CPSC-354 Report

Nathan Garcia
Chapman University

September 21, 2025

Abstract

Contents

1 Introduction

2 Week by Week

2.1 Week 1

2.1.1 MU Puzzle

The MU puzzle comes from the book *Gödel, Escher, Bach*. You start with the string **MI** and the goal is to turn it into **MU** by following four rules:

1. If a string ends with **I**, you can add a **U** at the end.
2. If a string starts with **M**, you can copy everything after the M.
3. If you see **III**, you can change it to **U**.

4. If you see **UU**, you can delete it.

The puzzle is about seeing if you can reach **MU** by only using these rules. It is not really about the letters themselves, but about how rules control what strings you can or cannot make.

2.1.2 HW1

The MU puzzle comes from the book *Gödel, Escher, Bach*. You start with the string **MI** and the goal is to turn it into **MU** by following four rules:

1. If a string ends with **I**, you can add a **U** at the end.
2. If a string starts with **M**, you can copy everything after the M.
3. If you see **III**, you can change it to **U**.
4. If you see **UU**, you can delete it.

At first, I tried small derivations. For example:

$$\texttt{MI} \Rightarrow \texttt{MIU} \Rightarrow \texttt{MIUIU}$$

or duplicating I's:

$$\texttt{MI} \Rightarrow \texttt{MII} \Rightarrow \texttt{MIIII}$$

From `MIIII`, I can replace `III` with `U`, giving `MUI`, but not `MU`. Every time, an extra `I` is left over, and there is no rule that deletes a single `I`.

Invariant Argument. Let $\#I(w)$ denote the number of `I`'s in string w. If we track $\#I(w) \pmod 3$, we find:

- Rule 1: $\#I$ unchanged.
- Rule 2: $\#I$ doubles. Over $\mathbb{Z}/3\mathbb{Z}$, $1 \mapsto 2$, $2 \mapsto 1$, never 0.
- Rule 3: Removes 3 I's, leaving the remainder mod 3 unchanged.
- Rule 4: Deletes U's only, so $\#I$ unchanged.

We start with `MI`, which has $\#I = 1$. This is congruent to $1 \pmod 3$. Because no rule ever makes $\#I \equiv 0 \pmod 3$, it is impossible to reach a string with $\#I = 0$.

Conclusion. The target `MU` has $\#I = 0$, which is divisible by 3. Since that is unreachable from `MI`, the puzzle is unsolvable. As a student, the cool part here is that the solution isn't about brute-force trying rules—it's about spotting a hidden invariant (the number of I's mod 3) that blocks the path completely.

2.2 Week 2

2.2.1 HW

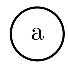
Problem. Consider the following list of Abstract Rewriting Systems (ARSs).

1. $A = \varnothing$.
2. $A = \{a\}$ and $R = \varnothing$.
3. $A = \{a\}$ and $R = \{(a, a)\}$.
4. $A = \{a, b, c\}$ and $R = \{(a, b), (a, c)\}$.
5. $A = \{a, b\}$ and $R = \{(a, a), (a, b)\}$.

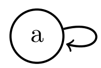
6. $A = \{a, b, c\}$ and $R = \{(a, b), (b, b), (a, c)\}$.
7. $A = \{a, b, c\}$ and $R = \{(a, b), (b, b), (a, c), (c, c)\}$.

Task. Draw a picture for each ARS above (nodes = elements of A, arrows = pairs in R). Then determine whether each ARS is *terminating*, *confluent*, and whether it has *unique normal forms*.

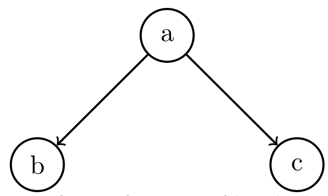
ARS 1: $A = \varnothing$ (no elements to draw)



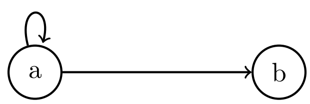
ARS 2: $A = \{a\},\ R = \varnothing$



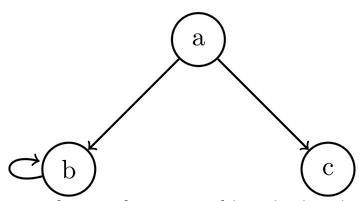
ARS 3: $A = \{a\},\ R = \{(a, a)\}$



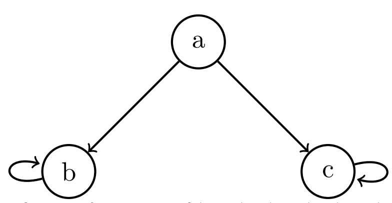
ARS 4: $A = \{a, b, c\},\ R = \{(a, b), (a, c)\}$



ARS 5: $A = \{a, b\},\ R = \{(a, a), (a, b)\}$



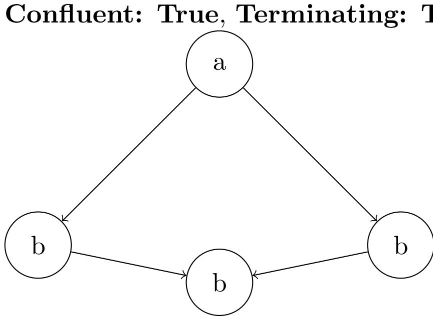
ARS 6: $A = \{a, b, c\},\ R = \{(a, b), (b, b), (a, c)\}$



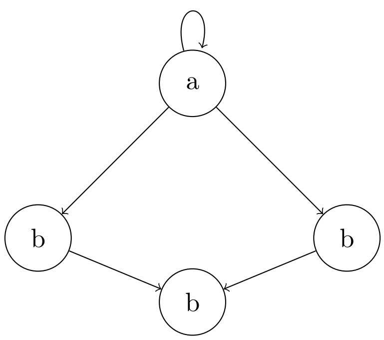
ARS 7: $A = \{a, b, c\},\ R = \{(a, b), (b, b), (a, c), (c, c)\}$

ARS	Terminating	Confluent	Has Unique Normal Forms
1	X	X	X
2	X	X	X
3		X	
4	X		X
5		X	X
6			
7			

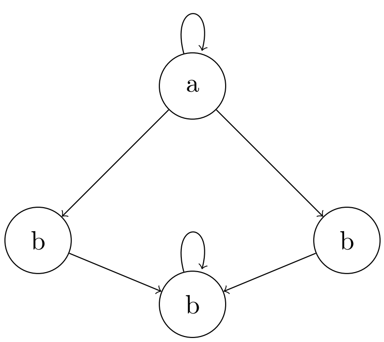
1. **Confluent: True**, **Terminating: True**, **Unique Normal Forms: False**



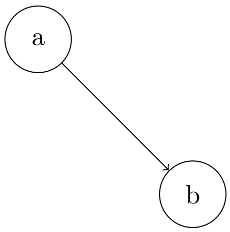
2. **Confluent: True**, **Terminating: False**, **Unique Normal Forms: True**



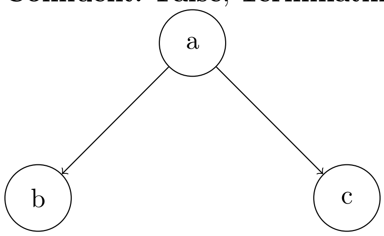
3. **Confluent: True**, **Terminating: False**, **Unique Normal Forms: False**



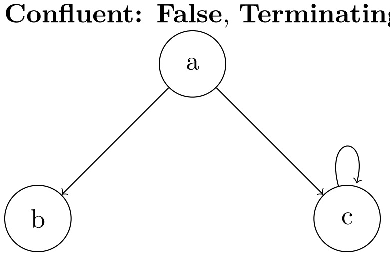
4. **Confluent: False**, **Terminating: True**, **Unique Normal Forms: True**



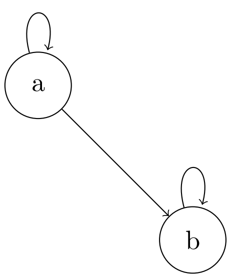
5. **Confluent: False**, **Terminating: True**, **Unique Normal Forms: False**



6. **Confluent: False**, **Terminating: False**, **Unique Normal Forms: True**



7. **Confluent: False**, **Terminating: False**, **Unique Normal Forms: False**



2.3 HW 3

2.3.1 Exercise 5

Exercise 5.

Rules:

$$ab \to ba, \quad ba \to ab, \quad aa \to \epsilon, \quad b \to \epsilon$$

Sample reductions:

$$abba \;\to\; bbaa \;\to\; baa \;\to\; aa \;\to\; \epsilon$$

$$bababa \;\to\; aaabbb \;\to\; aabbb \;\to\; abbb \;\to\; a$$

Non-termination: The rules $ab \to ba$ and $ba \to ab$ form an infinite loop:

$$ab \to ba \to ab \to ba \to \cdots$$

Non-equivalent strings: a and ϵ are not equivalent, since a single a cannot be eliminated.

Equivalence classes: Order does not matter (due to swapping). b's vanish. $aa \to \epsilon$ ensures only the *parity* of the number of a's matters.

$$I(w) = \#a(w) \bmod 2 \in \{0, 1\}$$

Thus there are exactly two equivalence classes:

$$\{w \mid \#a(w) \equiv 0 \pmod 2\} \quad \longmapsto \quad \epsilon$$

$$\{w \mid \#a(w) \equiv 1 \pmod 2\} \quad \longmapsto \quad a$$

Modified terminating system:

$$ab \to ba, \quad aa \to \epsilon, \quad b \to \epsilon$$

Termination follows from length and inversion-count measures.

Specification: The algorithm computes the *parity of the number of a's*, ignoring b's.

Exercise 5b.

Rules:

$$ab \to ba, \quad ba \to ab, \quad aa \to a, \quad b \to \epsilon$$

Sample reductions:

$$abba \;\to\; bbaa \;\to\; baa \;\to\; aa \;\to\; a$$

$$bababa \;\to\; aaabbb \;\to\; aabbb \;\to\; abbb \;\to\; a$$

Non-termination: As before, infinite swapping is possible.

Non-equivalent strings: ϵ and a are not equivalent: all b's vanish, and any positive number of a's reduces to a.

Equivalence classes: Order does not matter. b's vanish. $aa \to a$ collapses any positive number of a's to a single a.

$$J(w) = \begin{cases} 0 & \text{if } \#a(w) = 0 \\ 1 & \text{if } \#a(w) \geq 1 \end{cases}$$

Thus there are exactly two equivalence classes:

$$\{w \mid \#a(w) = 0\} \quad \longmapsto \quad \epsilon$$

$$\{w \mid \#a(w) \geq 1\} \quad \longmapsto \quad a$$

Modified terminating system:

$$ab \to ba, \quad aa \to a, \quad b \to \epsilon$$

This terminates and yields unique normal forms.

Specification: The algorithm computes whether the input contains at least one a, ignoring all b's.

2.4 Week 4

2.4.1 hw4

HW 4, PL 2025, Termination

For the definition of a *measure function*, see our notes on rewriting and, in particular, on termination.

HW 4.1. Consider the following algorithm (Euclid's algorithm for the greatest common divisor):

```
while b != 0:
    temp = b
    b = a mod b
    a = temp
return a
```

Conditions. Assume inputs $a, b \in \mathbb{N}$ with $b \geq 0$ and the usual remainder operation, i.e. for $b > 0$ we have $0 \leq a \bmod b < b$. (If $b = 0$, the loop is skipped and the algorithm terminates immediately.)

Measure function. Define

$$\mu(a, b) := b \in \mathbb{N}.$$

Proof of termination. If the loop guard holds ($b \neq 0$), one iteration maps the state (a, b) to

$$(a', b') = (\, b,\ a \bmod b\,).$$

By the property of the remainder,

$$0 \leq b' = a \bmod b < b = \mu(a, b).$$

Thus μ strictly decreases on every loop iteration and is bounded below by 0. Since $(\mathbb{N}, <)$ is well-founded, no infinite descending chain

$$\mu(a_0, b_0) > \mu(a_1, b_1) > \mu(a_2, b_2) > \cdots$$

exists. Hence only finitely many iterations are possible; the loop terminates and the algorithm halts. □

HW 4.2. Consider the following fragment of merge sort:

```
function merge_sort(arr, left, right):
    if left >= right:
        return
    mid = (left + right) / 2
    merge_sort(arr, left, mid)
    merge_sort(arr, mid+1, right)
    merge(arr, left, mid, right)
```

Define

$$\phi(left, right) := right - left + 1.$$

Claim. ϕ is a measure function for `merge_sort`.

Proof.

- *Well-defined, nonnegative.* For valid indices with $left \leq right$, we have $\phi(left, right) \in \mathbb{N}$ and $\phi \geq 1$. If $left > right$ the function is not called (or $\phi \leq 0$, and the base case applies immediately).
- *Base case.* When $left \geq right$, the function returns immediately; in this case $\phi(left, right) \leq 1$, i.e. there is no further recursion.

- *Strict decrease on recursive calls.* Suppose $left < right$ and let $n := \phi(left, right) = right - left + 1 \geq 2$. With $mid = \lfloor (left + right)/2 \rfloor$:

$$\phi(left, mid) = mid - left + 1 \leq \left\lfloor \frac{n}{2} \right\rfloor < n,$$

$$\phi(mid + 1, right) = right - (mid + 1) + 1 = right - mid \leq \left\lceil \frac{n}{2} \right\rceil < n.$$

  Thus both recursive calls strictly reduce the measure.

Since ϕ maps each call to a natural number that strictly decreases along every recursive edge and is bounded below, there are no infinite descending chains. By well-founded induction on ϕ, all recursive calls terminate.

□

3 Essay

4 Evidence of Participation

5 Conclusion

References

[BLA] Author, Title, Publisher, Year.