ECE570 Lecture 8: Alpha/Beta Prunning

Jeffrey Mark Siskind

School of Electrical and Computer Engineering

Fall 2013

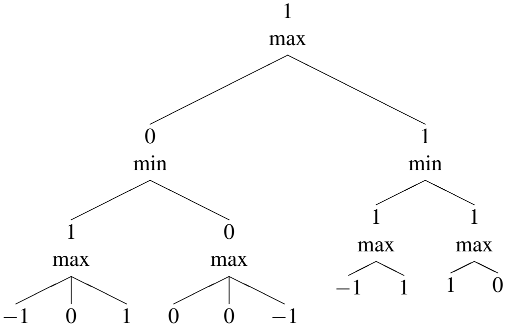
PURDUE UNIVERSITY

Set Comprehension

$$\{x \in L | p(x)\}$$

```
(define (remove-if-not p l)
 (cond
  ((null? l) '())
  ((p (first l)) (cons (first l) (remove-if-not p (rest l))))
  (else (remove-if-not p (rest l)))))
```

Game Trees



Breadth-First Maximization

$$\max_{x \in L} f(x)$$

```
(define (maximize f l)
 (reduce max (map f l) -1))
```

(define (w^* b)

$\vdots$

(* $p(b)$

    (maximize (lambda (m) (* $p(b)$ (w^* $b'(m,b)$)))

              $m(b)$))

 ...)

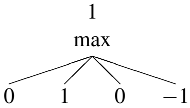
Depth-First Maximization

$$\max_{x \in L} f(x)$$

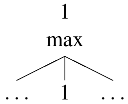
```
(define (maximize f l)
 (define (loop best-so-far l)
  (cond ((null? l) best-so-far)
        (else (loop (max (f (first l)) best-so-far)
                    (rest l)))))
 (loop -1 l))
```

(define (w^* b)
 $\vdots$
 ($\star$ $p(b)$
    (maximize (lambda (m) ($\star$ $p(b)$ (w^* $b'(m,b)$)))
              $m(b)$))
 ...)

Left-to-Right Pruning—I



Left-to-Right Pruning—II



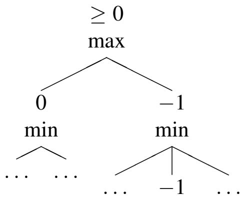
Left-to-Right Pruning—III

$$\max_{x \in L} f(x)$$

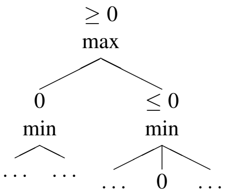
```
(define (maximize f l)
 (define (loop best-so-far l)
  (cond ((= best-so-far 1)) 1)
        ((null? l) best-so-far)
        (else (loop (max (f (first l)) best-so-far)
                    (rest l)))))
 (loop -1 l))

(define (w* b)
 ⋮
 (* p(b)
    (maximize (lambda (m) (* p(b) (w* b'(m,b))))
              m(b)))
 ...)
```

Alpha/Beta Pruning—I



Alpha/Beta Pruning—II



Alpha/Beta Pruning—III

- Each node has a *limit* parameter. Stop the search when a value as good or better than the limit is found and return the limit.
- Each node has a *best-so-far* value. Pass this value as the limit parameter for the child nodes.

Alpha/Beta Pruning—IV

$$w_l^*(b) = \begin{cases} w^0(b) & w^0(b) \neq 0 \vee m(b) = \{\} \\ p(b) \max\limits_{\substack{m \in m(b) \\ l' \text{ is best so far}}} p(b) w^*_{p(b)l'}(b'(m,b)) & \text{otherwise} \end{cases}$$

Alpha/Beta Pruning—V

$$\max_{\substack{x\in L\\ x' \text{ is best so far}}} f(x,x')$$

```
(define (maximize f l limit)
 (define (loop best-so-far l)
  (cond ((>= best-so-far limit)) 1)
        ((null? l) best-so-far)
        (else (loop (max (f (first l) best-so-far) best-so-far)
                    (rest l)))))
 (loop -1 l))
```

(define (w^* b l)

⋮

(⋆ $p(b)$

(maximize (lambda (m l') (⋆ $p(b)$ (w^* $b'(m,b)$ (⋆ $p(b)$ l'))))

$m(b)$

(⋆ $p(b)$ l)))

...)

Evaluation Function—I

$$\tilde{w}^0(b) = \begin{cases} 1 & w^0(b) = 1 \vee \text{estimates } w^*(b) = 1 \\ 0 & (w^0(b) = 0 \wedge m(b) = \{\}) \vee \text{estimates } w^*(b) = 0 \\ -1 & w^0(b) = -1 \vee \text{estimates } w^*(b) = -1 \end{cases}$$

$\tilde{w}^0(b)$ estimates the value of $w^*(b)$.

Bounded Game-Tree Search

$$
\tilde{w}^k(b) = \begin{cases} w^0(b) & w^0(b) \neq 0 \vee m(b) = \{\} \\ \tilde{w}^0(b) & k = 0 \\ p(b) \max\limits_{m \in m(b)} p(b)\tilde{w}^{k-1}(b'(m,b)) & \text{otherwise} \end{cases}
$$

$$
\tilde{m}^k(b) = \begin{cases} \{\} & w^0(b) \neq 0 \\ \{m \in m(b) | p(b)\tilde{w}^{k-1}(b'(m,b)) \geq p(b)\tilde{w}^k(b)\} & \text{otherwise} \end{cases}
$$

Evaluation Function—II

$$
\begin{array}{rcl}
\tilde{w}^0(b) & \in & \{-1, 0, 1\} \\
\tilde{w}^0(b) & \in & [-1, 1] \\
\tilde{w}^0(b) & \in & \left\{ \begin{array}{ll}
\{1\} & w^0(b) = 1 \\
(0, 1) & w^0(b) = 0 \wedge m(b) \neq \{\} \wedge \text{estimates } w^*(b) = 1 \\
\{0\} & w^0(b) = 0 \vee \text{estimates } w^*(b) = 0 \\
(-1, 0) & w^0(b) = 0 \wedge m(b) \neq \{\} \wedge \text{estimates } w^*(b) = 11 \\
\{-1\} & w^0(b) = -1
\end{array} \right.
\end{array}
$$

$\tilde{w}^k(b)$ also estimates the value of $w^*(b)$.
Thus it is also an evaluation function.
$\tilde{w}^k(b)$ is a better estimation than $\tilde{w}^0(b)$.
If $k_1 > k_2$, $\tilde{w}^{k_1}(b)$ is a better estimation than $\tilde{w}^{k_2}(b)$.