UC Berkeley EECS
Sr. Lecturer SOE
Dan Garcia

CS10 The Beauty and Joy of Computing

Lecture #19 Higher Order Functions II

2013-11-13

PRO SELF-DRIVING CARS CON

PRO

- Fewer accidents – 90% of accidents caused by human error
- Efficient travel since can create convoys
- Huge efficiency gains if you can work + drive

CON

- Who gets sued when there's an accident?
- Handing control back to driver takes ~5 sec
- Very expensive
- Could be dangerous if they can't handle case

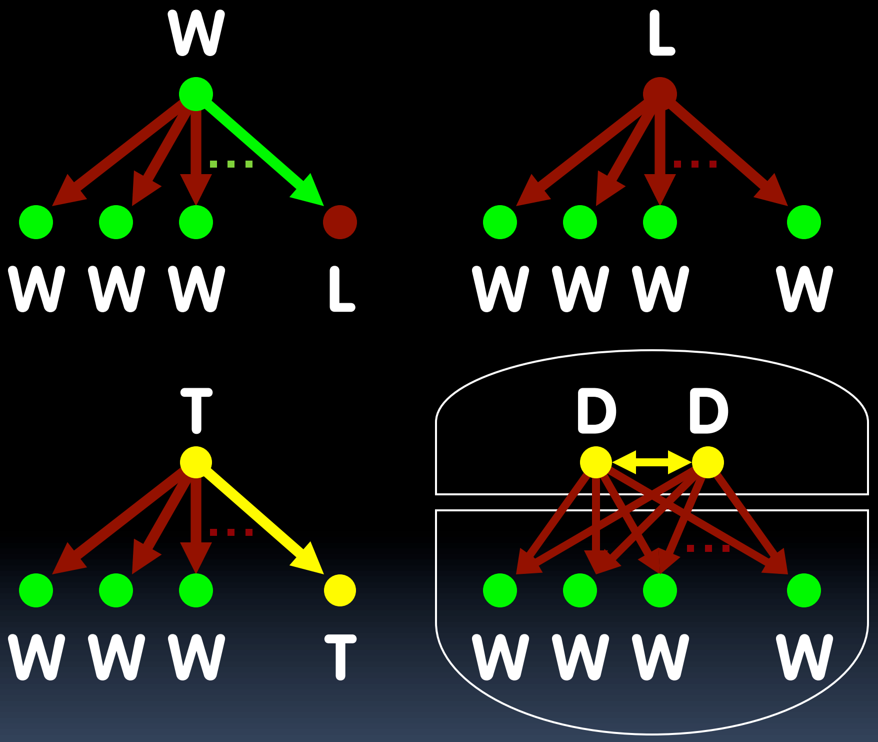
www.technologyreview.com/featuredstory/520431/driverless-cars-are-further-away-than-you-think/

I do research on Board Games…

- No chance, such as dice or shuffled cards
- Both players have complete information
  - No hidden information, as in Stratego & Magic
- Two players (Left & Right) usually alternate moves
  - Repeat & skip moves ok
  - Simultaneous moves not ok
- The game can end in a pattern, capture, by the absence of moves, or …

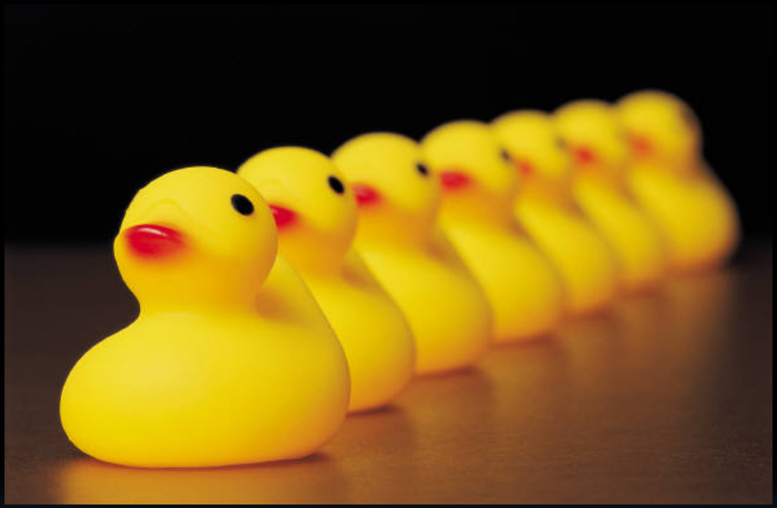
A Strong Solution visits every position

- For every position
  - Assuming alternating play
  - Value …
    (for player whose turn it is)
    - 🟢 Winning (∃ losing child)
    - 🔴 Losing (All children winning)
    - 🟡 Tieing (!∃ losing child, but ∃ tieing child)
    - 🟡 Drawing (can't force a win or be forced to lose)
  - Remoteness
    - How long before game ends?



Strong Solving Example: 1,2,…,10

- **Rules (on your turn):**
  - Running total = 0
- **Rules (on your turn):**
  - Add 1 or 2 to running total
- **Goal**
  - Be the FIRST to get to 10
- **Example**
  - Ana: “2 to make it 2”
  - Bob: “1 to make it 3”
  - Ana: “2 to make it 5”
  - Bob: “2 to make it 7” → photo
  - Ana: “1 to make it 8”
  - Bob: “2 to make it 10” I WIN!

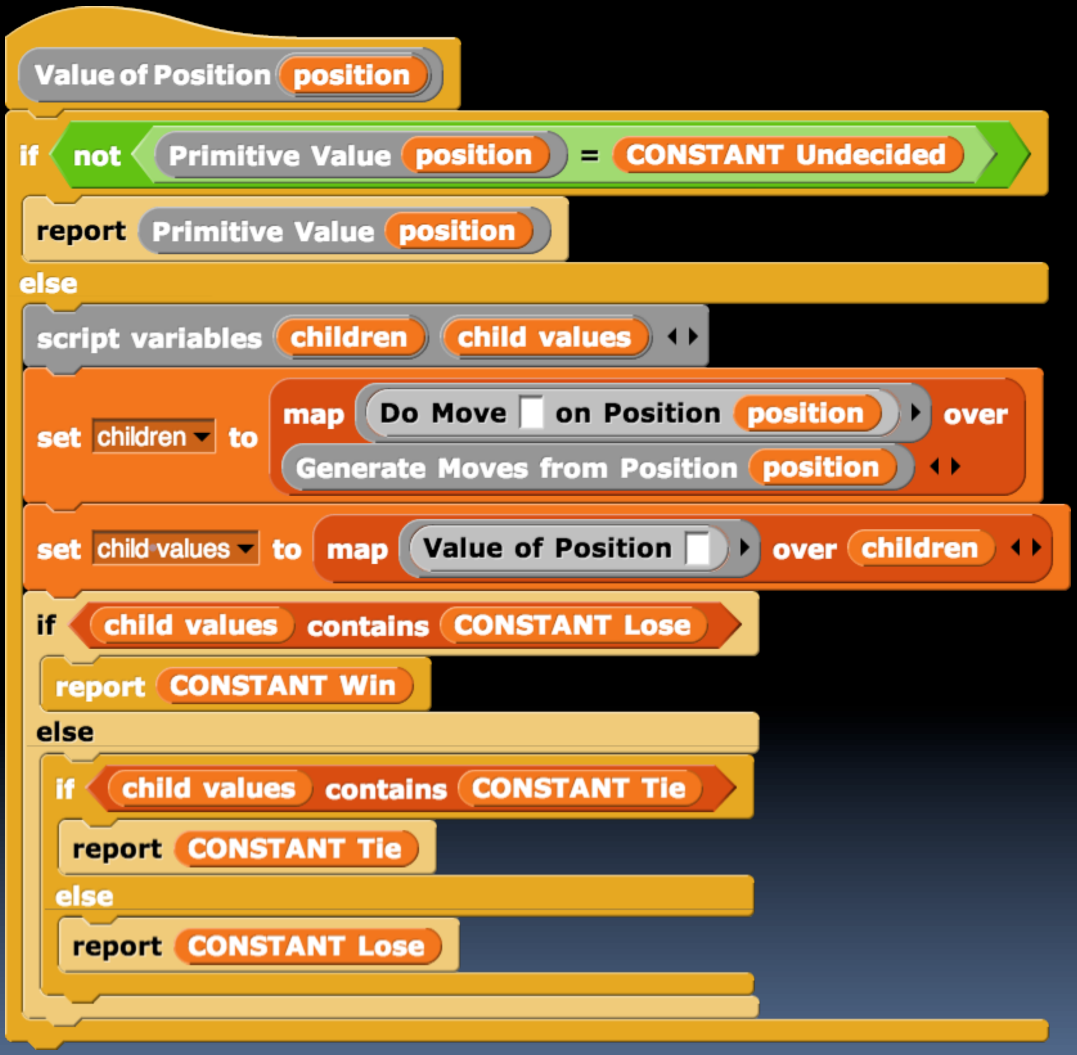

7 ducks (out of 10)

Let's write code to determine value!

- 0 = Win
- 1 = Lose
- 2 = Win
- 3 = Win
- 4 = Lose
- 5 = Win
- 6 = Win
- 7 = Lose
- 8 = Win
- 9 = Win
- 10 = Lose

- P = Position
- M = Move
- We only need 3 blocks to define a game
  - Do Move M on Position P
    - → a new Position
  - Generate Moves from Position P
    - → list of Moves
  - Primitive Value of Position P
    - → {win, lose, tie, undecided}
- Let's write **Value of Position P**

Answer



```
Value of Position (position)
if <not <(Primitive Value (position)) = (CONSTANT Undecided)>>
    report (Primitive Value (position))
else
    script variables (children) (child values)
    set [children] to (map (Do Move [] on Position (position)) over (Generate Moves from Position (position)))
    set [child values] to (map (Value of Position []) over (children))
    if <(child values) contains (CONSTANT Lose)>
        report (CONSTANT Win)
    else
        if <(child values) contains (CONSTANT Tie)>
            report (CONSTANT Tie)
        else
            report (CONSTANT Lose)
```

Garcia