# CPSC-354 Report

Ray Hettleman
Chapman University
`rhettleman@chapman.edu`

September 2, 2025

**Abstract**

This report collects my weekly notes, homework, and reflections for CPSC-354.

# Contents

# 1    Introduction

This section intentionally left blank for now.

# 2    Week by Week

## 2.1    Week 1

### 2.1.1    Notes and Exploration

We studied the MIU system from Hofstadter's *Gödel, Escher, Bach*. The task was to decide whether the string `MIII` can be derived from `MI` using the system's rules.

### 2.1.2    Homework

**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`.

   **Reasoning.** We begin with `MI`. Rule I allows `MIU`. Rule II doubles the sequence after M: `MI` $\Rightarrow$ `MII`, then `MIIII`, etc. Rule III can only replace consecutive `III` with a `U`. But because doubling produces powers of two `I`'s (1, 2, 4, 8, ...), we never get exactly three `I`'s. Thus, no sequence of rules produces `MIII`.
   **Conclusion.** It is impossible to derive `MIII` from `MI`. The parity of the number of `I`'s (always even after the first doubling) prevents reaching 3.

### 2.1.3    Questions

*What is a rule that could be implemented that, while still requiring many steps, makes the MU-Puzzle solvable?*

## 2.2    Week 2

### 2.2.1    Notes and Exploration

We explored Abstract Reduction Systems (ARS), focusing on termination, confluence, and unique normal forms (UNFs). Each ARS was represented as a graph, and we determined its key properties.

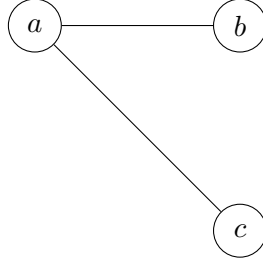### 2.2.2    Homework

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

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

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



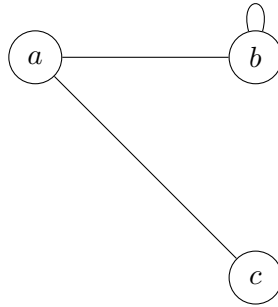Infinite sequence $a \to a \to \cdots$. Terminating: False. Confluent: True. UNFs: False.



**4.** $A = \{a, b, c\}$, $R = \{(a, b), (a, c)\}$

$b$ and $c$ are normal forms, so from $a$ two distinct endpoints are reachable. Terminating: True. Confluent: False. UNFs: False.



**5.** $A = \{a, b\}$, $R = \{(a, a), (a, b)\}$

$b$ is normal; all paths from $a$ can reach $b$. Terminating: False. Confluent: True. UNFs: True.



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

Non-terminating through $b \to b$; $c$ is normal but unreachable from $b$. Terminating: False. Confluent: False. UNFs: False.
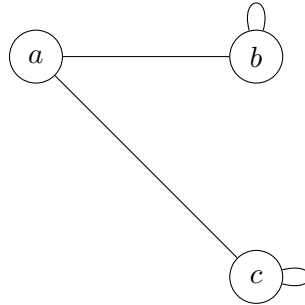


**7.** $A = \{a, b, c\}$, $R = \{(a, b), (b, b), (a, c), (c, c)\}$

Both $b$ and $c$ loop indefinitely. Terminating: False. Confluent: False. UNFs: False.

**Summary Table:**

| # | $(A, R)$ | confluent | terminating | unique NFs |
|---|---|---|---|---|
| 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 | 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) |

### 2.2.3 Questions

*Is there an easy way to tell from the graph if an ARS will terminate, or do you always have to trace every path?*

## 2.3 Week 3

### 2.3.1 Notes and Exploration

Exercises 5 and 5b extended ARS analysis with additional looping rules.

### 2.3.2 Homework

**Exercise 5.** $A = \{a, b\}$ with $a \to a$, $a \to b$. Loop $a \to a$ means not terminating, but since all paths eventually lead to $b$, the system is confluent with a unique NF.

**Exercise 5b.** Add $c$ with $a \to c$, $c \to c$. Now one branch terminates ($b$) and the other loops, breaking confluence.

### 2.3.3 Questions

*Could we make a version of 5b that still loops but somehow keeps unique normal forms?*

## 2.4 Week 4

### 2.4.1 Notes and Exploration

We introduced termination proofs and measure functions.

### 2.4.2 Homework

**HW 4.1 (Euclidean Algorithm).**

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

This algorithm terminates when $b > 0$ because $b$ decreases with each iteration. The measure function $m(a, b) = b$ is always non-negative and strictly decreases.

**HW 4.2 (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)
```

The measure $\varphi(left, right) = right - left + 1$ decreases with every recursive call. When it reaches 1, the recursion stops, proving termination.

### 2.4.3 Questions

*What would it look like to design a sorting algorithm that checks to see if an input is terminating?*

## 2.5 Week 5

### 2.5.1 Notes and Exploration

We practiced $\lambda$-calculus reduction and substitution.

### 2.5.2 Homework

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

**Step 1.** Parentheses group left:
$$((\lambda f.\lambda x.f(f(x)))(\lambda f.\lambda x.f(f(f(x))))).$$

**Step 2.** Apply $\beta$-reduction:
$$\mapsto \lambda x.(\lambda f.\lambda x.f(f(f(x))))((\lambda f.\lambda x.f(f(f(x))))x).$$

**Step 3.** Rename inner $x \to y$ and reduce:
$$(\lambda f.\lambda y.f(f(f(y))))x \mapsto \lambda y.x(x(x(y))).$$

**Step 4.** Apply again:
$$\lambda x.\lambda y.x^9(y).$$

**Conclusion.** The function composes $f$ twice and $f^3$ thrice, resulting in a ninefold application.

### 2.5.3 Questions

*What would happen if we swapped the two functions in the workout?*

## 2.6 Week 6

### 2.6.1 Notes and Exploration

We computed factorial using the fixed-point combinator `fix`.

### 2.6.2 Homework

Compute:

```
let rec fact = \n.  if n=0 then 1 else n * fact (n-1) in 3
```

Using the rules:

```
fix F ↦ F (fix F),  let rec f = e1 in e2 ↦ let f = (fix (\f.e1)) in e2.
```

We unfold recursively:

$$\text{fact 3} \mapsto \text{3 * (fix F) 2} \mapsto \text{3 * 2 * 1 = 6.}$$

**Conclusion.** Following only the allowed computation rules, `fact 3` reduces to **6**.

### 2.6.3 Questions

*Would using $\alpha$-conversion anywhere in the factorial example change the result, or just make it cleaner?*

## 2.7 Week 7

### 2.7.1 Notes and Exploration

Parsing and context-free grammars. We use the grammar (nonterminals `Exp`, `Exp1`, `Exp2`):

```
Exp  -> Exp '+' Exp1
Exp1 -> Exp1 '*' Exp2
Exp2 -> Integer
Exp2 -> '(' Exp ')'
Exp  -> Exp1
Exp1 -> Exp2
```
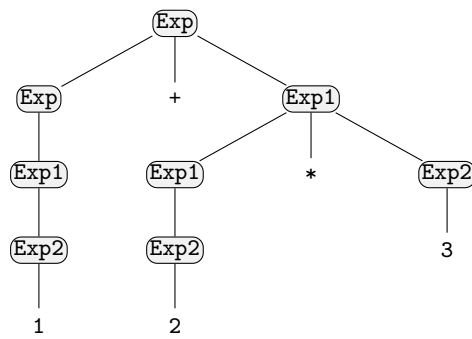
### 2.7.2 Homework: Parse Trees

Below are derivation trees (concrete syntax trees) for the required expressions. Nonterminals are boxed; terminals are leaves.
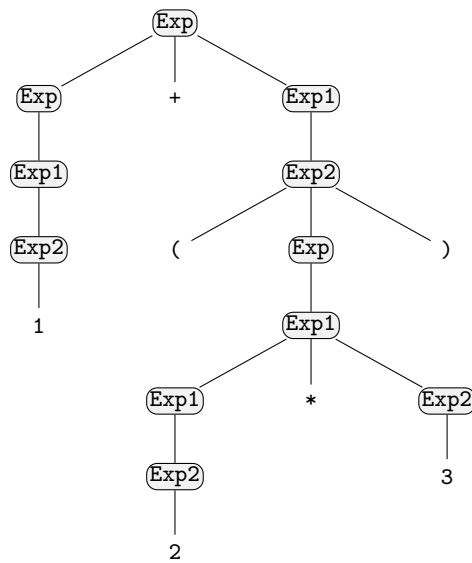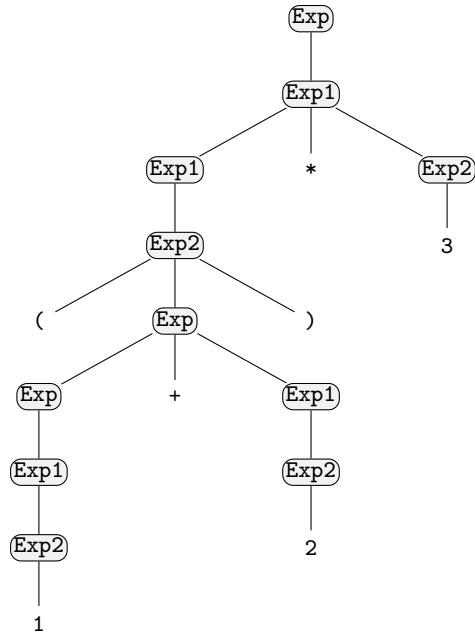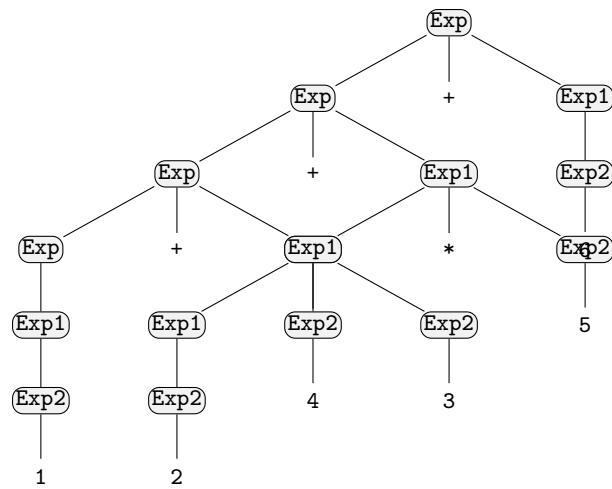
**(a)** 2+1



**(b)** 1+2*3



**(c)** 1+(2*3)



**(d)** (1+2)*3

(Exp)
(Exp1)
(Exp1) * (Exp2)
(Exp2) 3
( (Exp) )
(Exp) + (Exp1)
(Exp1) (Exp2)
(Exp2) 2
1

**(e)** `1+2*3+4*5+6`

(Exp)
(Exp) + (Exp1)
(Exp) + (Exp1) (Exp2)
(Exp) + (Exp1) * (Exp2) 6
(Exp1) (Exp1) (Exp2) (Exp2) 5
(Exp2) (Exp2) 4 3
1 2
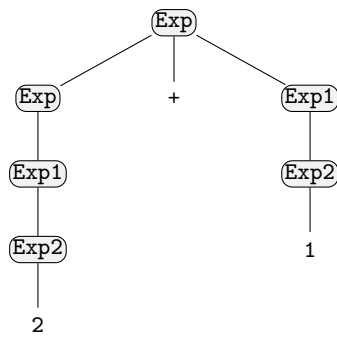
### 2.7.3 Questions

*Why does this grammar make '+' left-associative at the **Exp** level and '*' left-associative at the **Exp1** level, and how would we change it to make them right-associative?*

## 3 Essay (Synthesis)

This section intentionally left blank.

## 4 Evidence of Participation

This section intentionally left blank.

# 5 Conclusion

This section intentionally left blank.

# References

[BLA] Author, *Title*, Publisher, Year.