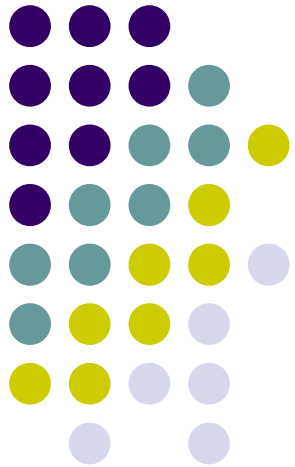
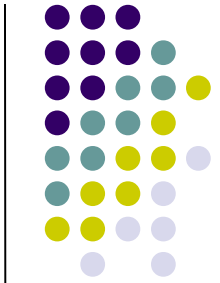
Introduction to Intelligent, Cognitive, and Knowledge-Based Systems

Command and Behavior Fusion

Gal A. Kaminka
galk@cs.biu.ac.il

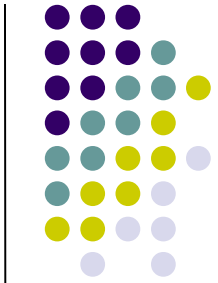


Previously …



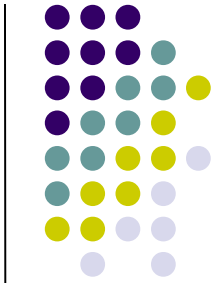
- Behavior Selection/Arbitration
- Activation-based selection
  - winner-take-all selection, behavior networks
  - *argmax* selection (priority, utility, success likelihood, …)
- State-based selection
  - Markovian: Sequencing through FSA
  - World models, preconditions and termination conditions

This week: Behavior Fusion



Behavior Fusion:

- Rosenblatt-Payton Command Fusion
- Potential Fields
- Fuzzy Control Rules
- Context-dependent fusion of behaviors



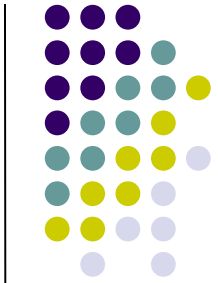
The Problem with Behavior Selection

Example: Brook's Subsumption Architecture

- Multiple layers, divided by competence
- All layers get all sensor readings, issue commands
- Higher layers override commands by lower layers
  - i.e., priority-based selection

What happens when a command is overriden?

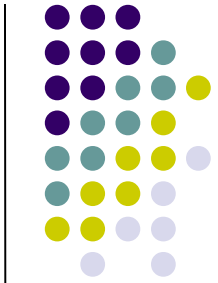
Information Loss in Selection



- Layer chooses best command for its competence level
- Implication: No other command is possible

- But layer implicitly knows about satisficing solutions
  - Satisficing: Loosely translated as “good enough”
- These get ignored if command is overriden

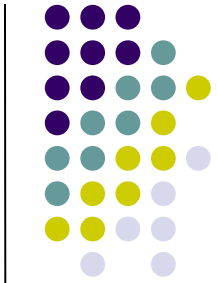
Why is this a problem



Example:

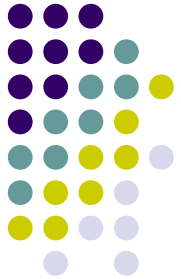
- Avoid-obstacle wants to keep away from left
  - [20,160] heading possible
  - Best: +90 degree heading
- Seek-goal wants to keep towards goal ahead
  - [-30,30] heading possible
  - Best: +0 degree heading
- No way to layer these such that overriding works
  - But [20,30] is good for both!

There is a deeper problem



- When a higher-level behavior subsumes another
  - It must take the other's decision-making into account
  - A higher behavior may have to contain lower behaviors
  - Include within itself their decision-making
- Example: Goal-seek overrides avoid
  - Must still avoid while seeking goal
  - May need to reason about obstacles while seeking

Rosenblatt-Payton Command Fusion

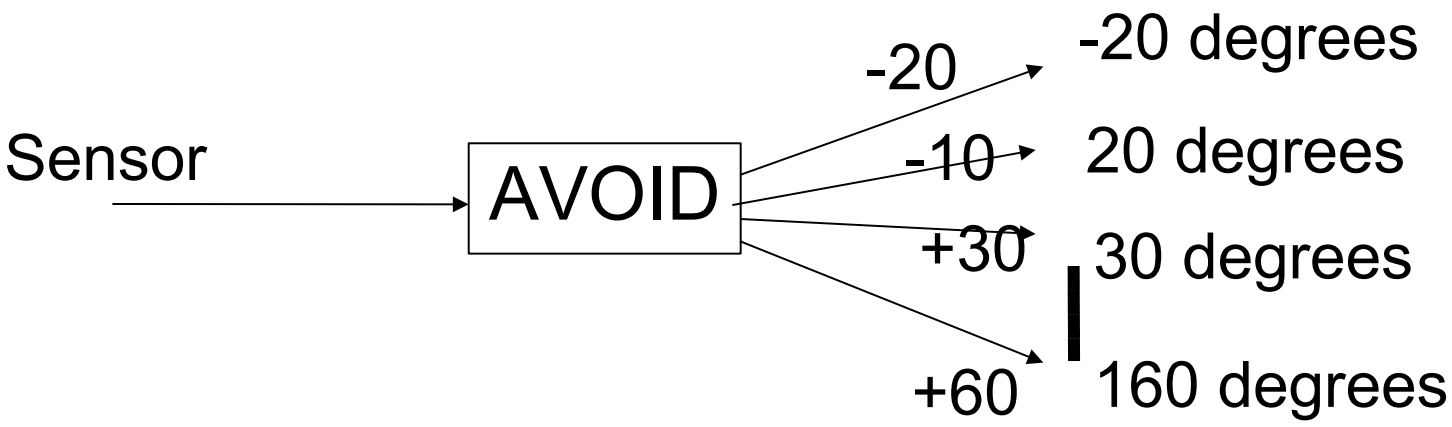


Two principles:

- Every behavior weights ALL possible commands
  - Weights are numbers [-inf,+inf]
  - “No information loss”
- Any behavior can access weights of another
  - Not just override its output
  - “no information hidden”

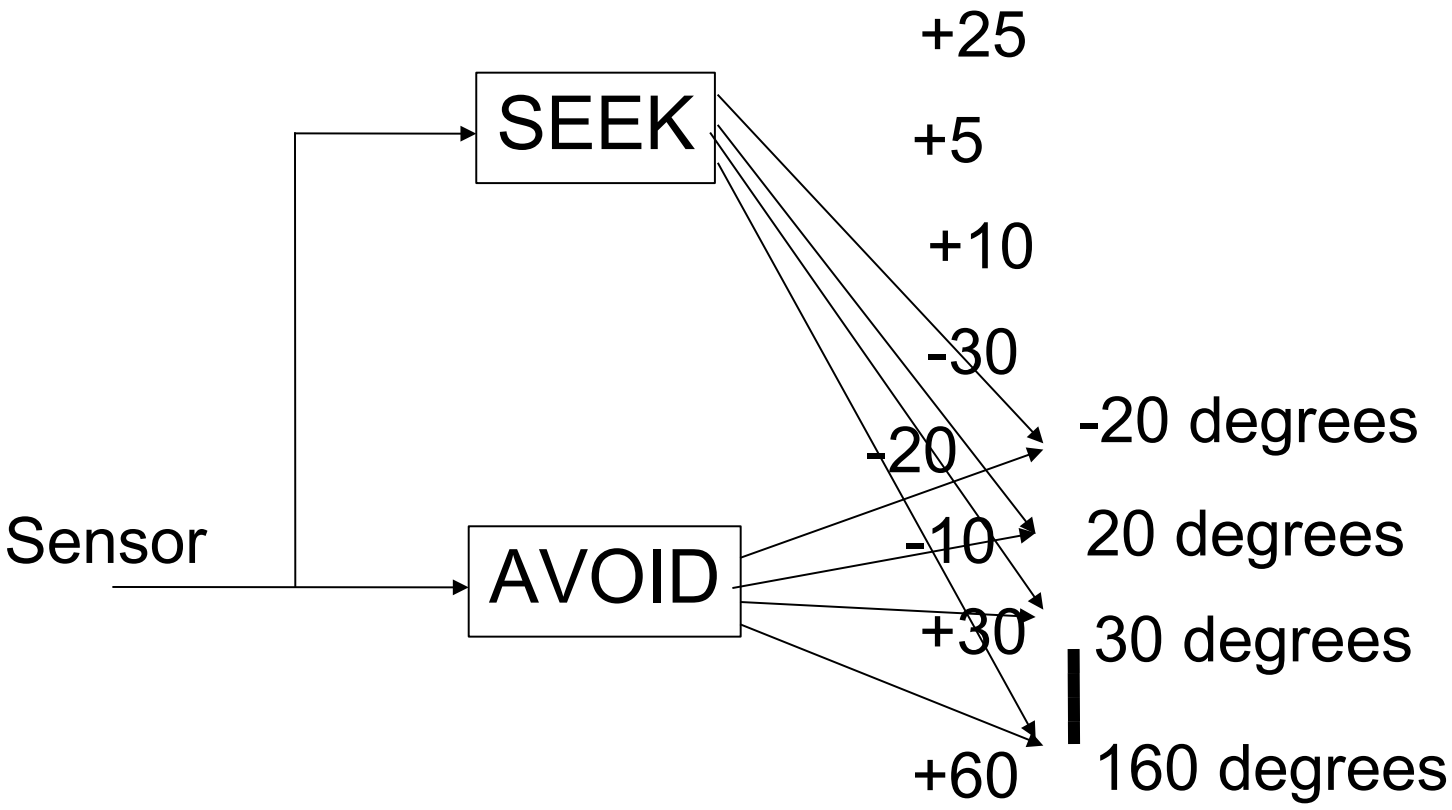
Weighting outputs

- AVOID does not choose a specific heading
- It provides preferences for all possible headings



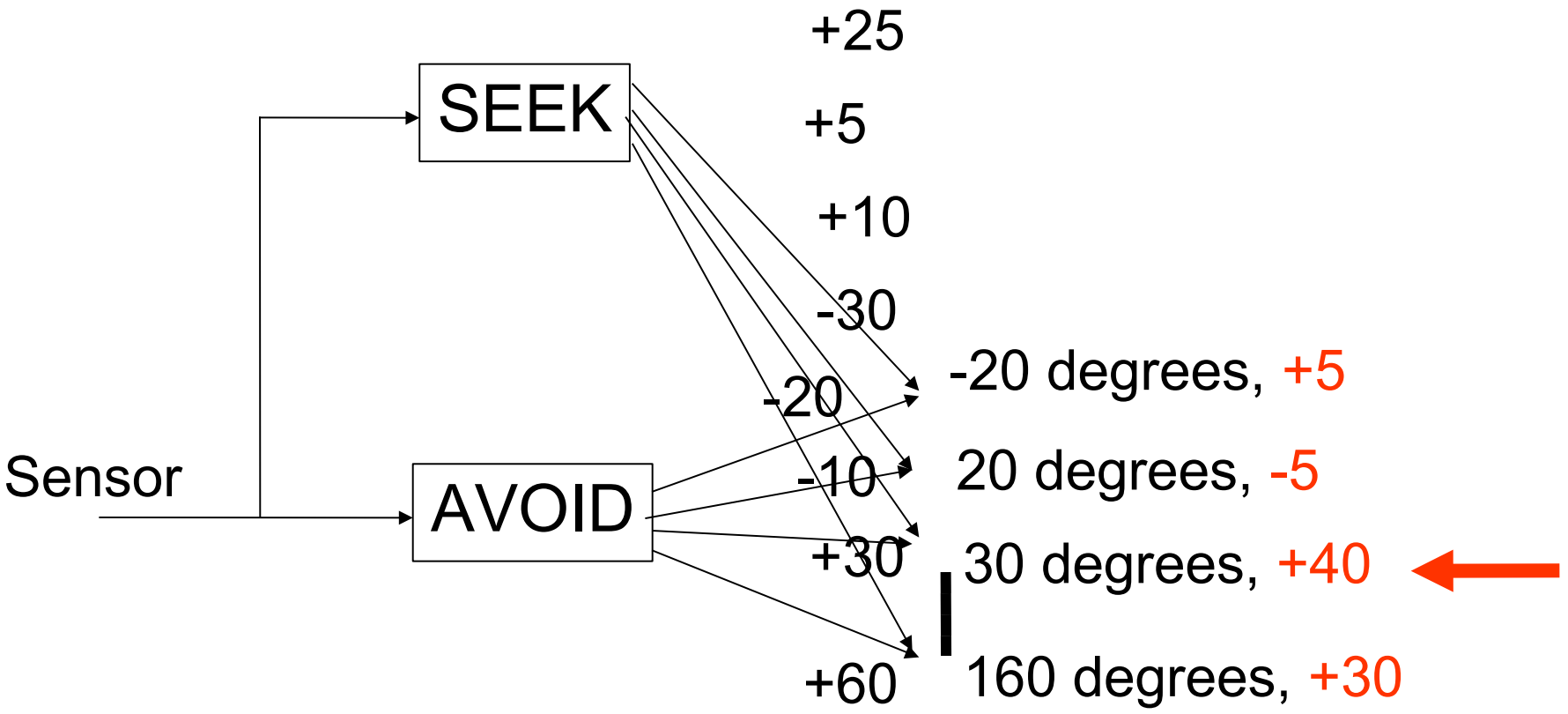
Merging outputs

- Now combine AVOID and SEEK-GOAL by adding

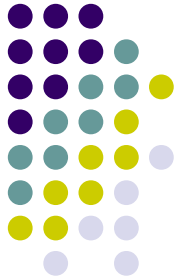


Merging outputs

- Now combine AVOID and SEEK-GOAL by adding
- Then choose top one (e.g., winner-take-all)

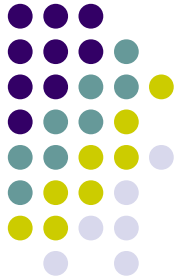


Advantages



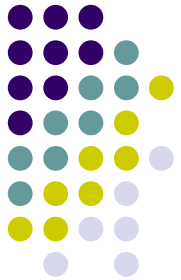
- Easy to add new behaviors that modify heading
  - Their output is also merged
- Considers all possibilities—finds useful compromises
- Negative weights possible, useful
  - Can forbid certain commands, in principle
- And….

Advantages



- Easy to add new behaviors that modify heading
  - Their output is also merged
- Considers all possibilities—finds useful compromises
- Negative weights possible, useful
  - Can forbid certain commands, in principle
- And…. Intermediate Variables

Rosenblatt-Payton 2nd Principle: No information hidden



- Not only all choices of behavior should be open
- Internal state can also be important for other layers
- In Rosenblatt-Payton, a variable is a vector of weights
  - We operate, combine, and reason about weights
  - Even as internal state variables

Example:

- Trajectory speed behavior (in article)
- Combines internal “Safe-Path”, “Turn-choices” vars

Example

Sensors → Trajectory Selection

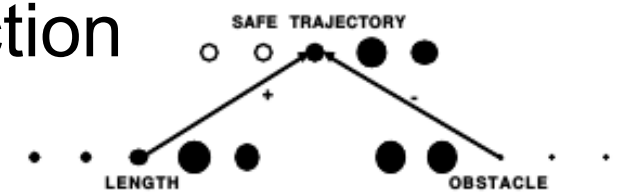


Figure 9. The combination of LENGTH and OBSTACLE inputs.

Trajectory Selection →
Safe Turn Choices
(adjacency safety)

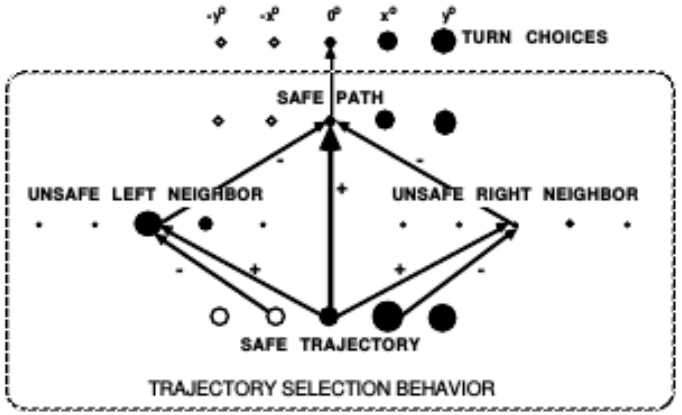


Figure 10. TRAJECTORY SELECTION behavior expresses preference for each turn choice.

Turn Choices →
Chosen Turn

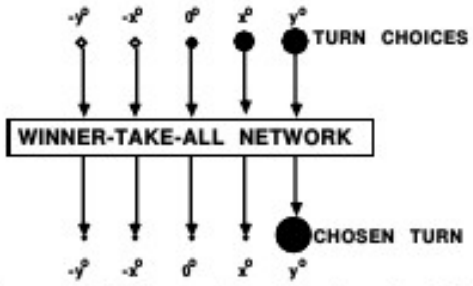


Figure 11. Winner-take-all network selects the most activated choice.

Chosen Turn + Safe Path
→ Speed
(*Note use of intermediate*
"Safe path")

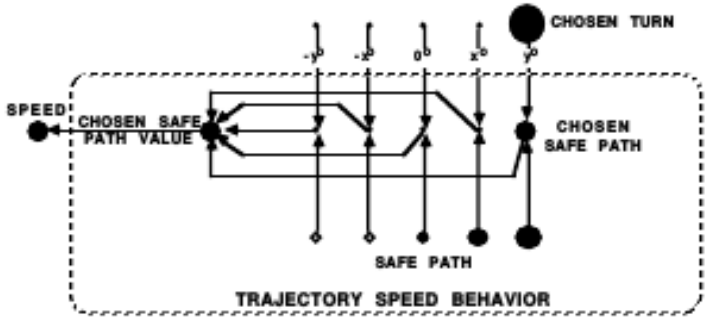
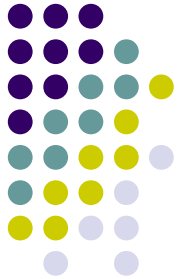
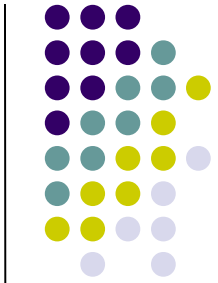


Figure 12. TRAJECTORY SPEED behavior.



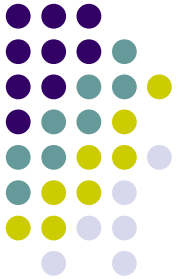
שאלות?

Potential Fields



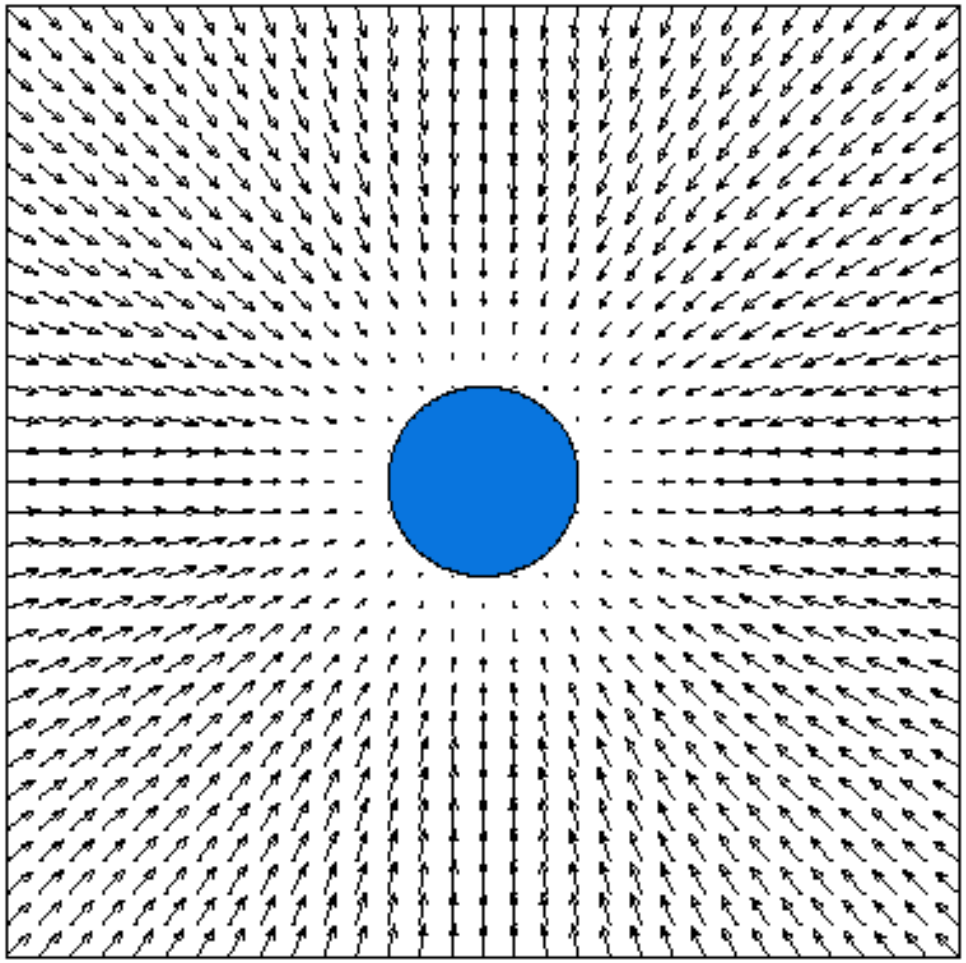
- Independently developed from Payton-Rosenblatt
- Inspired by Physics
  - Robot is particle
  - Moving through forces of attraction and repulsion
- Basic idea:
  - Each behavior is a force: Pushing robot this way or that
  - Combination of forces results in selected action by robot

Example: Goal-directed obstacle-avoidance

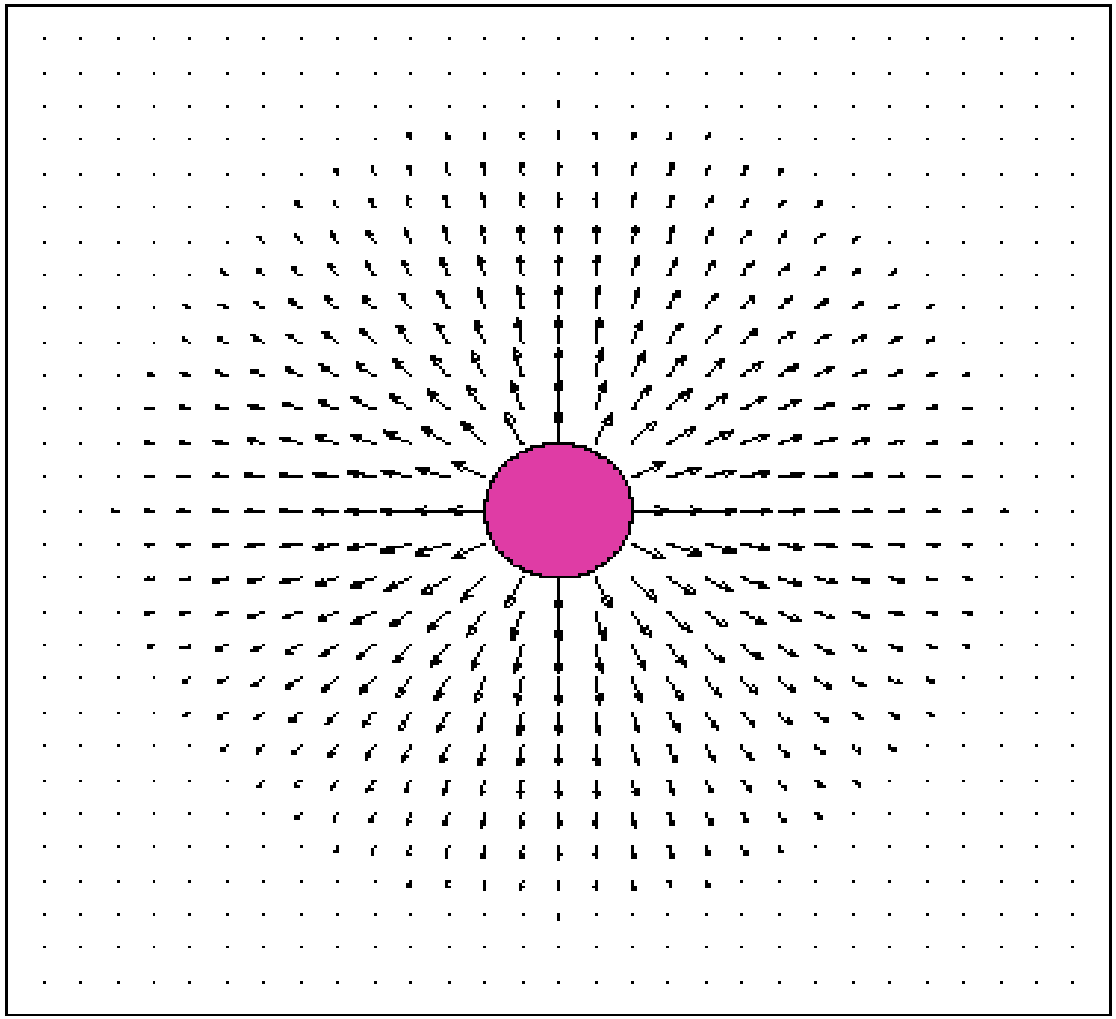


- Move towards goal while avoiding obstacles
- We have seen this before:
  - In Brooks' subsumption architecture (two layers)
  - In Payton-Rosenblatt command fusion
- Two behaviors:
  - One pushes robot towards goal
  - One pushes robot away from obstacle

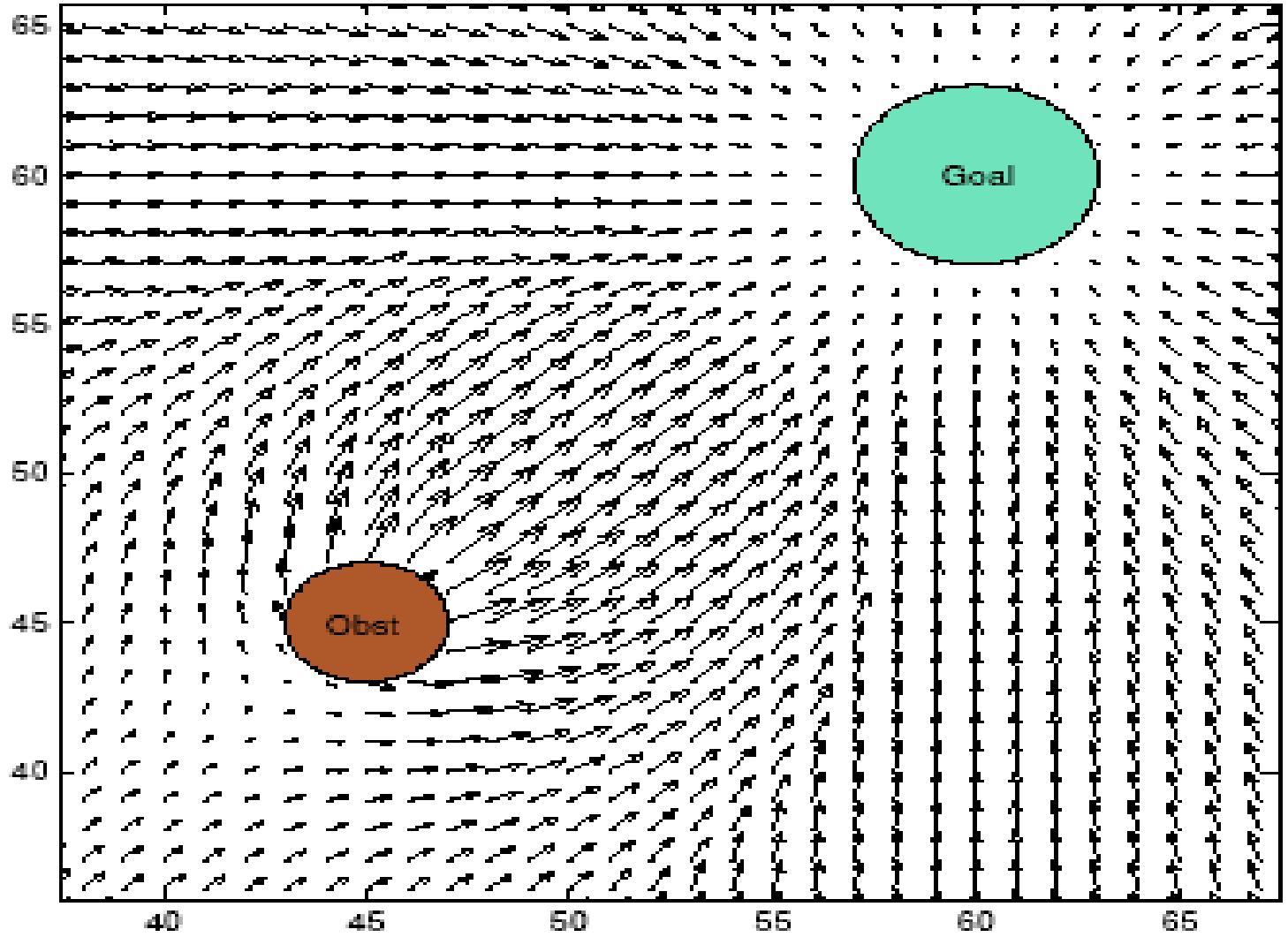
SEEK Goal



Avoid Obstacle

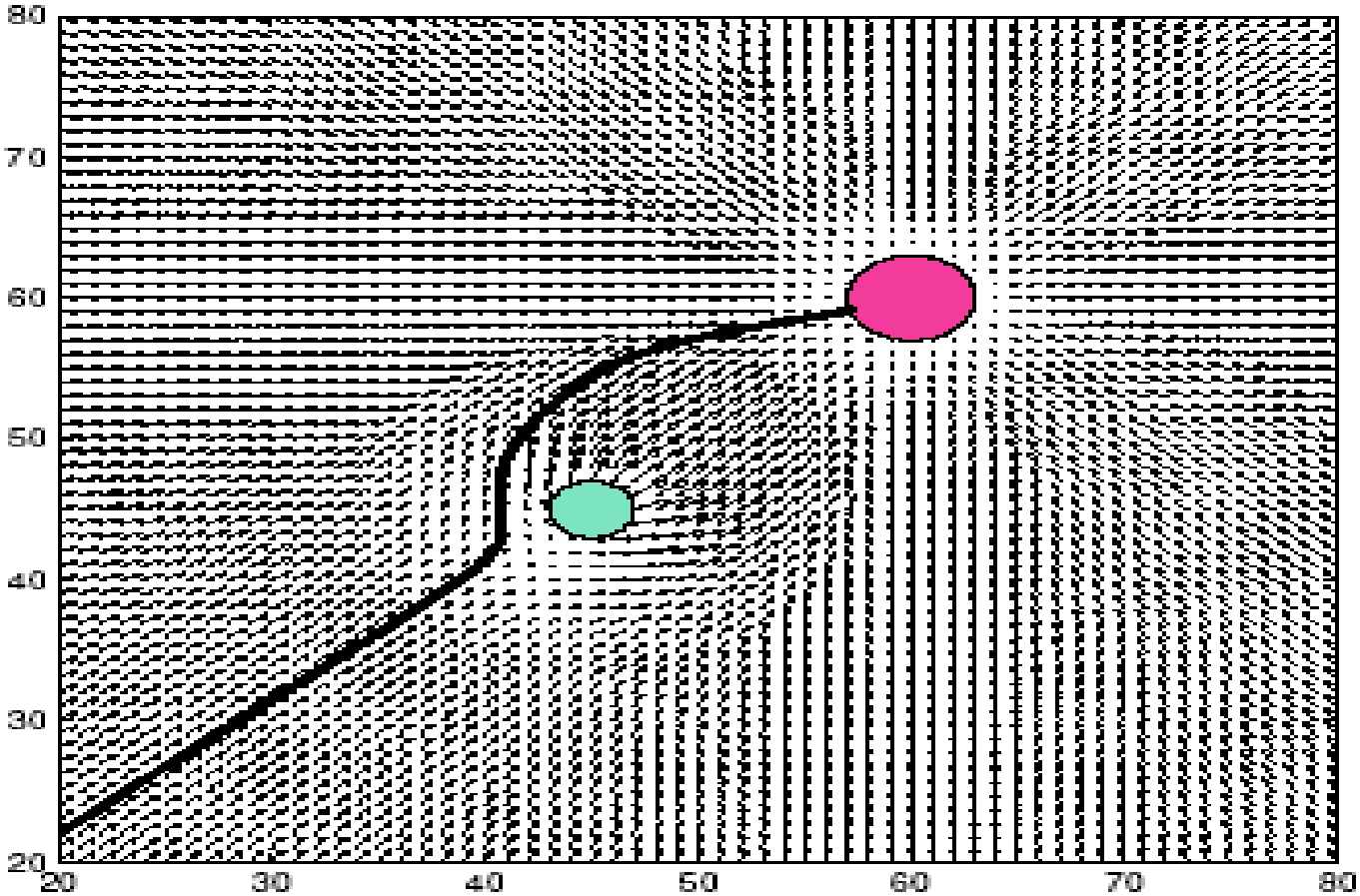


In combination....

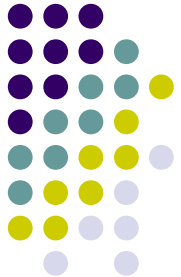


Run-time

- Robot calculates forces current acting on it
- Combines all forces
- Moves along the resulting vector

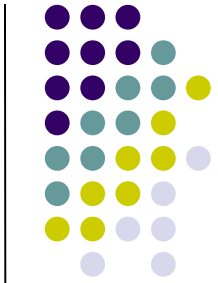


But how to calculate these fields?



- Translate a position into a direction and magnitude
  - Given X,Y of robot position
  - Generate force acting on this position (dx,dy)
- Do this for all forces
- Combine forces:
  - Final_dx = sum of all forces dx
  - Final_dy = sum of all forces dy

Example: SEEK



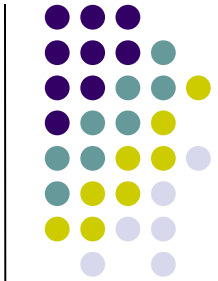
Given:

- (Xg, Yg) goal coordinates, (x,y) current position

Calculate:

- Find distance to goal d = sqrt((Xg-x)^2 + (Yg-y)^2)
- Find angle to goal a = $\tan^{-1}$((Yg-y)/(Xg-x))
- Now:
  - If d = 0, then dx = dy = 0
  - If d < s, then dx = d cos(a), dy = d sin(a)
  - If d >= s, then dx = s cos(a), dy = s sin(a)

Other types of potential fields



- Potential fiels useful in more than avoiding obstacles
- Can set in advance many different types
- Combine them dynamically

- Each field is a behavior
- All behaviors always active

Uniform potential field

- dx = constant, dy = 0
- e.g., for follow wall, or return to area

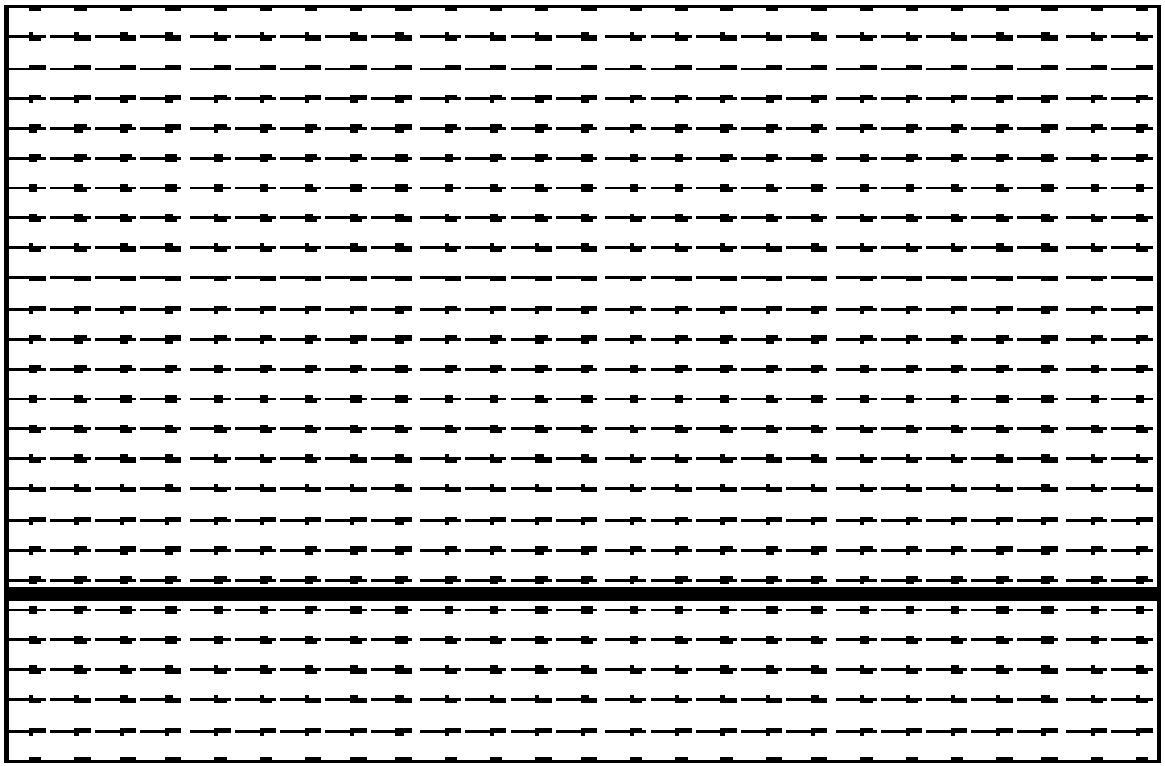


Figure 6: The uniform potential field.

Perpendicular field

- dx = +constant or -constant, depending on reference
- e.g., for avoid fence

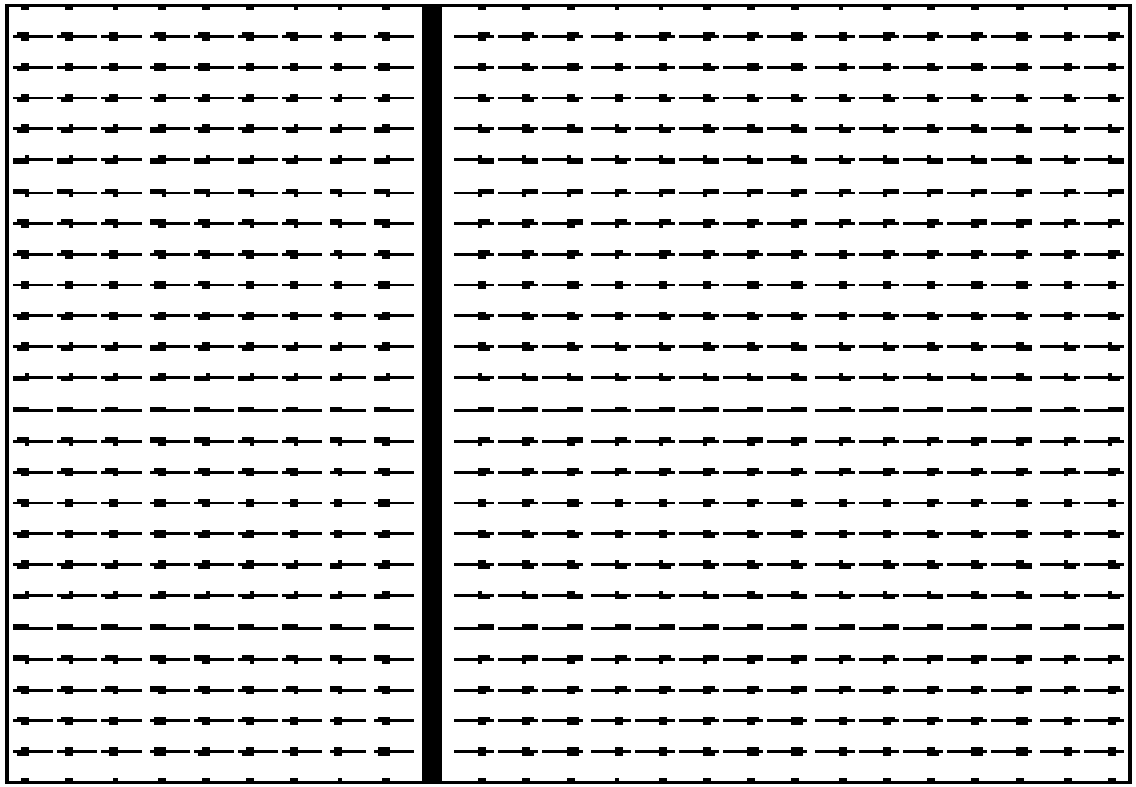
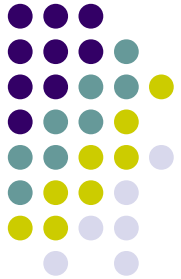
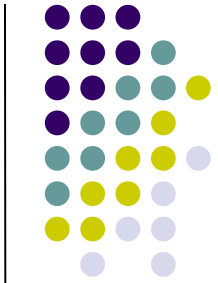


Figure 7: The perpendicular potential field.



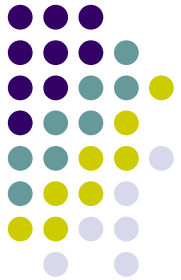
שאלות?

Problems with Behavior Fusion



- Local Minimum
- Context

Problem: Stuck!



- Can get stuck in local minimum
- All forces in a certain area push towards a sink
- Once robot stuck, cannot get out
- One solution: Random field
  - Distance d and angle a chosen randomly
  - Gets robot unstuck, but also unstable

A really difficult problem:

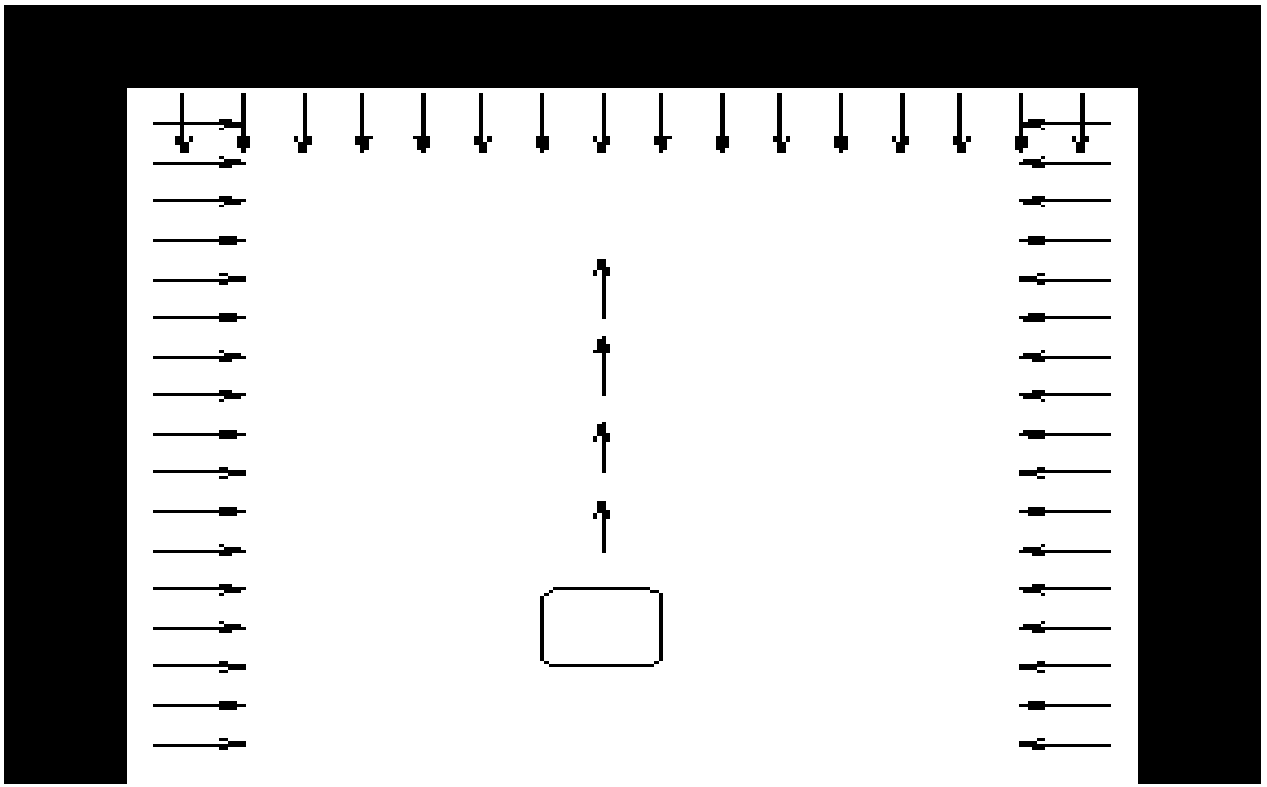
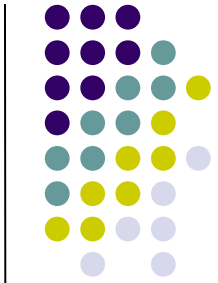


Figure 10: A world with a box canyon.

A surprising solution



- Balch and Arkin found a surprisingly simple solution
- Avoid-the-past field
- Repulsion from places already visited
- This is a field that changes dynamically

Problem: Context

Sensors → Trajectory Selection

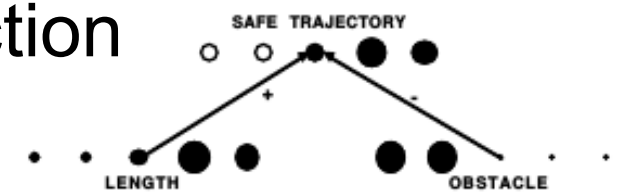


Figure 9. The combination of LENGTH and OBSTACLE inputs.

Trajectory Selection →
Additional Safety
(adjacency safety)

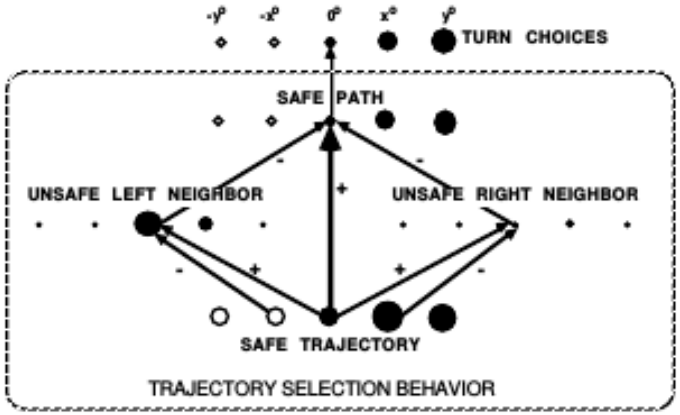
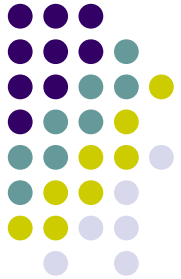


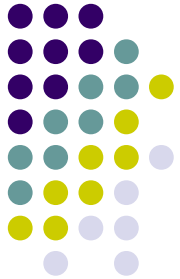
Figure 10. TRAJECTORY SELECTION behavior expresses preference for each turn choice.

Example



- Two behaviors
  - If goal is far, speed should be fast
  - if obstacle is close, speed should be stop
- Robot may not slow sufficiently
  - Because overall results compromises between stop and fast
- Options:
  - Tweak weights… Hack the numbers
  - Add distance to goal into obstacle-avoidance rules
  - Re-define fast to be slower

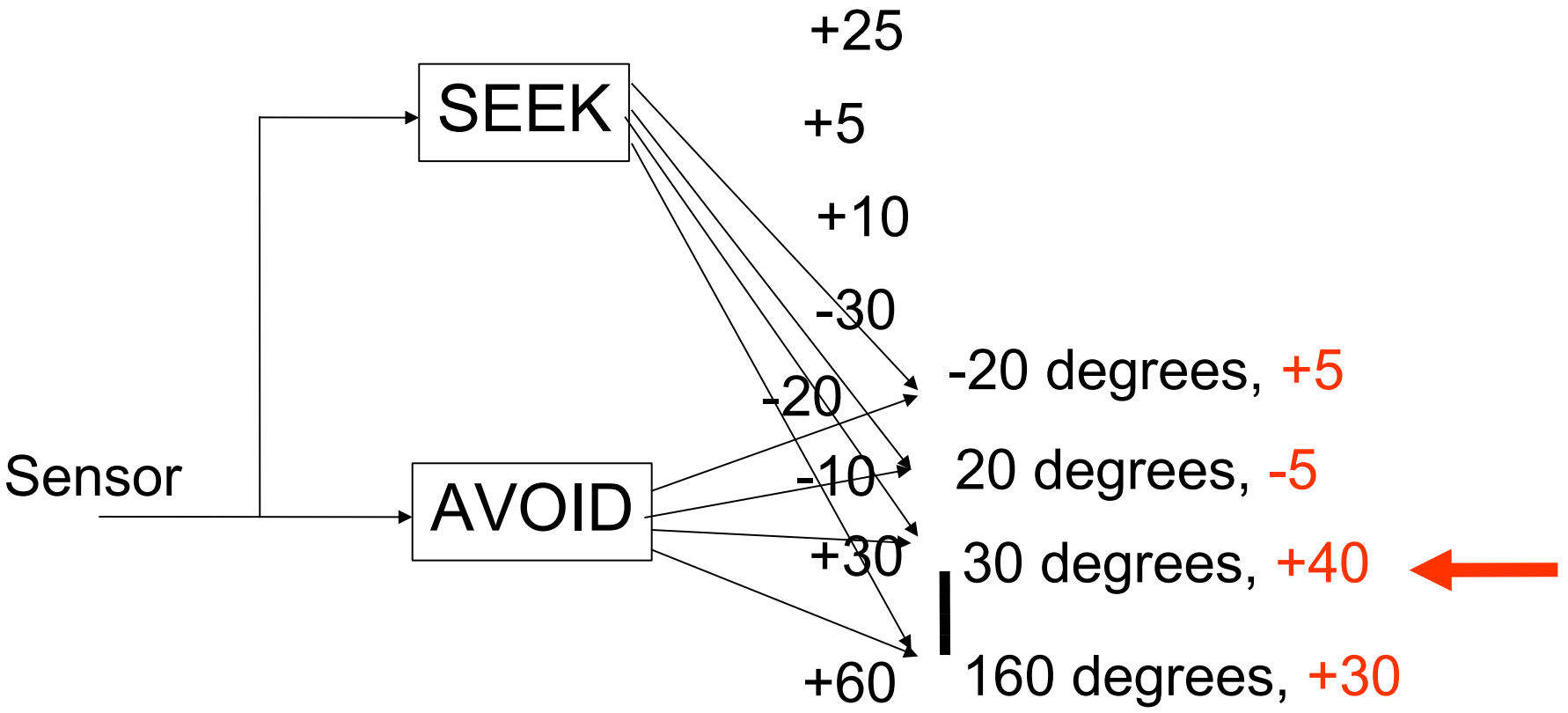
Behavior Weighting



- Use decision context to weight the behaviors
  - Dynamic weighting: change weights based on context
- Have a meta-behavior with context rules
  - IF obstacle-close THEN increase weight of AVOID
  - IF not obstacle-close THEN increase weight of SEEK
- Scale/weight based on context rules
  - Weights of AVOID, SEEK multiplied by the behavior weight

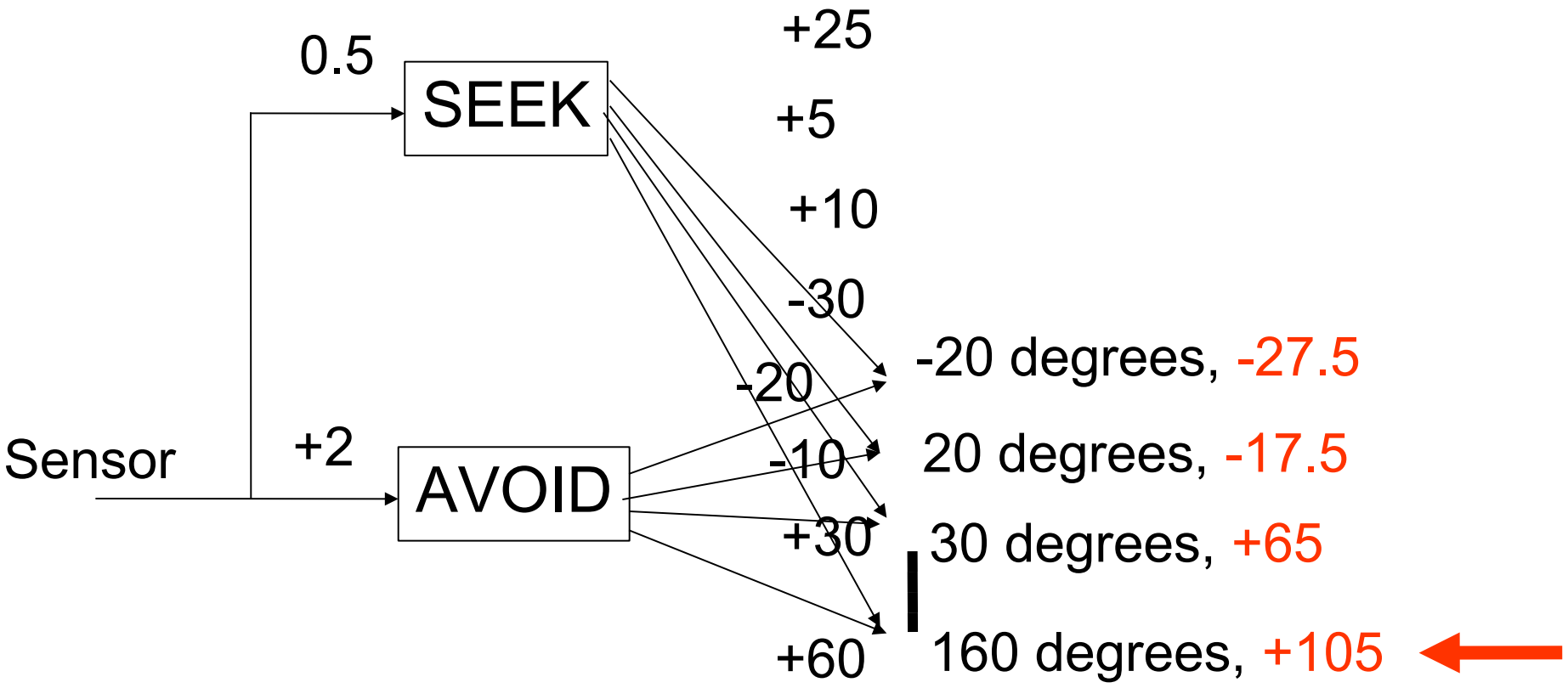
Merging outputs: Static

- Combine AVOID and SEEK-GOAL by adding
- Then choose top one (e.g., winner-take-all)



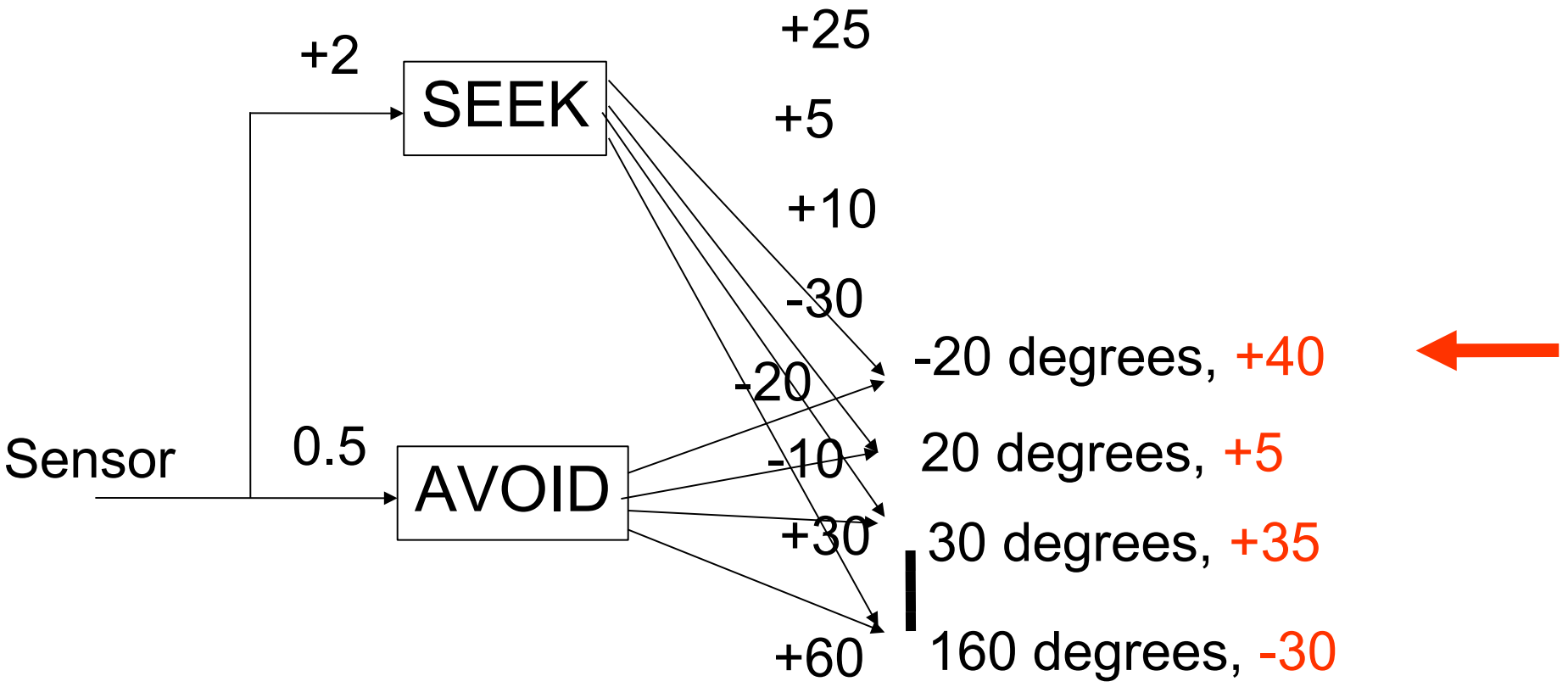
When obstacle close

- Combine AVOID and SEEK-GOAL by adding
- Then choose top one (e.g., winner-take-all)

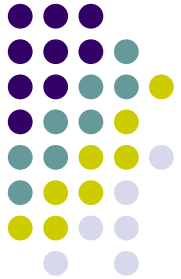


When obstacle not close

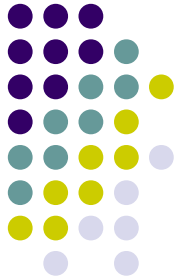
- Combine AVOID and SEEK-GOAL by adding
- Then choose top one (e.g., winner-take-all)



Final thoughts…



- Context rules combine activations and fusion
  - Activation of behavior by priority
  - Behavior has vector output (rather than single value)
  - When should we use this?
  - What about meta-meta behaviors?
- Compromises are a problem:
  - Can cause a selection of non-satisficing solution
  - Sometimes must choose!
- Compromises must be on package deals:
  - Otherwise, get behavior X on speed, behavior Y on heading!



שאלות?