COSC 4P78

Reactive Paradigm

Week 6

(Murphy and Matarić)

Brock University

Where were we?

Ah right! We were talking about how living creatures interact with the world and what a behaviour is.

- Well, we briefly talked about talking about it

So, let's get a bit of boilerplate stuff out of the way and then jump into an example.

Some quick terms

Behaviour

A close mapping between a sensor input and an actuator output. They can be divided into three categories:

- Reflexive (or Stimulus-Response / S-R) — traditional reflexes. e.g. pulling your hand away from a flame, or twitching your leg when someone smacks it with with an oversized mallet
- Reactive — learned and consolidated to be executed without conscious thought (muscle memory)
  - Note that they can still be changed by conscious thought. e.g. changing your gait to walk along a plank or around puddles
- Conscious — deliberate actions

The Reactive paradigm (what we're building up to) relies almost exclusively (or exclusively, depending on who you ask) on S-R reflexes.

Some quick terms

Reflexive behaviours

We can further categorize our reflexive behaviours.

- Reflexes — a response only lasts as long as the stimulus, and reponse is proportional to the intensity of the stimulus
- Taxes — the response is to move to a particular orientation
  - e.g. the tropotaxis of baby turtles moving towards the brightest light (the moon, hopefully), or the chemotaxis of ants following pheromone trails
- Fixed-action Patterns — the response continues for a longer duration than the stimulus
  - Imagine you're fleeing Jason Voorhees, and you close the door
  - Is he still there?
  - Make sure to remember to not call the cops or grab any good weapons

These are not necessarily mutually exclusive. If Jason sees a camp counsellor, he'll pursue them. If he loses sight of them, he'll continue lumbering in the same direction for quite some time (before somehow teleporting right behind his prey for a jump-scare).

Some quick terms

Innate Releasing Mechanisms

Sometimes a behaviour might not be so simple as *Input* $\rightarrow$ *Output*. There might be an additional requirement that some factor be present (or absent).

- A releaser is like a latch or a switch denoting whether or not an additional stimulus or condition is met
  - Thus, an innate releasing mechanism would just be a releaser innately present within the animal
- For example, releasers could include *Predator_Present*, or *Is_Hungry*
- One could also conceive of a compound releaser; simply a combination of factors. e.g. *Food_Present && Is_Hungry*

Some quick terms

Releasing mechanisms (cont)

If we wanted to, we could effectively chain mechanisms (implicitly).

```
enum        Releaser={PRESENT, NOT_PRESENT};
Releaser    food, hungry, nursed;
while (TRUE) {
    //<-Remember this spot
    food=sense();
    hungry=checkStateHunger();
    child=checkStateChild();
    if (hungry==PRESENT)
        searchForFood(); //Sets food to PRESENT when done
    if (hungry==PRESENT && food==PRESENT)
        feed(); //sets hungry=NOT_PRESENT when done
    if (hungry==NOT_PRESENT && parent==PRESENT)
        nurse(); //Sets nursed=PRESENT when done
    if (nursed==PRESENT)
        sleep();
}
```

Some quick terms

Releasing mechanisms (still cont)

What would that behaviour do if not hungry? It'll just sit around waiting to get hungry.

If we also wanted to consider predators, we could try adding:

```
predator=sensePredator();
if (predator==PRESENT)
      flee();
```

But, is that good enough? How would it actually react to a predator?

What we need is a way to *inhibit* other behaviours while fleeing.

Concurrent Behaviours

So, let's set aside inhibition for the moment and address a more basic desire: managing all of these concurrent behaviours.

- Because that'll be a common factor in designing a reactive architecture: individual behaviours are typically treated as acting in parallel

Sometimes, releasers might be enough to allow independent concurrent activities to still operate appropriately in sequence. But... not always.

Concurrent Behaviours

Unusual behaviours

- Equilibrium — consider a squirrel approaching a friendly hoomon on a bench.
  - I should run away!
  - Though he does have tasty-looking peanuts...
  - Result: just stand there
  - The two behaviours achieve a balance, and he just... sits there
- Dominance — suppose you're both hungry and sleepy
  - I need a sammich
  - I need to go take a nap
  - Hopefully, one wouldn't attempt both simultaneously (pick one!)
- Cancellation — Male sticklebacks (fish), when their territories overlap and they need to either defend their territory or attack the other fish... revert to building a new nest
  - The two stimuli appear to cancel each other out, so they just defer to the only remaining home-related stimulus

Perception

A releaser isn't actually enough to guide a behaviour.

- THERE'S A PREDATOR!
  - THAT MEANS RUN!
    - ★ ... where?
    - ★ RUN!

Often, you also require a *guide* for the behaviour. So, perception will provide cursory facts like the presence/absence of a release, but also additional information where relevant.

This brings us to *affordances* — *"perceivable potentialities of the environment for an action"*.

- So, basically, just a formal way to define potential external stimuli, like releasers or guides for behaviours
- For example, baby arctic terns can recognize a parent's red bill as a source of food (meaning it can feed if the releaser of being hungry is also present), but that red bill can also be a guide (because the baby knows to feed in the direction of the bill)

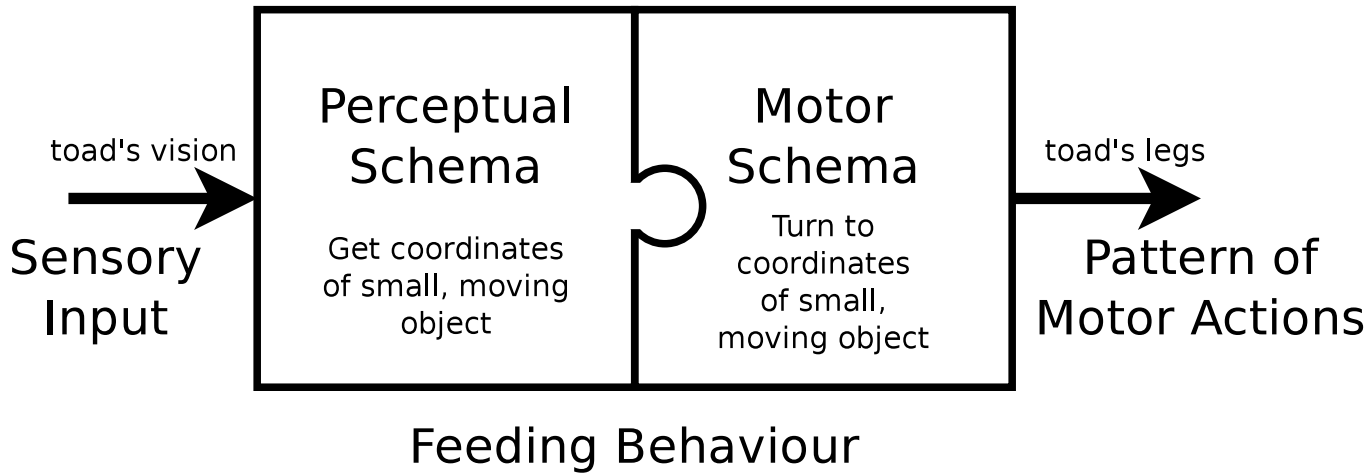
Perception

This should go without saying...

Perception is actually an immensely complicated supertopic, that we couldn't possibly fully address even with a full lecture on its own. Even machine vision (just one subtopic) couldn't be understood from a single complete lecture (or likely even course).
So, we're mostly just interested in knowing the role of perception as a stimulus to trigger or guide behaviours, and we'll be mostly limiting ourselves to those percepts that we need in the short term.

Schemas

Schemas are also slightly outside the scope of this course, but for the sake of a little perspective, consider the following example of the behaviour of a toad feeding:



If we wanted to *inhibit* such a schema, then we would do so by either suppressing the input, or inhibiting the output.

The Problem

So, great. We could probably model any simple reflexive behaviour in isolation. And the moment we needed more than one, we'd... deal with it. Somehow.

Basically, we need to more formally address how we can coordinate multiple independent (and simultaneous) S-R behaviours.

The Reactive Paradigm

Of course, we've largely already been discussing the reactive paradigm. All we need to do now is decide how to select (or combine) multiple behaviours; particularly when some may be mutually exclusive.

There are two basic approaches to handling *action selection* (choosing which of several commands to actually effect):

- Command arbitration — the selection of a single action/behaviour, to the exclusion of all others
  - I really want to go get that red ball, but my batteries are getting low and I really need to recharge them...
- Command fusion — the combination or integration of multiple behaviours into a single output action/behaviour for the robot
  - This could be two related tasks like balancing and walking simultaneously, or unrelated like walking and chewing gum

Potential Fields

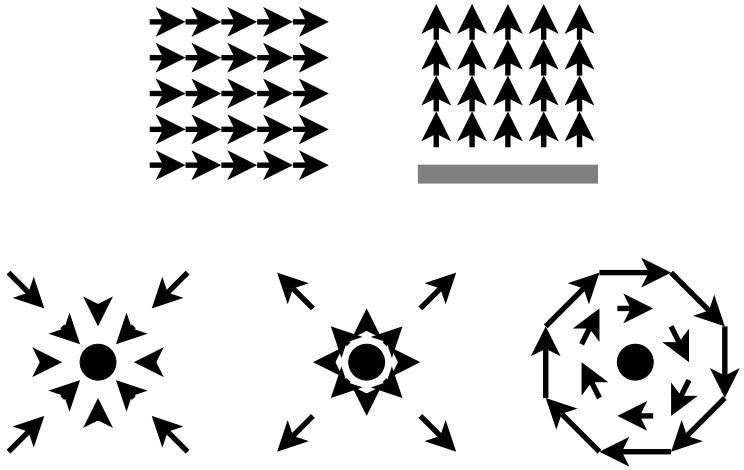
This is going to be more of a thought experiment than a complete explanation, but it's worth looking at.
Suppose you could superimpose a force field onto an environment. Any actors within that field would naturally be pushed into the prescribed direction.

- If it helps to visualize, imagine you wanted to make a marble move on a suspended blanket, and had the ability to pull down or push back up at arbitrary points along it

If you were to map out the field for the entire room/area, you could easily tell how the actor would move from any single point within the space, right?

Potential Fields

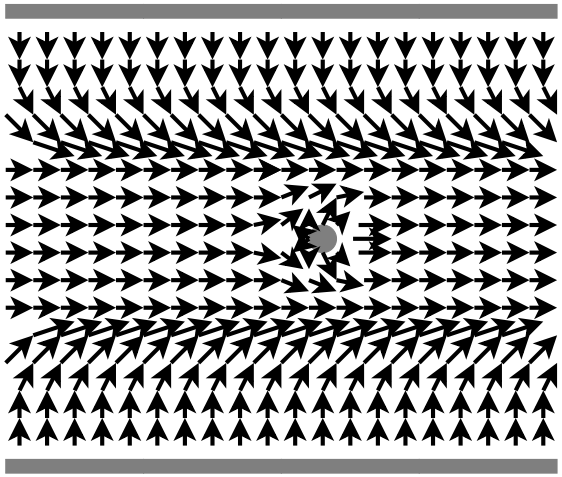
Primitive Fields



- Uniform — good for corridor following
- Repulsion — runaway/obstacle avoidance
- Attraction — move towards goal
- Perpendicular — corridor following
- Tangential — doors, docking, etc.

Potential Fields

A very simple example

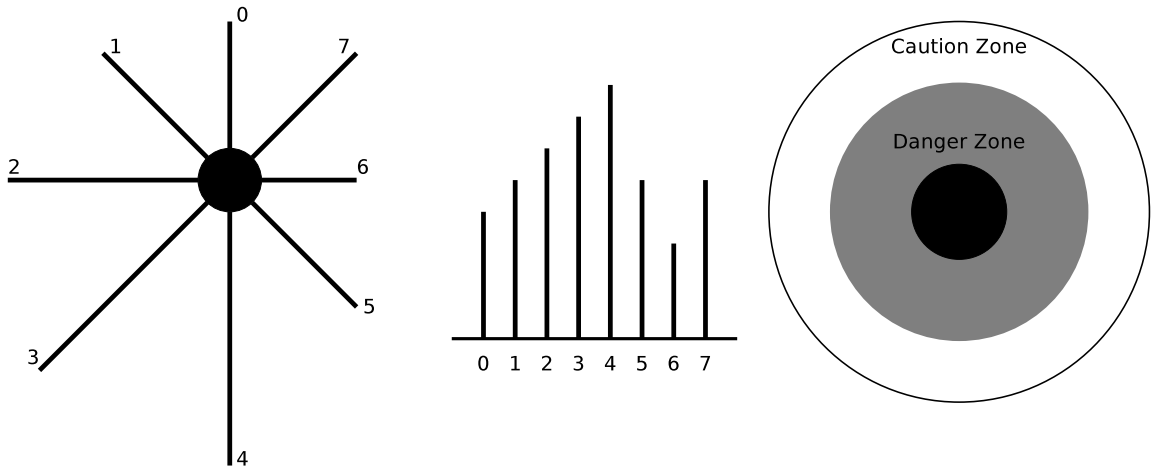


How could we use something like this for doors or docking?
How practical are potential fields, *really*?

Interjection!

Example time!

Before we get to arbitration, let's first briefly discuss one possible example: a robot with a continuous polar plot from 8 ultrasonic sensors.



We have two desired behaviours, so we'll discuss two controllers.

First Controller

Not smacking the wall

```
(case
    (if (minimum (sonars 0 1 7)) <- danger zone
        and
    (not stopped)
then
    stop)
(if ((minimum (sonars 0 1 7)) <- danger zone
        and
    stopped)
then
    move backward)
(otherwise
    move forward))
```

Second Controller

Avoiding obstacles while it's still easy

```
(case
    (if ((sonar 7 or 6) <= caution-zone
        and (sonar 1 or 2) >= caution-zone
    then
        turn left)
    if (sonar 0 or 1 or 2) <= caution-zone
        and (sonar 6 or 7) >= caution-zone
    then
        turn right)
)
```

Two Controllers, One Robot

Hmm...

It's easy to see why some form of action selection is necessary, right?

Avoiding a wall while moving forwards is one thing, but *stop* and *turn* might not play as nicely with each other.

- Of course, the devil's in the details. It's possible the two could be *fused* into *turn in place*

Subsumption

The theory behind subsumption is pretty simple:

- Create individual layers to address each task you wish the robot to perform
  - Ranging from low-level (e.g. piloting) to high-level (determining the next goal)
- A higher-level behaviour *subsumes* a lower-level by one or both of:
  - Suppressing its inputs — either nullifying them, or replacing them
  - Inhibiting its outputs — replacing or augmenting them

Subsumption

(Pre-example)

(We have plenty of time to talk about two-wheeled balancing bots, right?)

Subsumption

For example...

Let's look at trying to map out an area:

1. Mapping
2. Wandering
3. Avoiding
4. Running Away
5. Halting

Let's have a chat?

Subsumption

Final Thoughts

Actual subsumption is a bit more complicated than this.

Probably most importantly, the different layers can share sensor inputs across them.

- So, for example, our *avoidance* and *run away* controllers would both be complete controllers, with potentially the same sonar input
- This means that, even if a layer is being suppressed, it's still doing the work
  - By its very nature, subsumption is a *tasked* (or concurrent) architecture

One Last Thought

Example time! (again)

Let's consider the basic example of a robot with two whiskers. We want it to follow a wall. How hard would that be?

- If left switch triggers, go right
- If right switch triggers, go left
- If both switches trigger, back up

That should work... right?

... right?

One Last Thought

State

It's worth noting, just as an extra, that you can add additional state information, and yet still keep it reactive.

One easy mechanism is to simply maintain an additional flag or two. In this case:

- A flag remembers the last action taken
- If it *would* go left after it just tried right, then it should do something else
  - e.g. turn *far* to the left

You can keep it reactive by simply using the flag as a virtual input (i.e. just one more piece of data coming in with the sensors).

However, don't go overboard with this. Every such flag you add also adds quite a bit to the table, and adds further complexity to your behaviours.

- Suppose you were using a lookup table
- Each flag adds an extra *dimension*!

The Reactive Paradigm

So...

So... Reactive is good?

Well, in some ways it's *better*.

- We don't need to worry about the *frame problem* so much any more
- We can have *emergent behaviours*
- It's more natural and intuitive to design the individual modules

On the other hand, planning? Yeah... that's the funny thing. When you discard proper planning, you end up with... no proper planning.

- Also, while *Deliberative* let the robot "work out all the details" (requiring that it *know* pretty much everything about the situation), *Reactive* requires that the *roboticist* have every combination of inputs anticipated for in advance
- But neither is really practical for a comprehensive problem, right?

Ah dernit.

Next week!

Questions?

Comments?

Favourite pizza toppings?