

EE570—Artificial Intelligence

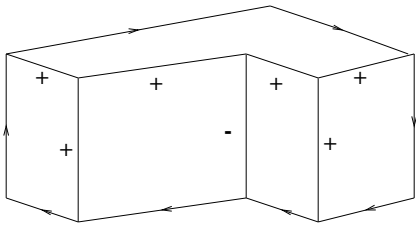
Lecture #13: Waltz Line Labeling

Jeffrey Mark Siskind

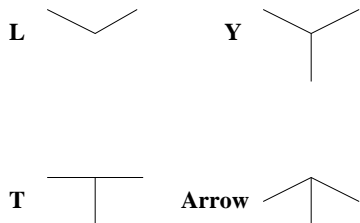
School of Electrical and Computer Engineering
Purdue University
Electrical Engineering Building, Room 330
465 Northwestern Avenue
West Lafayette IN 47907-2035 USA
voice: 765/496-3197
fax: 765/494-6440
qobi@purdue.edu
<http://www.ece.purdue.edu/~qobi>

Waltz Line Labeling

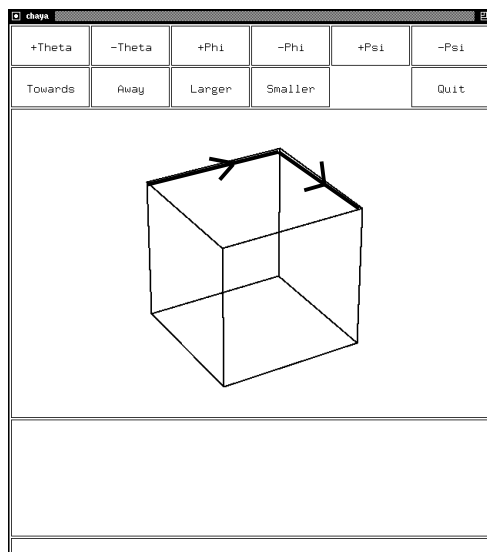
A sample line drawing



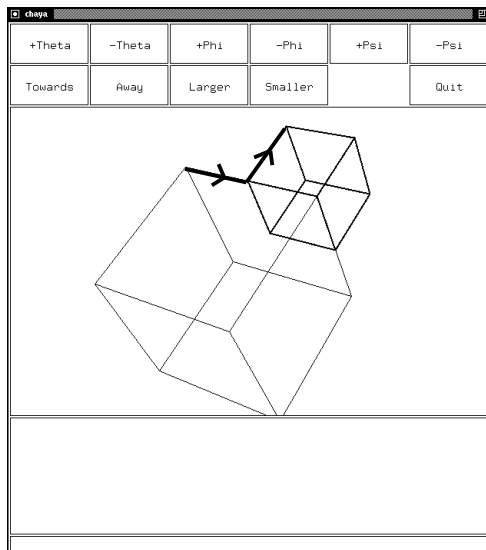
The four vertex types



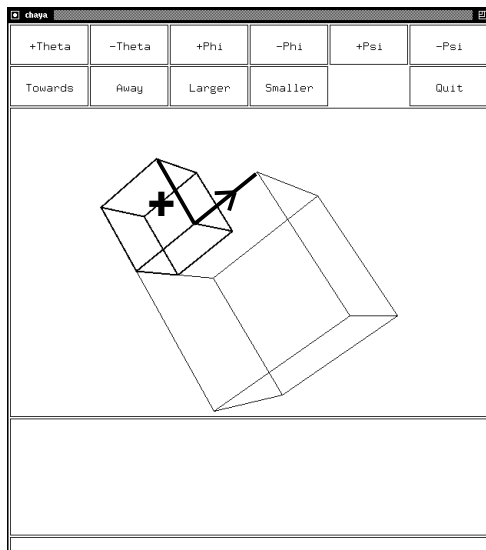
L Vertex—Case I



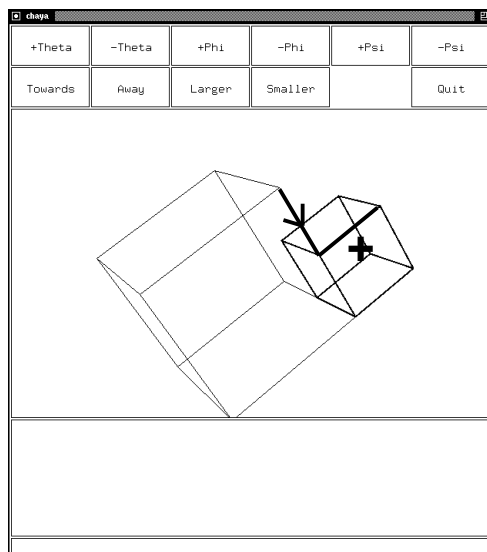
L Vertex—Case II



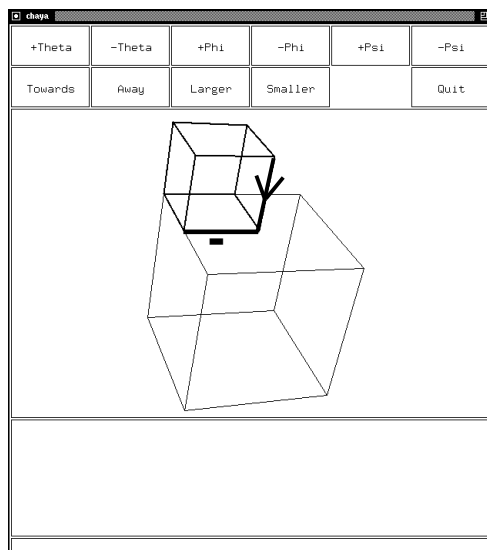
L Vertex—Case III



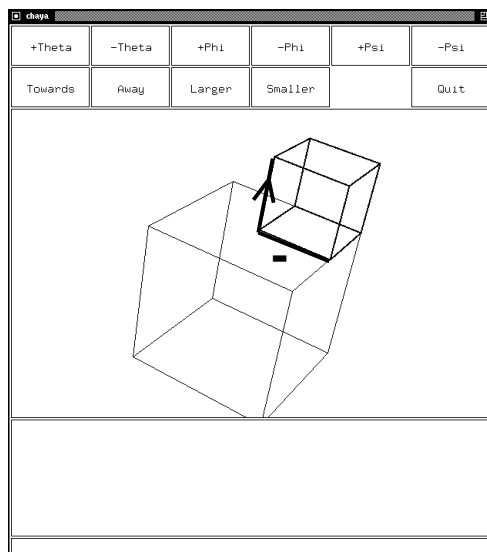
L Vertex—Case IV



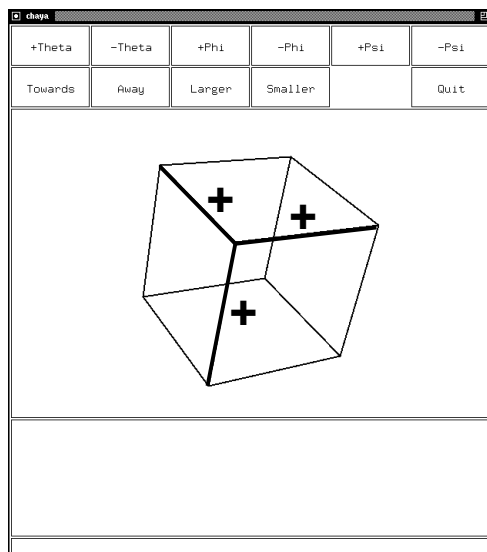
L Vertex—Case V



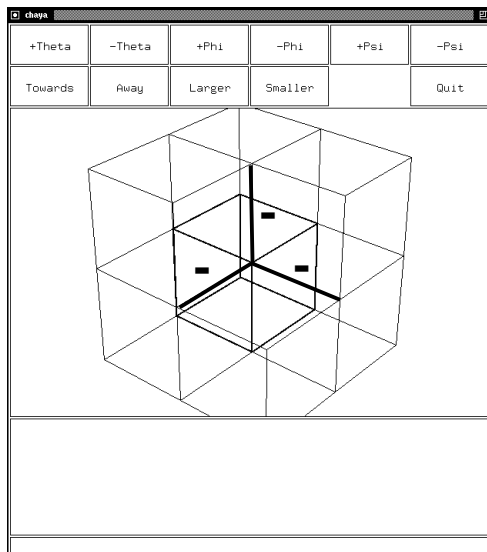
L Vertex—Case VI



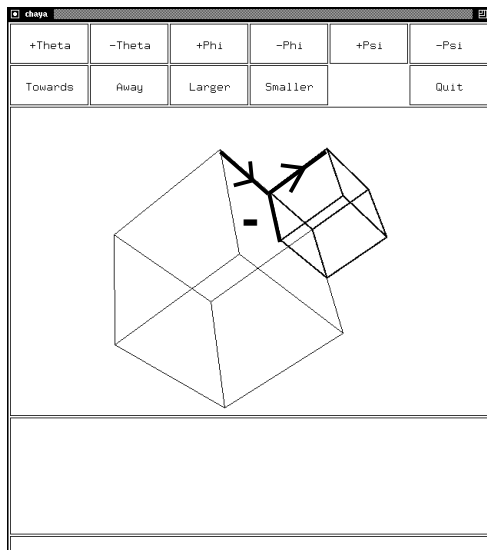
Y Vertex—Case I



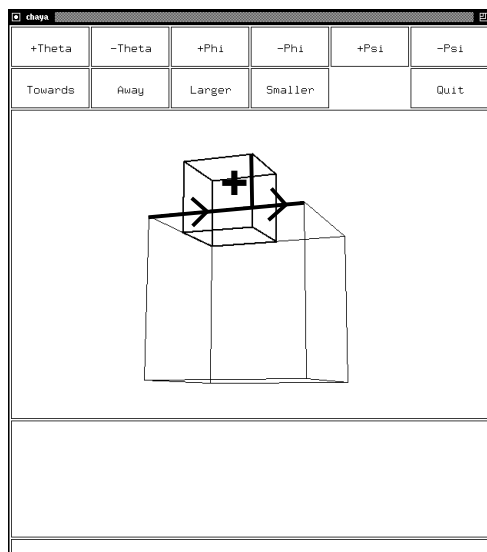
Y Vertex—Case II



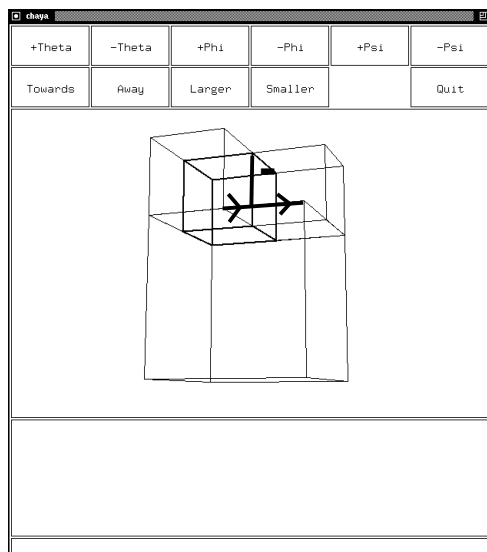
Y Vertex—Cases III, IV, and V



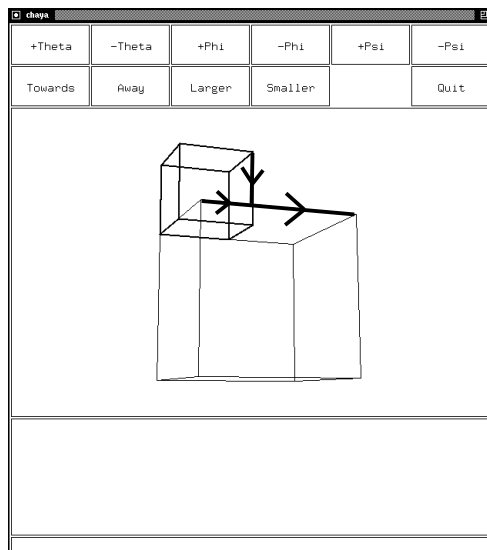
T Vertex—Case I



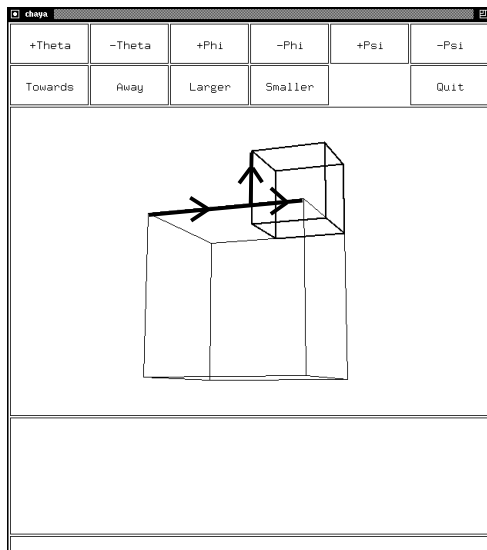
T Vertex—Case II



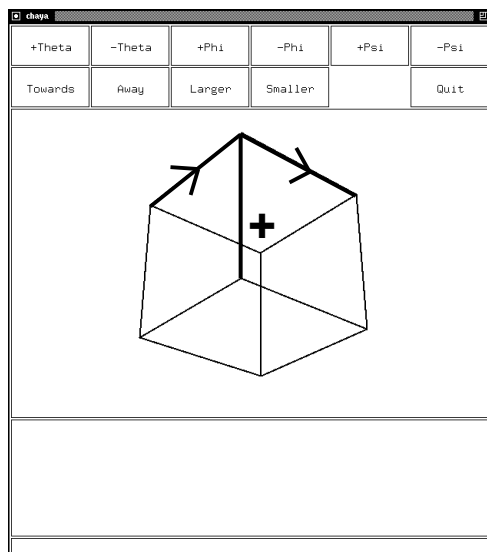
T Vertex—Case III



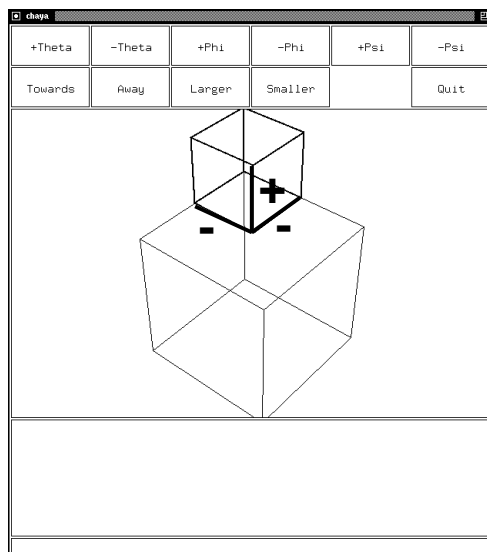
T Vertex—Case IV



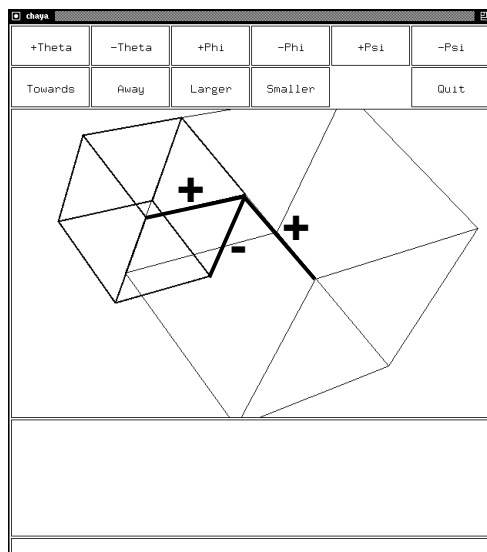
Arrow Vertex—Case I



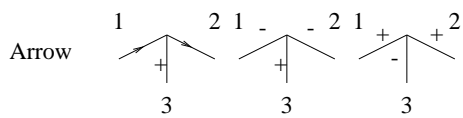
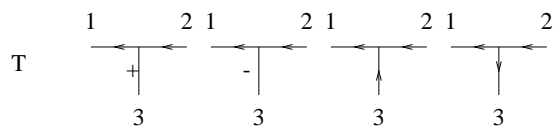
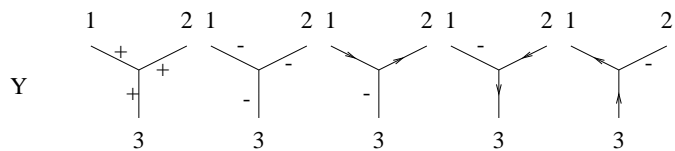
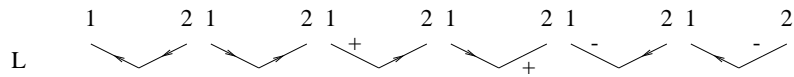
Arrow Vertex—Case II



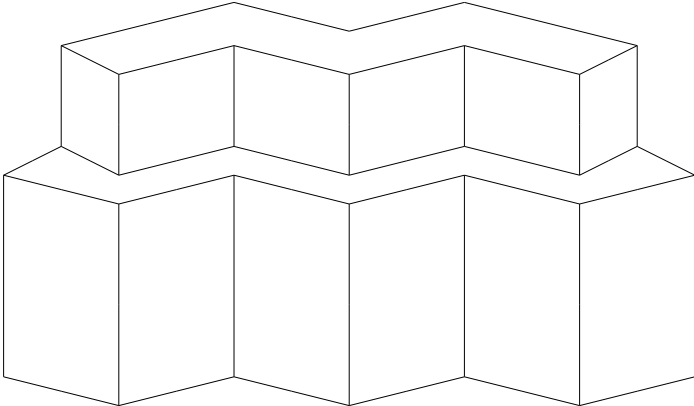
Arrow Vertex—Case III



The Constraints Imposed by Each Vertex Type



An Example of Propagation



Waltz Line Labeling in Scheme—I

```
(define-structure vertex
  point
  x y
  type
  side1 edge1
  side2 edge2
  side3 edge3)

(define-structure edge domain-variable u v)

(define *vertices* #f)

(define *edges* #f)
```

Waltz Line Labeling in Scheme—II

```
(define (set-vertex! edge side vertex)
  (case side
    ((u) (when (edge-u edge) (panic "This shouldn't happen"))
          (set-edge-u! edge vertex))
    ((v) (when (edge-v edge) (panic "This shouldn't happen"))
          (set-edge-v! edge vertex))
    (else (panic "This shouldn't happen"))))

(define (make-perimeter-vertex side1 edge1 edges)
  (make-vertex
   #f
   #f #f
   'perimeter
   side1 (vector-ref edges edge1)
   #f #f
   #f #f))
```

Waltz Line Labeling in Scheme—III

```
(define (make-l-vertex x y z side1 edge1 side2 edge2 edges)
  (let ((vertex (make-vertex
                 (make-3-by-1-matrix x y z)
                 #f #f
                 '1
                 side1 (vector-ref edges edge1)
                 side2 (vector-ref edges edge2)
                 #f #f)))
    (set-vertex! (vector-ref edges edge1) side1 vertex)
    (set-vertex! (vector-ref edges edge2) side2 vertex)
    vertex))
```

Waltz Line Labeling in Scheme—IV

```
(define (before-update-edge edge)
  (when *display*
    (let ((u (edge-u edge))
          (v (edge-v edge)))
      (draw-edge edge)
      (xdrawline *display* *display-pane*
                 *thick-flipping-gc*
                 (vertex-x u) (vertex-y u)
                 (vertex-x v) (vertex-y v))
      (xflush *display*)
      (usleep 100000))))
```

Waltz Line Labeling in Scheme—V

```
(define (after-update-edge edge)
  (when *display*
    (let ((u (edge-u edge))
          (v (edge-v edge)))
      (xdrawline *display* *display-pane*
                 *thick-flipping-gc*
                 (vertex-x u) (vertex-y u)
                 (vertex-x v) (vertex-y v))
      (draw-edge edge)
      (xflush *display*)
      (usleep 100000)
      (pause))))
```

Waltz Line Labeling in Scheme—VI

```
(define (create-edge)
  (make-edge (create-domain-variable '(+ - < >)) #f #f))

(define (attach-demons-to-edges! edges)
  (let ((display *display*))
    (set! *display* #f)
    (for-each-vector
      (lambda (edge)
        (attach-before-demon!
          (lambda () (before-update-edge edge))
          (edge-domain-variable edge))
        (attach-after-demon!
          (lambda () (after-update-edge edge))
          (edge-domain-variable edge)))
      edges)
    (set! *display* display)))
```

Waltz Line Labeling in Scheme—VII

```
(define (fred)
  (set! *observer-theta* 160)
  (set! *observer-phi* 0)
  (set! *observer-psi* -50)
  (set! *observer-distance* 10)
  (set! *scale* 1000)
  (set! *edges* (make-vector 9))
  (for-each-n (lambda (i) (vector-set! *edges* i (create-edge))) 9)
  (set! *vertices*
    (vector (make-l-vertex 0 0 0 'u 1 'v 6 *edges*)
            (make-arrow-vertex 0 2 0 'v 5 'u 6 'v 7 *edges*)
            (make-l-vertex 0 2 2 'u 5 'v 4 *edges*)
            (make-arrow-vertex 2 2 2 'v 3 'u 4 'v 8 *edges*)
            (make-l-vertex 2 0 2 'u 3 'v 2 *edges*)
            (make-arrow-vertex 2 0 0 'v 1 'u 2 'u 0 *edges*)
            (make-y-vertex 2 2 0 'u 7 'u 8 'v 0 *edges*)
            (make-perimeter-vertex 'v 1 *edges*)
            (make-perimeter-vertex 'v 2 *edges*)
            (make-perimeter-vertex 'v 3 *edges*)
            (make-perimeter-vertex 'v 4 *edges*)
            (make-perimeter-vertex 'v 5 *edges*)
            (make-perimeter-vertex 'v 6 *edges*)))
  (update-observer)
  (attach-demons-to-edges! *edges*)
  (redraw-display-pane))
```