Modularity for Genetic Programming

Anil Kumar Saini

PhD Student

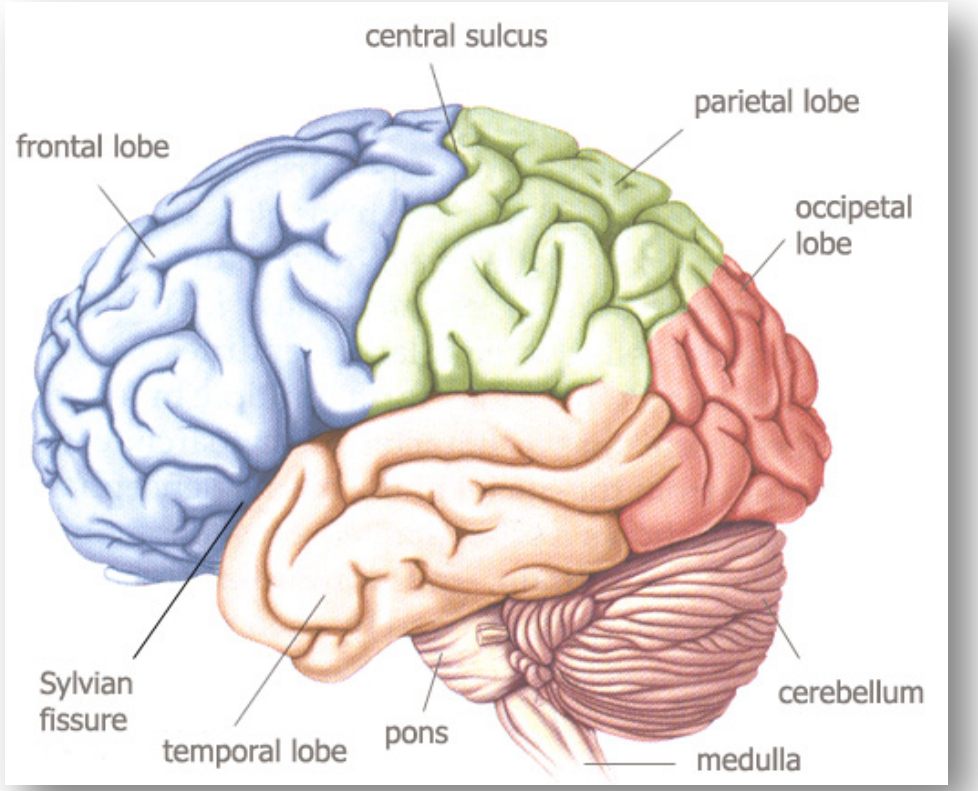
CICS, UMass Amherst

Outline

- Definition of modularity in different domains
- Advantages of modularity
- Modularity in GP systems
- Defining modularity for evolved programs
- Open problems

Modules in Biology

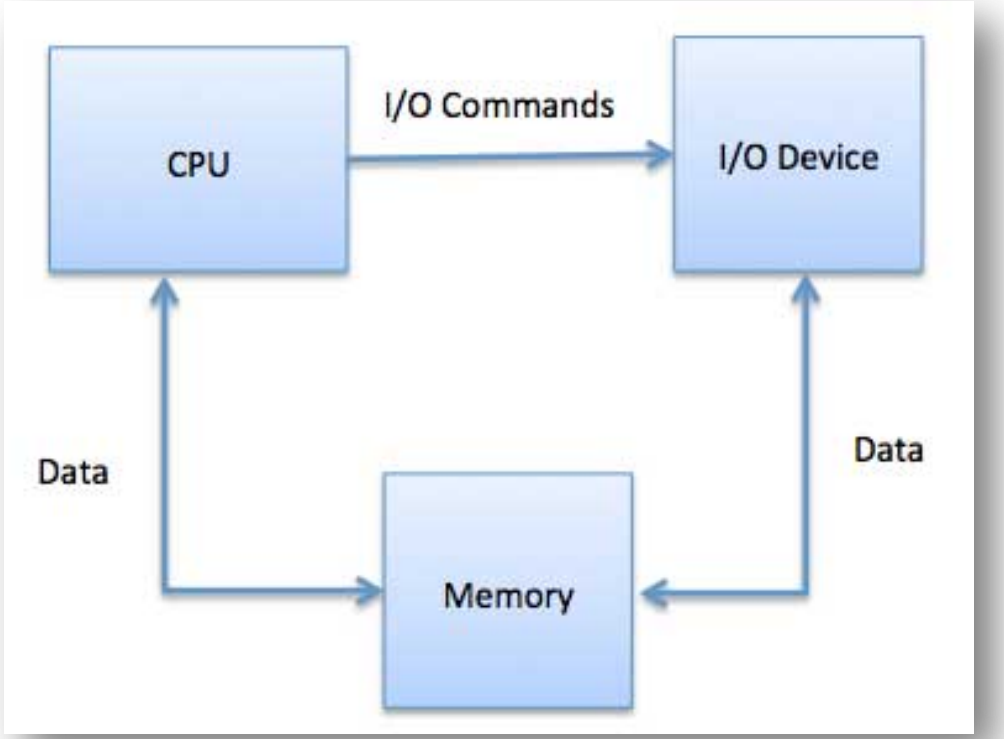
- *A module can be viewed as a semi-autonomous entity that evolves, functions or participates in development (or other processes) relatively independently from other modules*[1].
- Example, modules in human brain, gene regulatory networks, etc.



Modules in Human Brain

Modules in Software

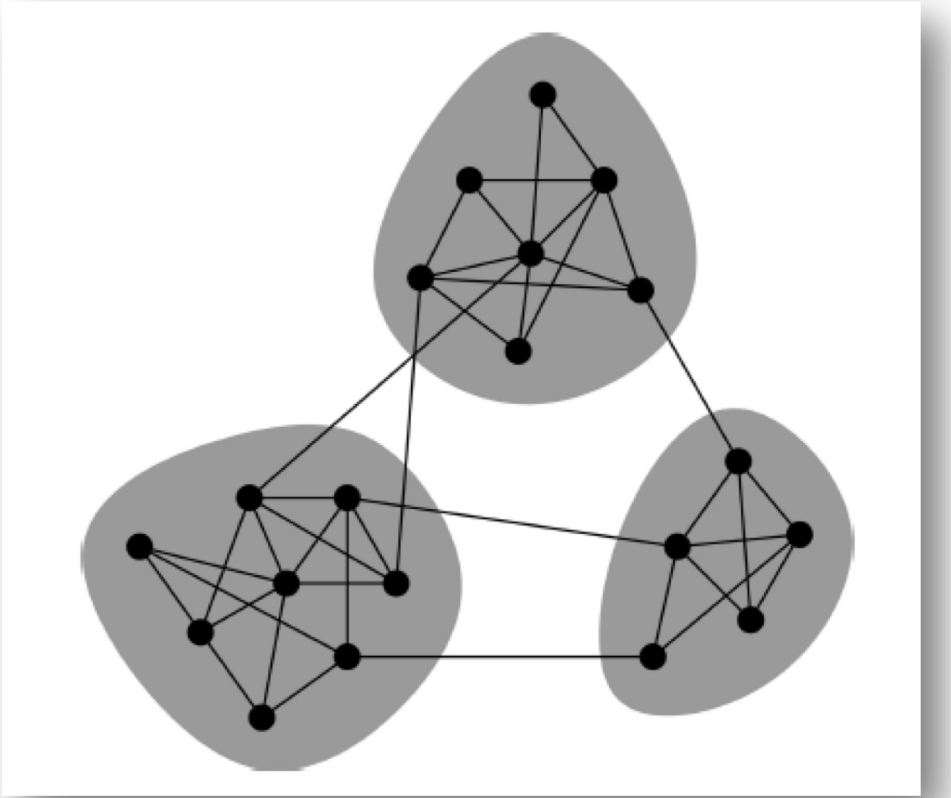
- *A module is a part of solution (i.e., something that may be clearly delineated from the solution), such that it exhibits some form of independence (full of partial) from the remaining part of solution (often referred to as context)[2].*
- Example, operating system, etc.



Operating system

Modules in Networks

- *Networks are modular if they contain highly connected clusters of nodes that are sparsely connected to nodes in other clusters[3].*



Modular Network [4]

Advantages of Modularity - Biology

- Modularity can enhance evolvability, an organism's capacity to generate adaptive heritable variation, for two reasons:
  - the organization of biological systems into modules may permit changes inside one module without perturbing other modules.
  - modules can be combined and reused to create new biological functions
- Under certain conditions, there is a direct correlation between modularity and fitness[3]

Advantages of Modularity - Software

- Modular software is easier to read
- Can make changes in one module without affecting other modules
- Can reuse modules. Example, library functions in various programming languages, etc.
- GP: intersection between evolutionary biology and software. We want evolvable and more readable programs

Modularity in GP Systems

- Automatically defined Functions (ADFs) by John R. Koza
  - Evolving programs can use ADFs as modules
  - Numbers and types of ADFs, and other restrictions specified in advance by the user
- Architecture-altering Operations by John R. Koza
  - Enables the programs to alter the number and types of ADFs during the process of evolution
  - More sophisticated and complicated genetic operators needed

Modularity in GP Systems

- Tag-based modules in PushGP
  - A set of instructions can be tagged with a number and later retrieved using the same number
  - Thought to be useful for the evolution of modular programs
  - Later found to be useful only for a small number of problems

Modularity in GP Systems

- Evolution hates modularity (in the short run)!
- Modular programs do not provide smooth fitness gradient as is provided by non-modular ones
  - In non-modular programs, every change (in the right direction), whether small or big, provides an improvement in the overall fitness, but in case of modular programs, until you get the whole module right, you might not notice any change in the fitness.

Defining Modularity for Evolved Programs

- Requirements to be considered a module?
  - Elements comprising the module should be together
  - There is a specific beginning and an end to the module
  - While using the module at different locations, the same copy should be used

Defining Modularity for Evolved Programs

Modularity composed of two parts: Reuse (U) and Repetition (P)

$$U = \frac{Execution\ steps\ coming\ from\ using\ modules}{Total\ execution\ steps}$$

$$P = \frac{Execution\ steps\ coming\ from\ using\ duplicates\ of\ modules}{Total\ execution\ steps}$$

Open research problems

- How to get evolution to use modules?
  - Additional selection pressure: in addition to fitness, use modularity value to select parents
- Evolutionary origins of modularity?
- Is modularity an essential ingredient to solve more complex problems?
- Why does evolution hate modularity?

References

1. Espinosa-Soto, Carlos, and Andreas Wagner. "Specialization can drive the evolution of modularity." *PLoS computational biology* 6.3 (2010): e1000719.
2. Krawiec, Krzysztof, and Bartosz Wieloch. "Functional modularity for genetic programming." *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009.
3. Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. 2013. The evolutionary origins of modularity. In Proc. R. Soc. B, Vol. 280. The Royal Society, 20122863
4. Newman, Mark EJ. "Modularity and community structure in networks." *Proceedings of the national academy of sciences*103.23 (2006): 8577-8582.
5. Poli, Riccardo, et al. *A field guide to genetic programming*. Lulu. com, 2008.

Thanks

aks@cs.umass.edu