Conception and Computation

2 Cases on the Expression and the Execution of Thought[1]

Remy Wang

December 15 2017

Usually, a researcher in computational biology implements her idea in a general purpose language like Python, which compiles down to machine code to be executed on commercial CPUs. This paper explores alternatives at both ends of this process: 1. I present the design of a visual programming interface that generates fast and maintainable programs, which allows the scientist to express her idea with ease 2. I present a reduction of the sequence alignment problem to shortest path with constraints, which is solved by ant colonies and slime mold with parsimonious resource, or by supermassive black holes at the speed of light.

[1] I received generous help from Professor Luis Ceze and Max Willsey for the first part of this project. Chandrakana Nandi helped me print out the matrix for the second part.

Conception

What Do You Do With Raw Data?

:) is a scientist at a computational biology lab. She splits her time between the wet lab and her office - in the wet lab, she turns rat tails into DNA sequences, or rather, FASTA files containing sequence reads. On her computer, she writes programs that analyze the reads and report meaningful statistics. One day :) was handed the source code of an analysis pipeline that contained the following snippet:

```
cmd = "python " + script_dir + "/combine_rs.py
    \ -rf " + stats_name + " -fastqs "
    \ + fastq_name_str + ' -combinetruncation'
execute(cmd)
```

In short, the code in python passes a string to be executed on the command line, and the command calls another python program that does something similar. It took :) and I hours to extract the diagram in Figure 1 from the source code - obviously, it is unmaintainable. In addition, it is slow. It misses the many opportunities of parallelism revealed in the diagram.

`Make` *makes it better*

:) and I determine to refactor the software to make it fast and maintainable. We soon realize that a similar problem is encountered by system programmers, namely to build software that involves the interaction of many source files and the outputs they produce. The

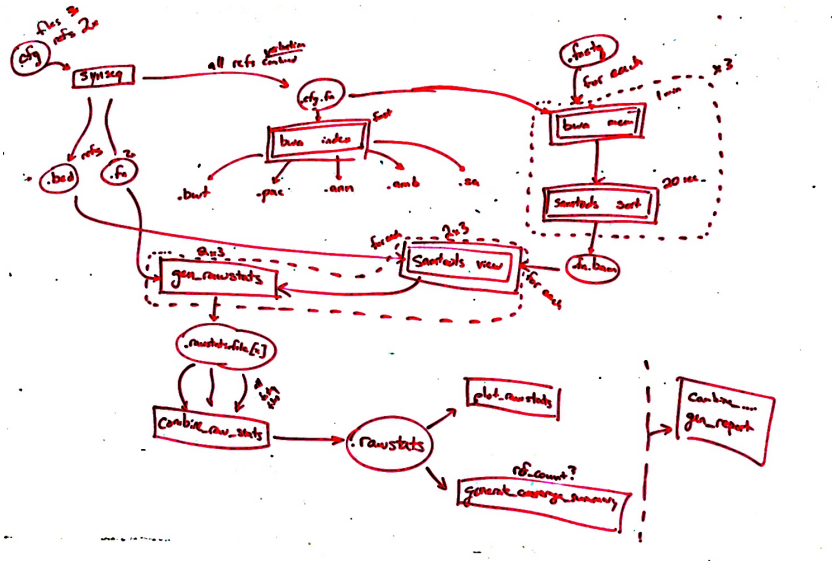

Figure 1: Workflow diagram extracted by hand

good news is, system programmers already have a working solution - `Makefiles` [2] [3].

A `Makefile` specifies the procedure to build a large system from its source files. Each `Makefile` consists of a set of *rules*, and each rule has 3 parts: the *source file*, the *compilation script*, and the *expected output* that the script should produce given the source file.

Although `Makefiles` are originally designed for build systems, it does not limit the script to only call compilers. It did not take long for us to repurpose `Make` for :)'s analysis software. The resulting `Makefile` collects all scripts and input/output files in one place, immediately making the system more manageable. In addition, the software runs faster even though we did not change any algorithm or data structure! The performance comes from two of `Make`'s optimizations: 1. if two rules do not depend on each other, `Make` executes them in parallel; 2. if a rule's input file is not modified, `Make` does not re-run that rule.

Draw it on the whiteboard, we will take it from there

:) and I were excited about the new implementation and shared it with the original developer, :(. To our surprise, :(did not want to use it. He said that Python is the only language he and his colleagues use, and they do not have time to learn `Make`. However, he really liked the diagram `Make` generated to represent the workflow (Figure 2), and asked if we can generate a similar one from Python.

We can do something even better. It turns out no two different

[2] Richard M. Stallman, Roland McGrath, and Paul D. Smith. *GNU Make: A Program for Directing Recompilation, for Version 3.81*. Free Software Foundation, 2004. ISBN 1882114833

[3] Johannes KÃűster and Sven Rahmann. SnakemakeâĂŤa scalable bioinformatics workflow engine. *Bioinformatics*, 28(19): 2520–2522, 2012. DOI: 10.1093/bioinformatics/bts480. URL `+http://dx.doi.org/10.1093/bioinformatics/bts480`

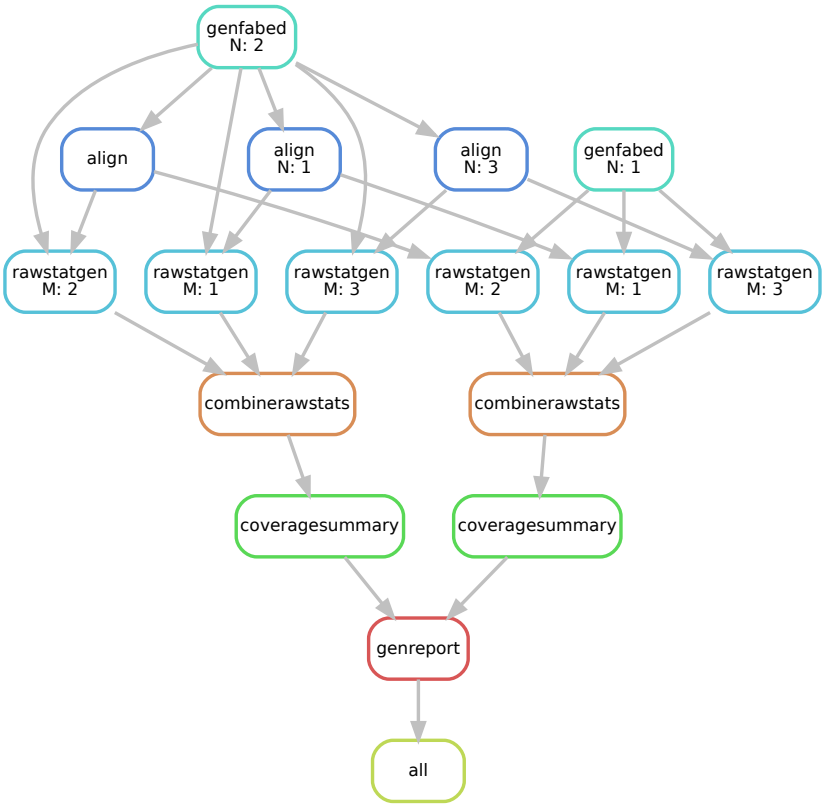

Figure 2: Workflow diagram generated by `snakemake -dag`

`Makefiles` generate the same diagram, i.e. there is a one-to-one correspondence between a `Makefile` and a diagram. Since `Make` already implements the function from a `Makefile` to a diagram, the inverse of that function will produce a `Makefile` from a diagram. With that in mind, we designed a visual programming interface, Pictorial Representation. PR allows :(to directly sketch out the workflow diagram and generates the corresponding `Makefile`.

Pictorial Representation Semantics

Each *diagram* in Pictorial Representation is a directed-acyclic *graph*, as shown in Figure 3. Each *node* in the graph contains a *command* to run at that step. In the command, the `type-face` part specifies what script / program is used, whereas the **bold faced** part specifies the input to that command. The input can also be a regular expression that represents a set of files. For each node, the incoming arrow signifies which other nodes the current node depends on, i.e. where the input files are generated from. A node without any incoming arrow signifies the start of the pipeline.

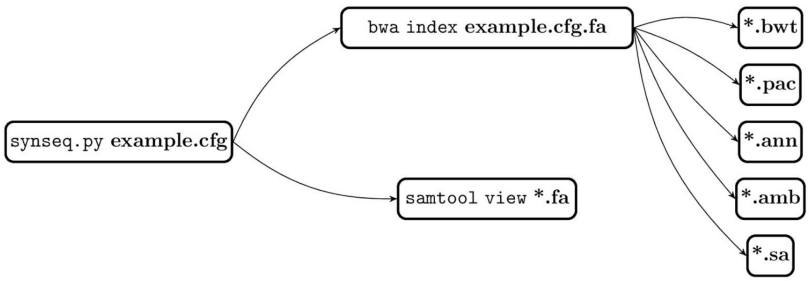

Figure 3: A fragment of the diagram for :)'s analysis pipeline

Summary

At high level, Pictorial Representation provides an expressive yet user friendly interface for scientists to specify complex workflows visually. At low level, PR generates fast yet maintainable code in `Make` that automatically incrementalizes and parallelizes analysis tasks.

Computation

:D is an international student from W. She just took a course in Computational Biology this term, and was fascinated by the Needleman-Wunsch [4] algorithm used to align sequences. She is excited to explain the algorithm to his little brother XD back home, but there is one problem - computers are expensive in W, and :D could not find one

[4] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3): 443 – 453, 1970. ISSN 0022-2836. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. URL `http://www.sciencedirect.com/science/article/pii/0022283670900574`

where they lived. Nevertheless, she explained the algorithm on paper to XD, hoping he would understand it without having to run a program. XD's very smart anyways.

Sequence alignment is finding the shortest path

:D : ":D :D :D :D"

XD : "XD XD XD XD"

:D : "Let's look at this matrix!"

XD : "XD"

:D : "On the top, we can see one sequence of DNA, and on the left side we see another one."

XD : "XD"

:D : "Starting from the top left corner, we want to go either right, down, or diagonally to eventually arrive at the lower right corner."

XD : "XD"

:D : "But we go right or down, we won't get as high a score as we go diagonally, so we want to go diagonally as much as possible."

XD : "So why can't we always go diagonally?"

:D : "Great question! Because we can only go diagonally when the corresponding letters from the sequences match."

XD : "OK I'm ready, let's fill this out!"

:D : ":D"

...

XD : "Wow, doesn't this look like some shortest path between the corners, with obstacles?"

:D : ":O"

XD : "Yes, it is! In class yesterday Mr. 8) just told us the Hypotenuse of a right angled triangle is shorter than the sum of the sides."

:D ":O"

XD : "So if we assign a higher score to a shorter step, then a diagonal step would be better than a horizontal step and a vertical one."

:D : "!!!"

XD : "Mr. 8) also told us ants can solve the shortest path problem [5] [6], so we can run this with ants!"

:D : "XD"

Ant-tomaton

So XD and :D cut up a piece of cardboard, and wherever there is a mismatch between the sequences, they put a piece of obstacle in the matrix. Within 5 minutes, they found a small ant mound outside their house. They lay out the matrix so that the ants can enter from the top left corner, and put a cookie on the lower right corner. (Figure

[5] Alexander Schrijver. On the history of the shortest path problem. *Doc. Math*, 155, 2012

[6] ACMDV Maniezzo. Distributed optimization by ant colonies. In *Toward a practice of autonomous systems: proceedings of the First European Conference on Artificial Life*, page 134. Mit Press, 1992

4) Of course it worked! Unfortunately, :D and XD could not record the experiment because they did not have a camera. Luckily, Mr. 8) has one, so XD brought the matrix to class the second day hoping Mr. 8) would help him record it.

Figure 4: Alignment matrix to be solved by ant colonies

Physarum Polycephalum, or the many-headed slime

XD did not know Mr. 8) has a big surprise for his class: he brought in an alien-looking creature called Physarum Polycephalum (slime mold), which can also find the shortest path [7]! Overjoyed, XD asked Mr. 8) to align the sequences with the slime mold.

Mr. 8) and XD took the pictures below (Figure 5)before the principle threatened them that if they don't throw out the slime mold, they both have to leave the school. We can see that the mold made it all the way to the 4th last match, before getting blocked by the obstacles shaken up by the principle.

They called it a success! Ever since, DNA alignment with ants and slime mold have become a lecture in Mr. 8)'s Computer-less Science

[7] Atsushi Tero, Ryo Kobayashi, and Toshiyuki Nakagaki. A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of theoretical biology*, 244(4): 553–564, 2007

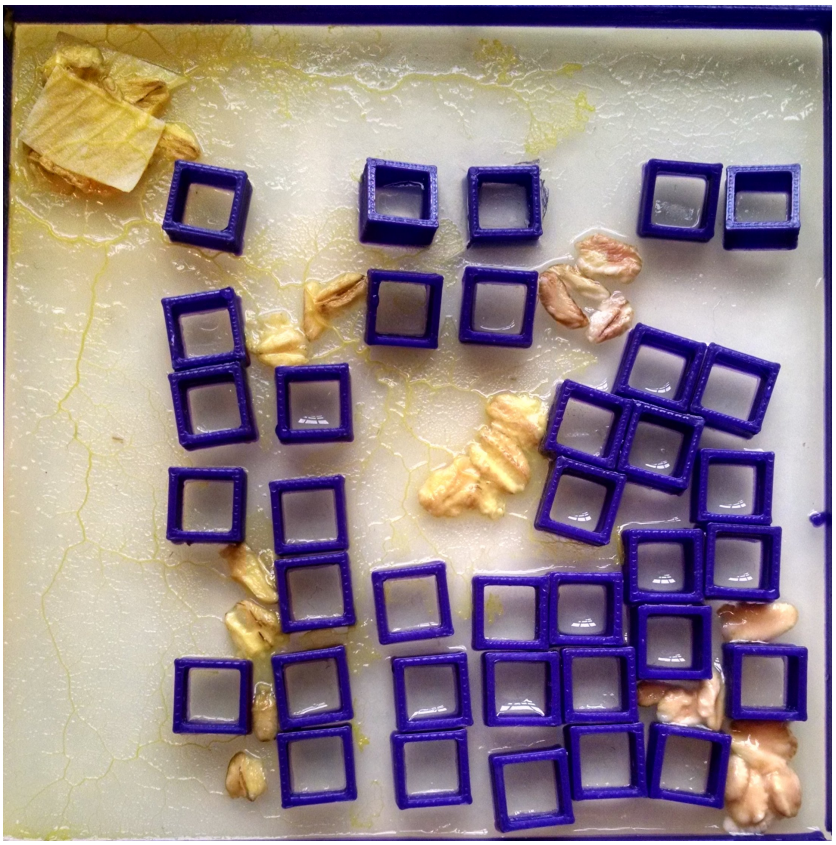

Figure 5: A fragment of the diagram for :)'s analysis pipeline

curriculum, and has been repeated by many other schools in W.

Supermassive black holes and grad students

There is still one part of the experiment XD and Mr. 8) are not satisfied with - they take too long! It can take several hours for the ants to find the optimal path, and up to a week for the slime mold. Fortunately, the fastest possible thing in nature - light, also follows the shortest path [8]. The difficulty is laying out the obstacles without completely blocking light. The best obstacle XD and Mr. 8) can think of, is a supermassive black hole. Since black holes distort the spacetime around them[9], XD and Mr. 8) can simply place a black hole everywhere there is a mismatch in the matrix. The drawback with that, of course, is that the placement of black holes would consume a lot of energy, which the school could not afford. Therefore XD and Mr. 8) asked for help from the local university, the University of W or UW in short. But it turns out they did not need to manipulate black holes at all - all they had to do was to place some pizzas at one corner of a conference room and lay out the obstacles with chairs, and the grad students found the shortest path to the pizzas within minutes (Figure 6)!

[8] Graham M Shore. Quantum gravitational optics. *Contemporary Physics*, 44 (6):503–521, 2003

[9] Ted Jacobson. Thermodynamics of spacetime: the einstein equation of state. *Physical Review Letters*, 75(7):1260, 1995

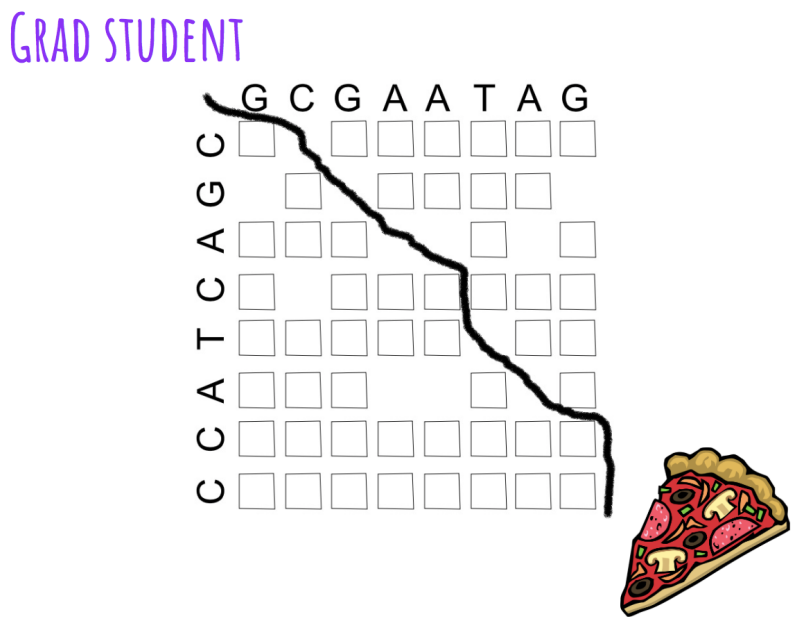

Figure 6: A fragment of the diagram for :)'s analysis pipeline

Future of thinking & doing

Through their own journeys, the pair of trios, :), :(and I, and :D, XD and 8) learned some important lessons about the expression and execution of thought.

Expression of thought

Although `Make` is less powerful than Python - in fact, it is not even Turing-complete, it still did a better job both in terms of performance and maintainability. Languages like `Make` are called Domain Specific Languages (DSLs) [10] in the programming languages research community. DSLs follow the philosophy of Less is More: by limiting the programs they can express, they express them more clearly and efficiently. Other DSLs for bioinformatics include [11] and [12]. We chose `Make` because it is well supported by its developers and has a large user base.

Furthermore, we need not limit ourselves to express our thoughts through text. There has long been visual programming languages and interfaces [13], with the oldest ones dating back to the 60's [14]. However, those languages and interfaces have not been widely adopted, because diagrams can be ambiguous depending on how one interprets them. By providing a denotational semantics [15], I make Pictorial Representation precise, so that the programmer can easily produce rigorous software.

Execution of thought

Since personal computers, professionals inside and outside the programming world have been spending more and more time in front of their computers. Even at home or with friends, some are glued to their different sized screens.

:D, XD and Mr. 8) have taught us that computation can happen anywhere in nature, even in the most boring looking ant hills and slime mold. I call on my fellow scientists to pay attention to the various computation already happening around them and empower themselves with the help of nature, instead of trying to reduce everything to bits and circuits. For the case of :D, XD and Mr. 8), that allowed them to teach computational biology in their developing home town with limited resources.

In the computer architecture research community, the method of adopting non-traditional physical processes for computation is called acceleration [16]. In fact, a quantum processor is an accelerator [17]. Slime mold and ants as accelerators consume extremely little energy (oats and cookies), and cost little to construct (cardboard boxes).

[10] Peter J Landin. The next 700 programming languages. *Communications of the ACM*, 9(3):157–166, 1966

[11] Kleberson J do A Serique, José L Santos, Dilvan de Abreu Moreira, et al. Biodsl: a domain-specific language for mapping and dissemination of biodiversity data in the lod. In *Congresso da Sociedade Brasileira de Computação, XXXVI; Brazilian e-Science Workshop, X*. Faculdade de Informática-FACIN, 2016

[12] Stuart Anthony Byma, Sam David Whitlock, Laura Flueratoru, Ethan Tseng, Christos Kozyrakis, Edouard Bugnion, and James Larus. Persona: A high-performance bioinformatics framework. In *USENIX Annual Technical Conference 2017*, number EPFL-CONF-229429, 2017

[13] Daniel D Hils. Visual languages and computing survey: Data flow visual programming languages. *Journal of Visual Languages & Computing*, 3(1): 69–101, 1992

[14] TO Ellis, John F Heafner, and WL Sibley. The grail project: An experiment in man-machine communications. Technical report, RAND CORP SANTA MONICA CA, 1969

[15] David A Schmidt and Denotational Semantics. A methodology for language development. 1997

[16] Adrian J Van De Goor and AJ Van De Goor. *Computer architecture and design*. Addison-Wesley Reading, Mass., 1989

[17] Jozef Gruska. *Quantum computing*, volume 2005. McGraw-Hill London, 1999

Declaration

I long for a future where science will not be reduced to bits and circuits, and where scientists are reconnected to nature and rediscover the joy of science.

References

Stuart Anthony Byma, Sam David Whitlock, Laura Flueratoru, Ethan Tseng, Christos Kozyrakis, Edouard Bugnion, and James Larus. Persona: A high-performance bioinformatics framework. In *USENIX Annual Technical Conference 2017*, number EPFL-CONF-229429, 2017.

TO Ellis, John F Heafner, and WL Sibley. The grail project: An experiment in man-machine communications. Technical report, RAND CORP SANTA MONICA CA, 1969.

Jozef Gruska. *Quantum computing*, volume 2005. McGraw-Hill London, 1999.

Daniel D Hils. Visual languages and computing survey: Data flow visual programming languages. *Journal of Visual Languages & Computing*, 3(1):69–101, 1992.

Ted Jacobson. Thermodynamics of spacetime: the einstein equation of state. *Physical Review Letters*, 75(7):1260, 1995.

Johannes KÃűster and Sven Rahmann. SnakemakeâĂŤa scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012. DOI: 10.1093/bioinformatics/bts480. URL `+http://dx.doi.org/10.1093/bioinformatics/bts480`.

Peter J Landin. The next 700 programming languages. *Communications of the ACM*, 9(3):157–166, 1966.

ACMDV Maniezzo. Distributed optimization by ant colonies. In *Toward a practice of autonomous systems: proceedings of the First European Conference on Artificial Life*, page 134. Mit Press, 1992.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970. ISSN 0022-2836. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. URL `http://www.sciencedirect.com/science/article/pii/0022283670900574`.

David A Schmidt and Denotational Semantics. A methodology for language development. 1997.

Alexander Schrijver. On the history of the shortest path problem. *Doc. Math*, 155, 2012.

Kleberson J do A Serique, José L Santos, Dilvan de Abreu Moreira, et al. Biodsl: a domain-specific language for mapping and dissemination of biodiversity data in the lod. In *Congresso da Sociedade Brasileira de Computação, XXXVI; Brazilian e-Science Workshop, X*. Faculdade de Informática-FACIN, 2016.

Graham M Shore. Quantum gravitational optics. *Contemporary Physics*, 44(6):503–521, 2003.

Richard M. Stallman, Roland McGrath, and Paul D. Smith. *GNU Make: A Program for Directing Recompilation, for Version 3.81*. Free Software Foundation, 2004. ISBN 1882114833.

Atsushi Tero, Ryo Kobayashi, and Toshiyuki Nakagaki. A mathematical model for adaptive transport network in path finding by true slime mold. *Journal of theoretical biology*, 244(4):553–564, 2007.

Adrian J Van De Goor and AJ Van De Goor. *Computer architecture and design*. Addison-Wesley Reading, Mass., 1989.