3806ICT - Week 3 Lab

Nick van der Merwe - s5151332 - nick.vandermerwe@griffithuni.edu.au

April 22, 2021

Question 1

Give definitions of locomotion and manipulation. What are their shared features and differences?

Locomotion is defined as a robot's ability to move itself by exerting force on the environment whereas manipulation is its ability to move objects by exerting force upon them.

Question 2

What are advantages and disadvantages of legged robots?

Easiest format to see this in would be lists:

Advantages

- They can go over more complicated obstacles without getting stuck (slanted ground, steps, et cetera)
- Causes less damage to terrain than wheeled robots

Disadvantages

- Movement speed
- Complexity - actuators and structure are a lot more complicated
- Harder to control - must consider balance and stability
- Less energy efficient due to:
  - Terrain
  - Centre of gravity moves while walking
  - Picking up the legs

Question 3

What is DOF? If a robot can only move forward and backward, how many DOFs does it have? In most cases, how many DOFs does a robot leg has?

DOF stands for **d**egrees **o**f **f**reedom, and its defined by the number of joins in each leg. To have a leg that only moves forwards and backwards, it would have two joints: this is because its limited to doing a lift and swing motion. To move backwards it just swings in the other direction than normal. Most robot legs have three joints.

Question 4

What is a gait of a legged robot? Enumerate all lift and release events of a robot with 4 legs. Give two examples of gaits for such a robot.

Our formula to find how many states there are is $2^k = 2^4 = 16$ states.

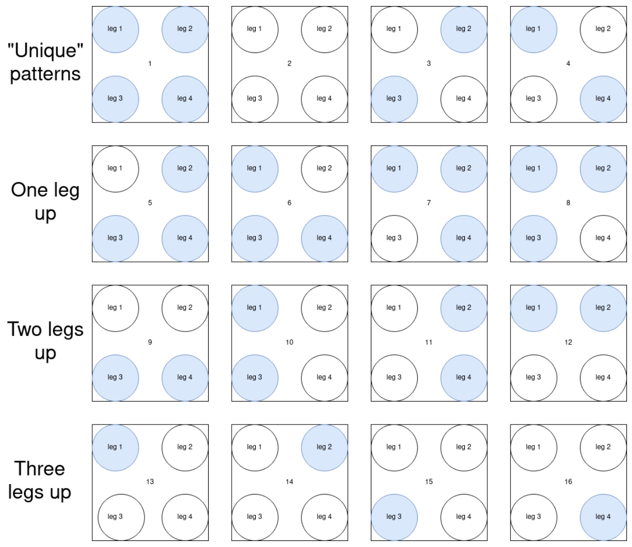
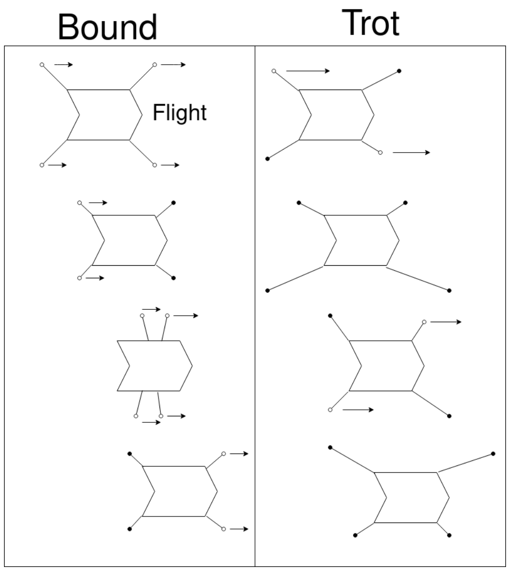


Figure 1: Blue means a leg is down and white is the leg is up

Now to depict two gaits of a four legged robot:



Question 5

Formulate the Monkey and Banana Problem in STRIPS:A monkey is at location A in a lab. There is a box in location C. The monkey wants the bananas that are hanging from the ceiling in location B, but it needs to move the box and climb onto it in order to reach them

States

- Initial State MonkeyLocation(A), MonkeyLevel(low), BoxLocation(C), BananasLocation(B),
- Goal state Have(bananas)

Actions

- _Move(X, Y)_
  - Pre: Box(X), Location(?Y) HandsFree(), MonkeyNOTOnBox(?X), MonkeyLocation(?Y), At(X, Y)
  - Effects+: Carrying(?X),
  - Effects-: FreeHands(), At(?X, ?Y)
- _MoveBox(FromPlace, ToPlace)_
  - MonkeyLocation(FromPlace), BoxLocation(FromPlace), MonkeyLevel(low)
  - BoxLocation(ToPlace), not BoxLocation(FromPlace), MonkeyLocation(ToPlace), not MonkeyLocation(FromPlace)
- _ClimbUp(Place)_
  - AtLocation(Place), BoxAt(Place), MonkeyLevel(low)
  - MonkeyLevel(high), not MonkeyLevel(low)
- _TakeBananas(Place)_
  - MonkeyLocation(Place), BananasLocation(Place), MonkeyLevel(high)
  - Have(Bananas)

Question 6

Evaluate the subsumption architecture in terms of: support for modularity, niche targetability, ease of portability to other domains, robustness

There are two perspectives on whether subsumption has good modularity. Firstly, its granted hea subsumption is a form of the reactive paradigm and its main architecture revolves around modules, it can be considered as highly modular. This is especially visible as modules are layered and reflects parts of object oriented programming as they have the ability to override, subsume and contain one another (when they're layered). Only thing that's arguably missing is inheritance, but that can probably be simulated by layering as well.

The counter argument to this is that its very rarely taken advantage of in practice when subsumption is used [Tang, 2017]. Overall, one would say there's more argument that its modularity is high as it's capable of many OOP properties - even if it's rarely used.

The niche targetability is high as we can design the system in a very specific way depending on the situation. For instance, it can compile down onto a programmable-array logic circuit [Tang, 2017].

For portability, due to the fact that the system operates in layers if we tried to adjust an existing robot to a different application chances are that it would take too much effort to do. As in, if there's a fundamental behaviour in layer 0, or even the second last layer that we want to change it could be a lot of effort. On top of this, adapting to picking different actions based on stimulus is challenging For this reason its portability is quite low, which is different from a reactive paradigm. [Murphy, 2000]

Overall its robustness to failure is arguable. Sensors failing is usually detected, and can be countered by having more than than the minimum number of sensors. Furthermore, the lower layers will continue to operate when higher ones are added, however, the problem arises at high layers. These can only inhibit and suppress lower layers as they actively interfere with them, and therefore they can cause undetected errors in the robot. Due to this its robustness is typically evaluated as low because there is no inbuilt method to identify these failures. [Murphy, 2000] [ShippensburgUniversity, 2007]

In summary, it's modularity is good since it has many OOP functions and the niche targetability is high as it can be programmed on very low level systems. It's main weakness out of the four is its ease of portability to different uses, and its robustness to failure is generally considered low.

Question 7

Describe the Hybrid paradigm in terms of: (a) sensing, acting, and planning, and (b) sensing organization.

Basically the planning is done on its own, and the sensing and acting are done together. That's to say that information, sensed or cognitive, is put through the plan paradigm and that gives us directives. These directives allow our sense-act section to decide what to do when it senses information. Its 'hybrid-ness' is visible through this, as it introduces planning from other architectures into a reactive architecture's sensing and acting. This assists with the downside of the subsumption model having difficulty adapting different actions to senses that are similar. The specific way P, SA acts is visible in figure 2.

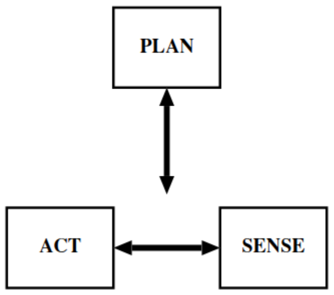


Figure 2: The plan, sense act model for hybrid architecture

The sensing organisation also merges the hierarchical and reactive styles as both planning and acting requires the sensory information. In the model this causes it to be more complex than the hierarchical one, our global world model can have its own sensor and the behaviours can selectively listen to sensors while being able to create virtual sensors. An example is visible in figure 3.

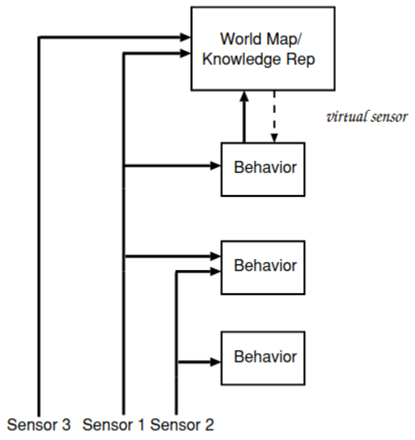


Figure 3: Sensing organisation of hybrid paradigm. Taken from [Murphy, 2000]

Question 8

Look up technical reports on Shakey. Compare Shakey with the Hybrid architectures. Now consider the possible impact of the radical increases in processing power since the 1960's. Do you agree or disagree with the statement that Shakey would be as capable as any Hybrid if it were built today? Justify your answer.

The overwhelming issue with shakey in terms of its architecture is the fact that it effectively closed its eyes when it was not in the sensing stage of its plan-sensing-acting paradigm, which would happen to be majority of the time [Murphy, 2000]. This is due to both the hardware on board (SDS 940 computer [Nilsson, 1984]) and that the STRIPS paradigm is slow. For reference, a SDS 940 computer has a whopping maximum main of 64 kilowords and is old enough that its processing speed was measured by microseconds to add numbers, not hertz.

To try to convert it, it takes 77 microseconds on the SDS 940 to a 24 bit floating operation. In other words it has (1/0.000077 = 12987 FLOPS(24 bit)(floating pointer operations per second)) [Systems, 1965]. On the other hand, an AMD Ryzen Threadripper 3990X has a whopping 13,209GFLOPS, or 13209000000000 FLOPS (32bit) [Chiappetta, 2020]. Note that later on Shakey was updated to a PDP-10 [Nilsson, 1984], but that's still a similar comparison that its effectively millions of times slower than today's technology.

Not to mention, in terms of the planning algorithmic speed, the hybrid paradigm has O(n) algorithms [Murphy, 2000] whereas STRIPS planning varies depending on the situation, but will always be PSPACE-complete; that is, it will never exceed $O(n^k)$ in complexity.

However, even if we manage to keep up with STRIPS's much higher demand for processing, the integral problem with it still exists: its single threaded. A hybrid can see a situation, plan it, and while its acting if it goes wrong it can replan to react to the situation. On the other hand, STRIPS would finish acting out its plan then realise something went wrong. For this reason, the STRIPS model would have to be changed to compete with hybrid robots.

References

[Chiappetta, 2020] Chiappetta, M. (2020). Amd threadripper 3990x review: A 64-core multi-threaded beast unleashed. `https://web.archive.org/web/20200318200417/https://hothardware.com/reviews/amd-ryzen-threadripper-3990x-cpu-review?page=3`.

[Murphy, 2000] Murphy, R. R. (2000). Introduction to ai robotics. `https://www.researchgate.net/file.PostFileLoader.html?id=53ad1225d4c118ad118b4705&assetKey=AS%3A272118967668741%401441889625514`.

[Nilsson, 1984] Nilsson, N. J. (1984). Shakey the robot. `https://ai.stanford.edu/~nilsson/OnlinePubs-Nils/shakey-the-robot.pdf`.

[ShippensburgUniversity, 2007] ShippensburgUniversity (2007). Reactive paradigms basics. `https://web.cs.ship.edu/~djmoon/roboteam/robotics-notes/murphyc4.pdf`.

[Systems, 1965] Systems, S. D. (1965). Sds 940 time-sharing computer system.

[Tang, 2017] Tang, F. (2017). Cs512: Robotics. `https://www.cpp.edu/~ftang/courses/CS521/notes/reactive%20architecture.pdf`.