The ACL2s Language

Pete Manolios
Northeastern

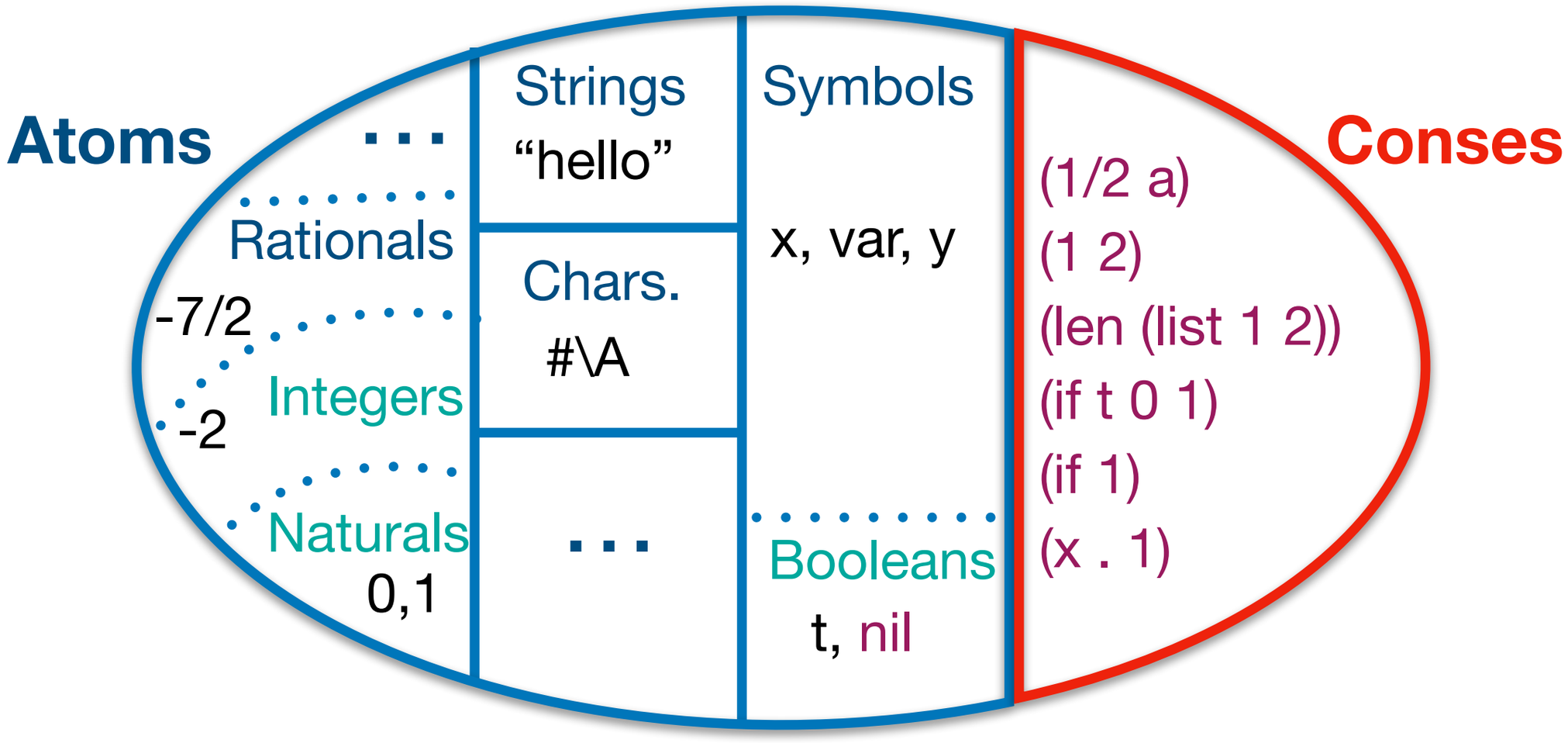
Logic and Computation, 1/16/2018

Objectives

- Lists
- Contract violations
- Termination
- ACL2s Demo

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019

Expressions

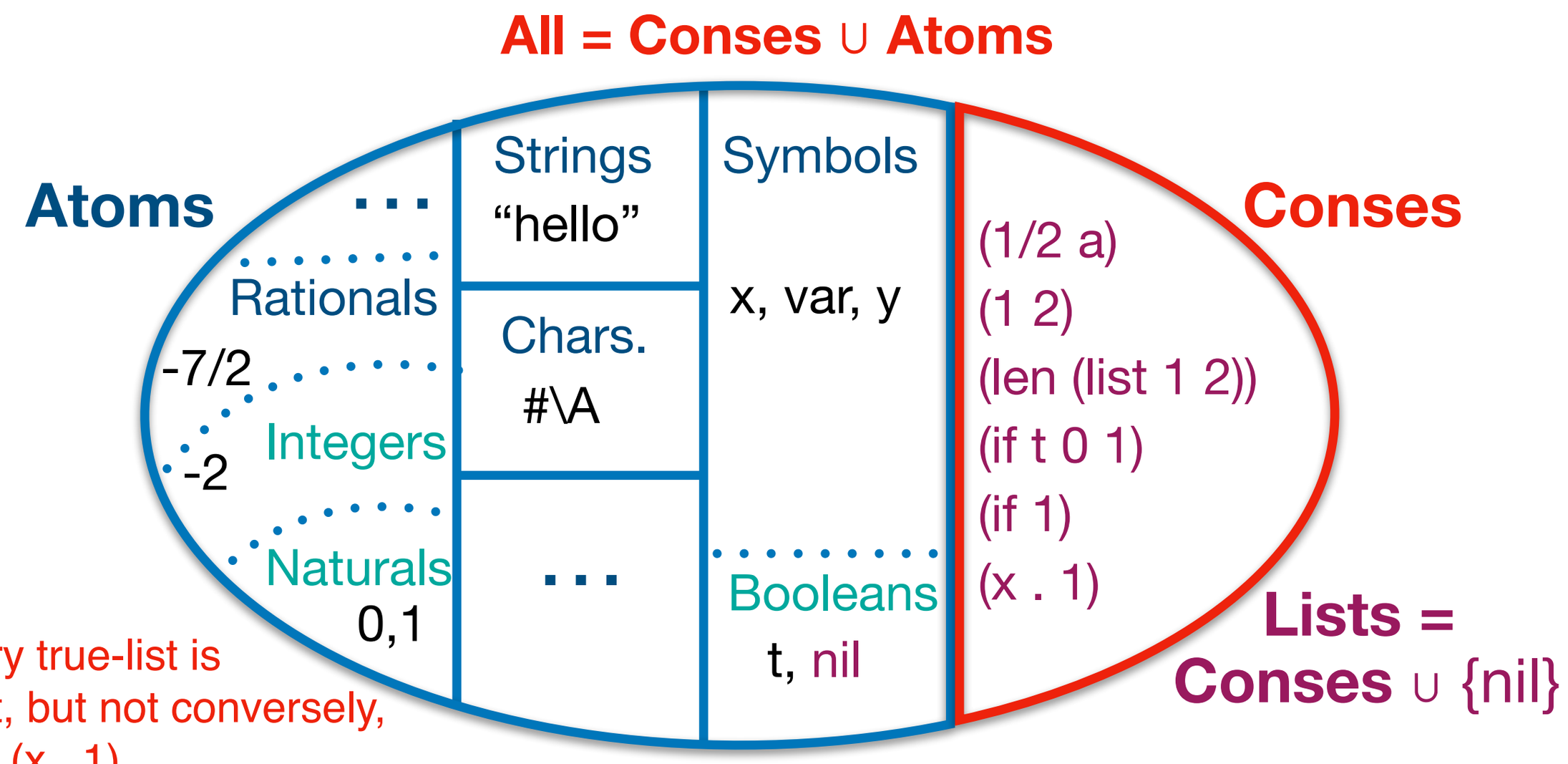


Expressions are elements of the universe, but not conversely

Which conses are expressions?

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019

ACL2 Universe: Conses



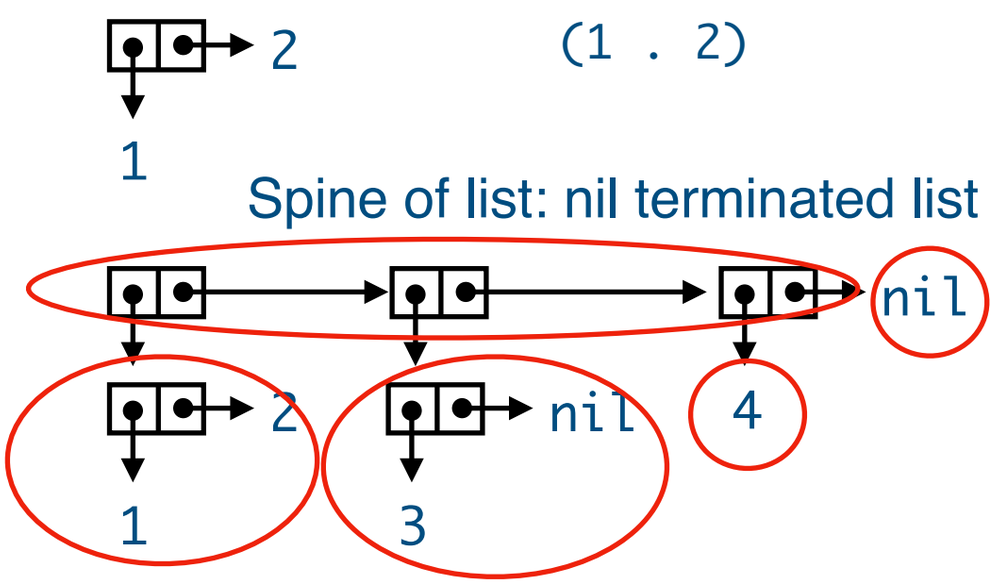
True-Lists = $\cup_{i \in \mathbb{N}} TL_i$

$TL_0 = \{ () \}$, $TL_{i+1} = TL_i \cup \{(\texttt{cons x l}): x \in \texttt{All}, l \in TL_i\}$

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019

Conses, Dotted Pair Notation

- Built-in functions & signatures
  - `consp: All → Boolean`
  - `cons: All × All → Cons`
  - `car: List → All`
  - `cdr: List → All`



- A cons is just a pair, eg, `(cons 1 2)`
- Since pairs are nested, conses are trees, eg,
  `(cons (cons 1 2) (cons (cons 3 nil) (cons 4 nil))`

- Dotted pair notation simplification rules
- `(x . nil)` → `(x)`
- `(… . (x …))` → `(… x …)`

```
((1 . 2) . ((3 . nil) . (4 . nil)))
((1 . 2) . ((3) . (4)))
((1 . 2) (3) . (4))
((1 . 2) (3) 4)
```

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019

Conses

- Built-in functions & signatures
  - consp: All → Boolean
  - cons: All × All → Cons
  - car: List → All
  - cdr: List → All
  - ⟦(cons x y)⟧ = (⟦x⟧ . ⟦y⟧)
  - ⟦(consp x)⟧ = t iff ⟦x⟧ is of the form (...) but is not ()
  - ⟦(car x)⟧ = a, when ⟦x⟧ = (a . b), nil otherwise
  - ⟦(cdr x)⟧ = b, when ⟦x⟧ = (a . b), nil otherwise
  - Examples
    - ⟦(consp nil)⟧ = nil (since nil = ())
    - ⟦(car ())⟧ = nil, ⟦(cdr ())⟧ = nil (since nil is a List)
    - ⟦(consp (cons nil nil))⟧ = t (since ⟦(cons nil nil)⟧ = (nil . nil))
    - ⟦(car (cdr (cons (if t 3 4) (cons 1 ()))))⟧ = 1

```
(definec true-listp (l :all) :bool
  (if (consp l)
      (true-listp (rest l))
    (equal l ())))
```

Macros

- List construction is prevalent, so ACL2s provides `list`
- `(list` x_1 x_2 $\cdots$ x_n`)` abbreviates (or is shorthand for)
  `(cons` x_1 `(cons` x_2 $\cdots$ `(cons` x_n `nil)` $\cdots$`)`
- Notice that `list` takes an arbitrary number of arguments
  - ⟦`(list)`⟧ = `()`
  - ⟦`(list 1)`⟧ = `(1 . nil)` = `(1)`
  - ⟦`(list 1 2)`⟧ = `(1 . (2 . nil))` = `(1 2)`
- `list` is a macro: it gets expanded into an expression
- `first, rest` abbreviate `car, cdr`
- `caar, cadr, cdar,` …., abbreviate `(car (car …)), (car (cdr …)),` …
- `second, third`, …., equivalent to `cadr, caddr,` …

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019

Cond

- ```
  (cond (c1 e1)
        (c2 e2)
        ...
        (cn en))
  ```

  expands into

  ```
  (if c1
      e1
    (if c2
        e2
      ...
      (if cn
          en
          nil)))
  ```

- We will always use `t` for the last test in a `cond` (so the `nil` will not be reachable)
  ```

ACL2s Demo

- ACL2s perspective
- Creating a project, syncing files, refresh
- Loading hwk1
- Split window
- ACL2s mode
- REPL
- Atoms
- Definitions
- Line
- GUI interface
- Termination
- Contracts

Slides by Pete Manolios for CS2800, Logic & Computation, NU 2019