CPSC 354 — Report

Ray Hettleman rhettleman@chapman.edu

September 28, 2025

Contents

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

2 Assignment 2: ARS Pictures & Properties 2

3 Assignment 3: Exercises 5 and 5b 3

4 Assignment 4: Termination 4
4.1 HW 4.1 4
4.2 HW 4.2 5

5 Assignment 5: Lambda Calculus Workout 5

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

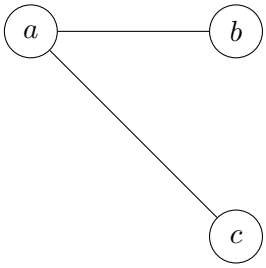
1. $A = \varnothing$ No nodes/edges. *Terminating:* True. *Confluent:* True. *UNFs:* True.

2. $A = \{a\},\ R = \varnothing$ 

Normal forms: a. Terminating: True. Confluent: True. UNFs: True.

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

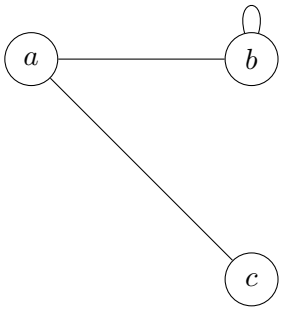
Infinite $a \to a \to \cdots$: Terminating: **False**. Confluent: **True**. UNFs: **False**.



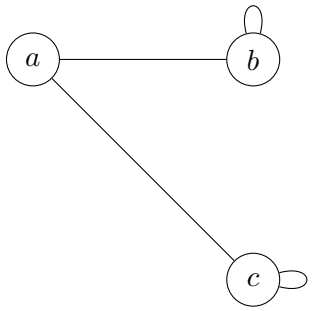
4. $A = \{a, b, c\},\ R = \{(a, b), (a, c)\}$
b, c are normal; from a two distinct NFs. Terminating: **True**. Confluent: **False**. UNFs: **False**.

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 Exercises

HW 4.1

Consider the algorithm:

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

This is the Euclidean algorithm for gcd. It terminates whenever $b > 0$, since the measure function $m(a, b) = b$ decreases strictly at each step and never goes negative.

HW 4.2

Consider 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)
```

We can take as measure function

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

This is the size of the interval being sorted. Each recursive call splits the interval into smaller subintervals, so the measure decreases until it reaches 1, at which point the recursion stops.

5 Assignment 5: Lambda Calculus Workout

Problem (restated)

The workout asks us to evaluate the following λ-expression step by step:

$$(\,\lambda f.\,\lambda x.\,f(f(x))\,)\,(\,\lambda f.\,\lambda x.\,f(f(f(x)))\,).$$

This is an exercise in applying the rules of α- and β-reduction, together with the conventions on parentheses described in lecture.

Solution

Step 1: Parenthesize clearly. Application associates to the left, while abstraction extends as far right as possible. So the above expression is shorthand for

$$((\lambda f.\,\lambda x.\,f(f(x)))\,(\lambda f.\,\lambda x.\,f(f(f(x))))).$$

The outermost redex is the application of the first abstraction to the second.

Step 2: First β-reduction. The β-rule says

$$(\lambda v.\,M)\,N \;\;\mapsto\;\; M[N/v],$$

i.e. replace the formal parameter v in M by the argument N. Here the parameter is f, and $M = \lambda x.\,f(f(x))$, $N = \lambda f.\,\lambda x.\,f(f(f(x)))$.

So:

$$(\lambda f.\,\lambda x.\,f(f(x)))\,(\lambda f.\,\lambda x.\,f(f(f(x))))\;\;\mapsto\;\;\lambda x.\,(\,(\lambda f.\,\lambda x.\,f(f(f(x))))(\,(\lambda f.\,\lambda x.\,f(f(f(x))))(x))\,).$$

Step 3: Substitution details. Notice how the argument N has been substituted everywhere f appeared in M. No variable clashes occur here, but in more complex cases we may need α-conversion (e.g. renaming bound variables) to avoid capture.

Step 4: Simplify inner application. Let us reduce inside the body:

$$(\lambda f.\,\lambda x.\,f(f(f(x))))(x).$$

To avoid confusion between bound and free variables, we first rename the bound x in the abstraction using α-conversion:

$$\lambda f.\,\lambda y.\,f(f(f(y))).$$

Now substitution is safe:

$$(\lambda f.\,\lambda y.\,f(f(f(y))))(x)\;\mapsto\;\lambda y.\,x(x(x(y))).$$

Step 5: Continue reduction. Substituting back into the outer expression:

$$\lambda x.\,(\,(\lambda f.\,\lambda y.\,f(f(f(y))))(\lambda y.\,x(x(x(y))))\,).$$

Again apply β-reduction:

$$\mapsto \lambda x.\,\lambda y.\,(\lambda y.\,x(x(x(y))))(\lambda y.\,x(x(x(y))))(\lambda y.\,x(x(x(y)))).$$

Step 6: Observe the pattern. At this point, we can already see the structure: the outer function doubled the "triple application" into a "ninefold application." More reductions would expand this even further, but the essential insight is that

$$(\,\lambda f.\,\lambda x.\,f(f(x))\,)\,(\,\lambda f.\,\lambda x.\,f(f(f(x)))\,)\;\mapsto\;\lambda x.\,\lambda y.\,x^9(y),$$

where $x^9(y)$ means x applied to y nine times.

Conclusion

This workout demonstrates several important skills:

- Using the β-rule correctly with careful substitution.
- Applying α-conversion to avoid variable capture.
- Following the precedence rules for parentheses.
- Recognizing patterns in repeated function application.

The final result shows how nested function composition quickly explodes in size, illustrating the expressive power of λ-calculus.