

9. Representation Strategies for Data Types

[lecture 09 -- Interfaces & Representation\(1\).pdf](#)

define-datatype

We can automate mundane data type definitions

the general form of define-datatype

```
(define-datatype type-name type-pred-name  
  { (var-name { (field-name predicate*) } ) }
```

ex:

```
LcExp ::= Identifier  
       ::= (lambda (Identifier) LcExp)  
       ::= (LcExp LcExp)
```

```
(define-datatype lc-exp lc-exp?  
  (var-exp  
    (var identifier?))  
  (lambda-exp  
    (bound-var identifier?)  
    (body lc-exp?))  
  (app-exp  
    (rator lc-exp?)  
    (rand lc-exp?)))
```

ex:

```
S-list ::= ({S-exp}*)  
S-exp ::= Symbol | S-list
```

⇓

```
S-list ::= ()  
        ::= (S-exp . S-exp)  
        ::= symbol | S-list
```

```
(define-datatype s-list s-list?  
  (empty-s-list)  
  (non-empty-s-list  
    (first s-exp?)  
    (rest s-list?)))
```

```
(define-datatype s-exp s-exp?  
  (symbol-s-exp  
    (sym symbol?))  
  (s-list-s-exp  
    (slst s-list?)))
```

GPT Generated Explanation

In Scheme, `define-datatype` is a construct used to define new, custom datatypes. It's particularly useful for creating complex or recursive data structures. Here's how it generally works:

1. **Naming the Datatype:** You start by giving your new datatype a name. This name is then used to refer to this kind of data throughout your program.
2. **Defining Constructors:** You define one or more constructors for your datatype. A constructor is a function that creates instances of your datatype. Each constructor can take different types of arguments, allowing for different ways to create instances of your datatype.
3. **Creating Recognizer Functions:** For each datatype, a recognizer (or predicate) function is automatically created. This function, typically named with a `?` at the end, is used to check if a given piece of data is of the defined datatype.
4. **Defining Selectors:** For each argument of a constructor, a selector function is often generated. These functions are used to extract these arguments from an instance of the datatype.

Example:

```
(define-datatype point point?
  (make-point
    (x-coordinate number?)
    (y-coordinate number?)
  )
)
```

I'm not sure if i understand the usage potential of define-datatype.