# Disjoint Union Types in P0

## Project 9 / Group 8

Jason Balaci

McMaster University

April 2021

# Table of Contents


1. Objective & Implementation
2. Examples
3. Implementation Evaluation and Notes
4. Future Work

# Table of Contents


1. Objective & Implementation
2. Examples
3. Implementation Evaluation and Notes
4. Future Work

# What are Disjoint Union Types?

This is a text in the first frame. This is a text in the first frame. This is a text in the first frame.

# How were they implemented?

# Grammar Changes

- Text visible on slide 1

# Grammar Changes

- Text visible on slide 1
- Text visible on slide 2

# Grammar Changes

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 3

# Grammar Changes

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 3
- Text visible on slide 4

# Table of Contents


1. Objective & Implementation
2. Examples
3. Implementation Evaluation and Notes
4. Future Work

# Example: Maybe & Either

# Example: Lists

# Example: Strings

## Remark

Sample text

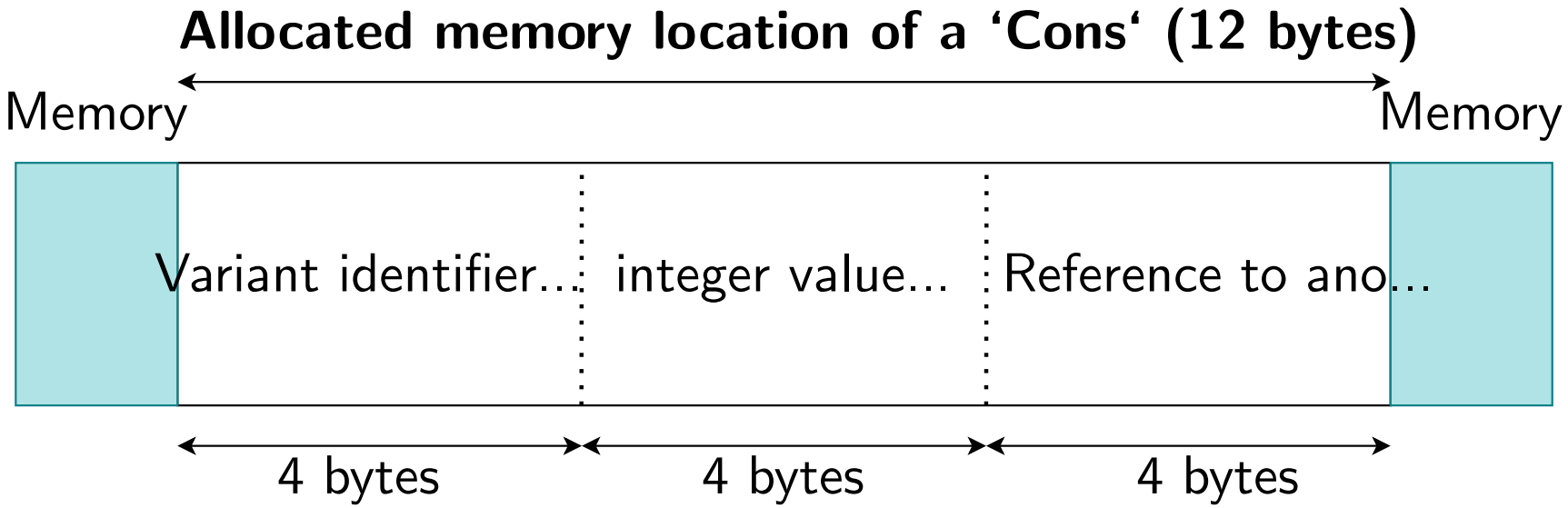
# Table of Contents

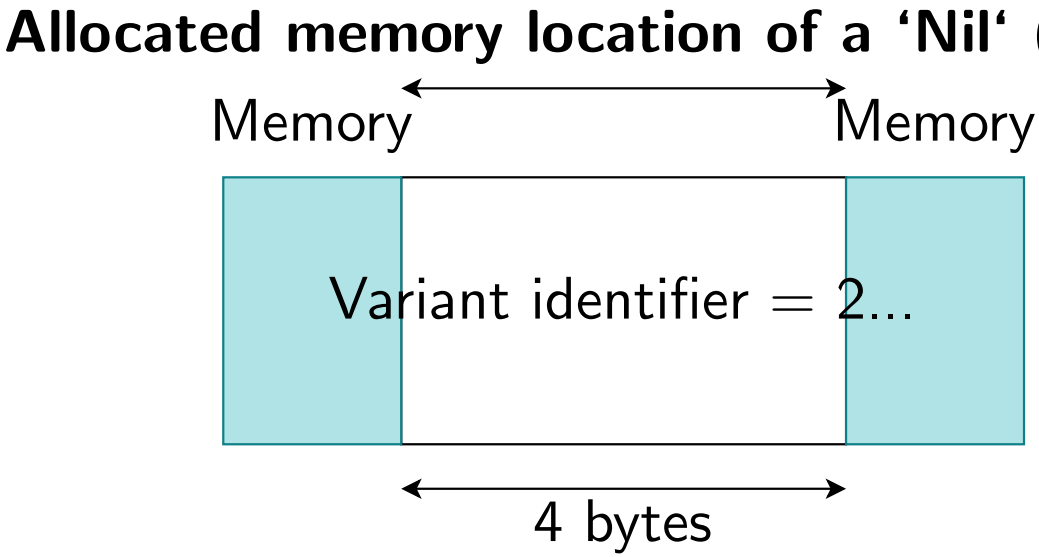

1. Objective & Implementation
2. Examples
3. Implementation Evaluation and Notes
4. Future Work

# Memory Impact & Management

Each instance of an ADT is located on the heap, and instances of local/global ADTs are pointers to the locations of their corresponding ADT on the heap.

# Example: Lists in Memory



# Notable Design Decisions

- Text visible on slide 1

# Notable Design Decisions

- Text visible on slide 1
- Text visible on slide 2

# Notable Design Decisions

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 3

# Notable Design Decisions

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 3
- Text visible on slide 4

# Notes on Runtimes

# Table of Contents


1. Objective & Implementation
2. Examples
3. Implementation Evaluation and Notes
4. Future Work

- Improved Memory Management
  - Memory freeing!
  - Memory reuse!
  - Built-in disjoint union type allocation specialization!

- Improved Memory Management
  - Memory freeing!
  - Memory reuse!
  - Built-in disjoint union type allocation specialization!
- Type variables!
  - Polymorphic disjoint union types! No more StringLists, IntLists, BooleanLists!
  - More code reuse!

- Improved Memory Management
  - Memory freeing!
  - Memory reuse!
  - Built-in disjoint union type allocation specialization!
- Type variables!
  - Polymorphic disjoint union types! No more StringLists, IntLists, BooleanLists!
  - More code reuse!
- More built-in types and syntactic sugars (Strings, Lists, Maps, "abcd..")
  - Strings, Lists, Maps
  - Stronger syntactic sugar for String generation (e.g., "abcd..." for quickly instantiating large strings)

# References