

Conditionals

; maybe-wanted : image -> image



Conditionals in Algebra

General format of conditionals in algebra:

$$\left\{ \begin{array}{ll} \textit{answer} & \textit{question} \\ \dots & \\ \textit{answer} & \textit{question} \end{array} \right.$$

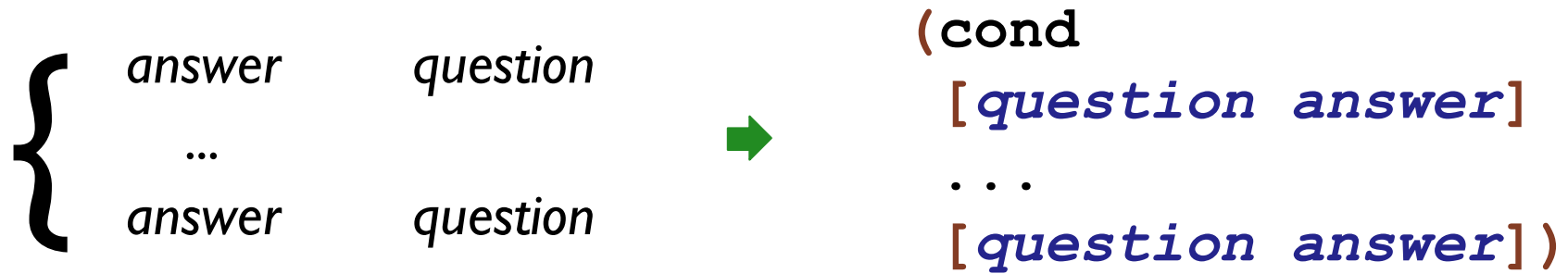
Example:

$$\text{abs}(x) = \left\{ \begin{array}{ll} x & \text{if } x > 0 \\ -x & \text{otherwise} \end{array} \right.$$

$$\text{abs}(10) = 10$$

$$\text{abs}(-7) = 7$$

Conditionals in Racket


{ *answer* *question*
 ...
 answer *question* → (cond
 [*question answer*]
 ...
 [*question answer*])

Conditionals in Racket

```
(cond  
  [question answer]  
  . . .  
  [question answer])
```

- Any number of `cond` “lines”
- Each line has one *question* expression and one *answer* expression
- Last *question* can be `else` for “otherwise”

```
(define (absolute x)  
  (cond  
    [(> x 0) x]  
    [else (- x)]))
```

```
(absolute 10) → 10
```

```
(absolute -7) → 7
```

Evaluation Rules for cond

First question is literally **true**:

```
(cond
  [true answer]
  ...
  [question answer]) → answer
```

i.e., keep only the first answer

Example:

```
(* 1 (cond
       [true 0])) → (* 1 0) → 0
```

Evaluation Rules for cond

First question is literally **false**:

```
(cond
  [false answer]
  [question answer]
  ...
  [question answer]) → (cond
  [question answer]
  ...
  [question answer])
```

i.e., throw away the first line

Example:

```
(+ 1 (cond
  [false 1]
  [true 17])) → (+ 1 (cond
  [true 17]))
→ (+ 1 17) → 18
```

Evaluation Rules for cond

First question isn't a value, yet:

```
(cond  
  [question answer]  
  ...  
  [question answer])
```

→

```
(cond  
  [nextques answer]  
  ...  
  [question answer])
```

where *question* → *nextques*

i.e., evaluate first question as sub-expression

Example:

```
(+ 1 (cond  
      [(< 1 2) 5]  
      [else 8]))
```

→

```
(+ 1 (cond  
      [true 5]  
      [else 8]))
```

→ (+ 1 5) → 6

Evaluation Rules for cond

No true answers:

`(cond)` → *error*

Just an `else`:

`(cond
[else answer])` → *answer*

Programming with Conditionals



```
(define clyde
```

```
; maybe-wanted : image -> image
```

```
(define (maybe-wanted who)
```

```
  (cond
```

```
    [(image=? who clyde)
```

```
     (above (text "WANTED" 32 "red") who)]
```

```
    [else
```

```
     who]))
```

WANTED



```
(maybe-wanted
```

→

Programming with Conditionals



```
(define clyde
```

```
; maybe-wanted : image -> image
```

```
(define (maybe-wanted who)
```

```
  (cond
```

```
    [(image=? who clyde)
```

```
     (above (text "WANTED" 32 "red") who)]
```

```
    [else
```

```
     who]))
```



```
(maybe-wanted
```

