Building blocks in Scheme

Jose Abel Castellanos Joo

January 29, 2023

Announcements

- Homework 1 is due this Wednesday

Announcements

- Homework 1 is due this Wednesday
- Franklin will hold recitations this week!

Announcements

- Homework 1 is due this Wednesday
- Franklin will hold recitations this week!
    - Zoom link: `https://unm.zoom.us/j/92093055438`

Agenda I

Parenthesis are auxiliary symbols in math

$$x * (y + z) = (((x)) * ((y + ((z)))))$$

In some programming languages, parenthesis are auxiliary symbols too, in some cases

```
x*(y + z) = (((x))*((y + ((z)))))
```

Parenthesis have a special meaning in Scheme!

Definitions

Definition

A **cons cell** is a data structure containing two pointers, also known as an **ordered pair**.

Definitions

Definition

A **cons cell** is a data structure containing two pointers, also known as an **ordered pair**.

- The `cons` function taking two arguments and returns an "cons cell" containing such arguments.

Definitions

Definition

A **cons cell** is a data structure containing two pointers, also known as an **ordered pair**.

- The `cons` function taking two arguments and returns an "cons cell" containing such arguments.
- The `car` function accepts a "cons cell" as single argument and returns its first component, i.e. `(car (cons x y))` $\to$ `x`.

Definitions

Definition

A **cons cell** is a data structure containing two pointers, also known as an **ordered pair**.

- The `cons` function taking two arguments and returns an "cons cell" containing such arguments.
- The `car` function accepts a "cons cell" as single argument and returns its first component, i.e. `(car (cons x y))` $\rightarrow$ `x`.
- The `cdr` function accepts a "cons cell" as single argument and returns its second component, i.e. `(cdr (cons x y))` $\rightarrow$ `y`.

S-expressions in Schemes

There are two kind of expressions in scheme:

S-expressions in Schemes

There are two kind of expressions in scheme:

- Pairs

S-expressions in Schemes

There are two kind of expressions in scheme:

- Pairs
- Non-pairs,

S-expressions in Schemes

There are two kind of expressions in scheme:

- Pairs
- Non-pairs, also known as atomic expressions

Recursive definition

Definition

- Any atomic expression `x` is a **s-expression**.

Recursive definition

Definition

- Any atomic expression `x` is a **s-expression**.
- If `x` and `y` are **s-expressions**, then `(cons x y)` is also a **s-expression**.

Recursive definition

Definition

- `'()` is a **list**.

Recursive definition

Definition

- `'()` is a **list**.
- If `x` is a s-expression and `y` a **list**, then `(cons x y)` is a **list**.

Thought provoking

- Is `'()` an atom or a list?

Thought provoking

- Is `'()` an atom or a list? or both?
- Are s-expressions actually lists?

Thought provoking

- Is `'()` an atom or a list? or both?
- Are s-expressions actually lists?
- What **is** `(+ 2 3)`?

Thought provoking

- Is `'()` an atom or a list? or both?
- Are s-expressions actually lists?
- What **is** `(+ 2 3)`? A list or an s-expression?

Thought provoking

- Is `'()` an atom or a list? or both?
- Are s-expressions actually lists?
- What **is** `(+ 2 3)`? A list or an s-expression?
- What are **programs** in Scheme?