CPSC 354 — Report

Ray Hettleman rhettleman@chapman.edu

September 21, 2025

Contents

1 Assignment 1: MIU System — Can we derive `MIII` from `MI`?

Problem (restated)

Starting from `MI`, and using the MIU rules:

I. If a string ends with `I`, you may append `U`.

II. From `Mx` you may infer `Mxx`.

III. Replace any occurrence of `III` by `U`.

IV. Delete any occurrence of `UU`.

Determine whether `MIII` is derivable. Explain your reasoning.

Solution

Idea. Track the count of `I`'s. Rule I and Rule IV do not change that count. Rule II doubles it; Rule III removes 3 but can only apply if 3 consecutive `I`'s already exist.

Starting from `MI` (one `I`), the only growth is doubling. So after the first step the number of `I`'s is always even. Removing triples never gives exactly three. So `MIII` cannot appear.

Conclusion

It is **impossible** to derive `MIII` from `MI`. Reason: the number of `I`'s is never equal to three under these rules.

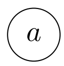
2 Assignment 2: ARS Pictures & Properties

Task (restated)

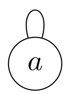
For each given abstract reduction system (ARS) (A, R): draw the graph, decide whether it is terminating, confluent, and whether it has unique normal forms (UNFs).

Instances

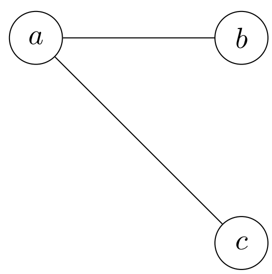
1. $A = \varnothing$ No nodes/edges. *Terminating:* True. *Confluent:* True. *UNFs:* True.



2. $A = \{a\},\ R = \varnothing$
Normal forms: a. Terminating: True. Confluent: True. UNFs: True.

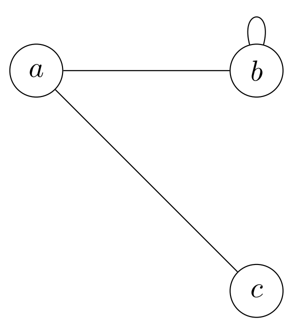


3. $A = \{a\},\ R = \{(a, a)\}$
Infinite $a \to a \to \cdots$: Terminating: **False**. Confluent: **True**. UNFs: **False**.

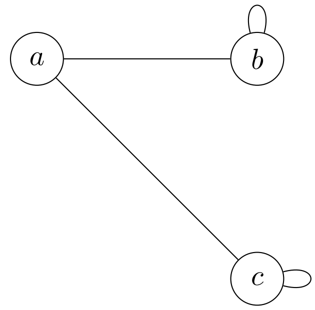


4. $A = \{a, b, c\},\ R = \{(a, b), (a, c)\}$
b, c are normal; from a two distinct NFs. Terminating: **True**. Confluent: **False**. UNFs: **False**.

5. $A = \{a, b\},\ R = \{(a, a), (a, b)\}$
b is normal; every element has the (unique) NF b. Terminating: **False**. Confluent: **True**. UNFs: **True**.



6. $A = \{a, b, c\},\ R = \{(a, b), (b, b), (a, c)\}$
Non-terminating via $b \to b$; c is normal, b has no NF. Confluent: **False**. Terminating: **False**. UNFs: **False**.



7. $A = \{a, b, c\}$, $R = \{(a, b), (b, b), (a, c), (c, c)\}$
Both b and c loop; no NFs reachable from a. Terminating: **False**. Confluent: **False**. UNFs: **False**.

Summary Table

#	(A, R)	confluent	terminating	unique normal forms
1	$(\varnothing, \varnothing)$	True	True	True
2	$(\{a\}, \varnothing)$	True	True	True
3	$(\{a\}, \{(a, a)\})$	True	False	False
4	$(\{a, b, c\}, \{(a, b), (a, c)\})$	False	True	False
5	$(\{a, b\}, \{(a, a), (a, b)\})$	True	False	True
6	$(\{a, b, c\}, \{(a, b), (b, b), (a, c)\})$	False	False	False
7	$(\{a, b, c\}, \{(a, b), (b, b), (a, c), (c, c)\})$	False	False	False

All 8 combinations

confluent	terminating	unique NFs	example
True	True	True	e.g. ARS 2 (or 1)
True	True	False	*Impossible*
True	False	True	ARS 5
True	False	False	ARS 3
False	True	True	*Impossible*
False	True	False	ARS 4
False	False	True	*Impossible*
False	False	False	ARS 6 (or 7)

3 Assignment 3: Exercises 5 and 5b

Problem (restated)

Exercise 5 asks us to analyse a given ARS and check for termination, confluence, and unique normal forms. Exercise 5b asks us to consider a small variation and explain the difference.

Solution to Exercise 5

For the ARS with $A = \{a, b\}$ and rules $a \to a$, $a \to b$:

- There is a loop $a \to a$, so the system is **not terminating**.

- From a we can still reach b, and b is normal. Every path from a eventually has the option to reach b, and once at b no rules apply.
- Thus the ARS is **confluent**: all reductions can be joined at b.
- Normal forms are unique: everything reduces to b.

Solution to Exercise 5b

Now suppose we add c with $a \to c$ and $c \to c$:

- Again, a can reduce to both b and c.
- But c loops forever and never reaches b.
- That means the system is no longer confluent: starting from a you can end in either the normal form b or the non-terminating loop on c.
- Unique normal forms are therefore lost.

Conclusion

Exercise 5 shows a non-terminating but confluent system (everything has the unique NF b). Exercise 5b shows that adding another looping branch breaks confluence, since from a different outcomes are possible.

4 Assignment 4: Termination

4.1 HW 4.1

Consider the following algorithm:

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

Conditions. Termination is guaranteed if a, b are nonnegative integers and $b > 0$ initially.

Measure function. Define $\varphi(a, b) = b$. At each step b is replaced by $a \bmod b$, which is strictly smaller than b but always nonnegative. Therefore φ decreases with every iteration.

Conclusion. Since $\varphi(a, b)$ is a nonnegative integer that strictly decreases, the loop must terminate. The algorithm is in fact Euclid's algorithm for $\gcd(a, b)$.

4.2 HW 4.2

Consider the following fragment of an implementation 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)
```

Measure function. Define $\varphi(left, right) = right - left + 1$, the length of the current subarray.

Proof. Each recursive call works on a subarray that is strictly smaller than the current one:

- Initially, $\varphi(left, right) = n$ (array length).
- The recursive calls split the array into halves, so each subproblem has measure about $n/2$, strictly less than n.
- The base case $left \geq right$ gives $\varphi(left, right) \leq 1$, so recursion stops.

Conclusion. Because the measure is a nonnegative integer that strictly decreases with each recursive call, `merge_sort` always terminates.