrahend down to nearest lower
                multiple of 10)
            const roundedSubtrahend = Math.floor(subtrahend / 10) * 10;
            const adjustment = subtrahend - roundedSubtrahend;

            output += ‘<p><strong>Step 1: Round Subtrahend Down</strong></p>‘;
            output += ‘<p>Original Subtrahend: ${subtrahend}</p>‘;
            output += ‘<p>Rounded Subtrahend: ${roundedSubtrahend}</p>‘;
            output += ‘<p>Adjustment (amount subtracted): ${adjustment}</p>‘;

            // Perform subtraction with rounded subtrahend
            const intermediateResult = minuend - roundedSubtrahend;

            output += ‘<p><strong>Step 2: Subtract Rounded Subtrahend</strong></p>‘;
            output += ‘<p>${minuend} - ${roundedSubtrahend} = ${intermediateResult}</p>‘;

            // Apply adjustment
            const finalResult = intermediateResult + adjustment;

            output += ‘<p><strong>Step 3: Apply Adjustment (Add back the subtracted amount)</strong></p>‘;
            output += ‘<p>Preliminary Difference: ${intermediateResult}</p>‘;
            output += ‘<p>Adjustment to add: ${adjustment}</p>‘;
            output += ‘<p>Final Difference: ${intermediateResult} + ${adjustment} = ${finalResult}</p>‘;

            // Final result
            output += ‘<p><strong>Result: ${minuend} - ${subtrahend} = ${finalResult}</strong></p>‘;

            outputElement.innerHTML = output;
            typesetMath(); // Keep typesetMath for potential formatting

        } catch (error) {
            outputElement.textContent = ‘Error: ${error.message}‘;
        }
    };

    function typesetMath() {
        if (window.MathJax && window.MathJax.Hub) {
            MathJax.Hub.Queue(["Typeset", MathJax.Hub]);
        }
    }
});
```