Must:

- Deal with abstraction & reasoning
- Deal with the link between abstraction/reasoning and perception
    - Start from perceptual data, from “experience”
- Be crazy ambitious, almost impossible (but with ability to get started in a simple way)
- Be open-ended or extensible (high potential complexity and high diversity)
    - AI game designers vs. game players?
- Have clear, accessible metrics
- Which would measure **generalization power**
- Have some real-world applicability or side-effect that provides value in the real world
- **Have clear alignment between useful real-world applications and success on benchmark metrics**

Possible topics:

- Theorem proving
    - Pros: abstraction / reasoning; extensible; open-ended
    - Cons: not applied enough; may not generalize to other tasks
- Program synthesis from input/output pairs
    - But what kind? Need a problem domain
- Program synthesis from some high-level under-specified description (language, figures)
- Game environments
- AI game designers vs. game players
- …

Applications:

- Knowledge engines, open Q&A
- Assistants, “do engines”
- Programming assistants
- Music / art creation assistants

Inspiration

Programming language -> env programs -> (interpreter) -> observable environment -> agent -> behavior programs -> (interpreter) -> new situations

Try to maximize {agency|observations}, i.e. try to maximally generalize (with respect to affordances) from as little data as possible.
Goal: not best behavior policy, but rather maximal {I = Generalization / Experience}
Which is basically a proxy for {Power / Experience} (or {Power / Time}) which is proxy for satisfying evolutionary constraints.

Benchmark:

Orientation: building human assistants or virtual interns

Core idea: showing few examples of a high-level task, and getting the AI to generalize to new entries

Any domain is fine

Simple example: a replacement for regex

Regex golf

See also: excel formulas

Source: a set of programs (not randomly generated, must match useful programs)
Taks: program gives you a few positive examples (and negative?) examples (input/output pairs?), must predict the next output given an input

- Variant: you have an oracle and must guess the answer in the fewest possible trials (harder to turn into a competition)

Goal: generalize to new programs

- Have a way to measure program complexity

Regex golf
Input/output pairs for string transformation programs (e.g. extraction / formatting regexes)

- There should not be any obvious structure in program space
- Webpage information extraction

Navigation tasks on wikipedia (knowledge querying)
Web navigation tasks (do-engine style)

- What is the most useful thing to automate on the web?

Text-based games
Combination of vision and text

“Learning to execute a high-level query language against a world of information (e.g. the internet), given a few examples”

Tools to solve:

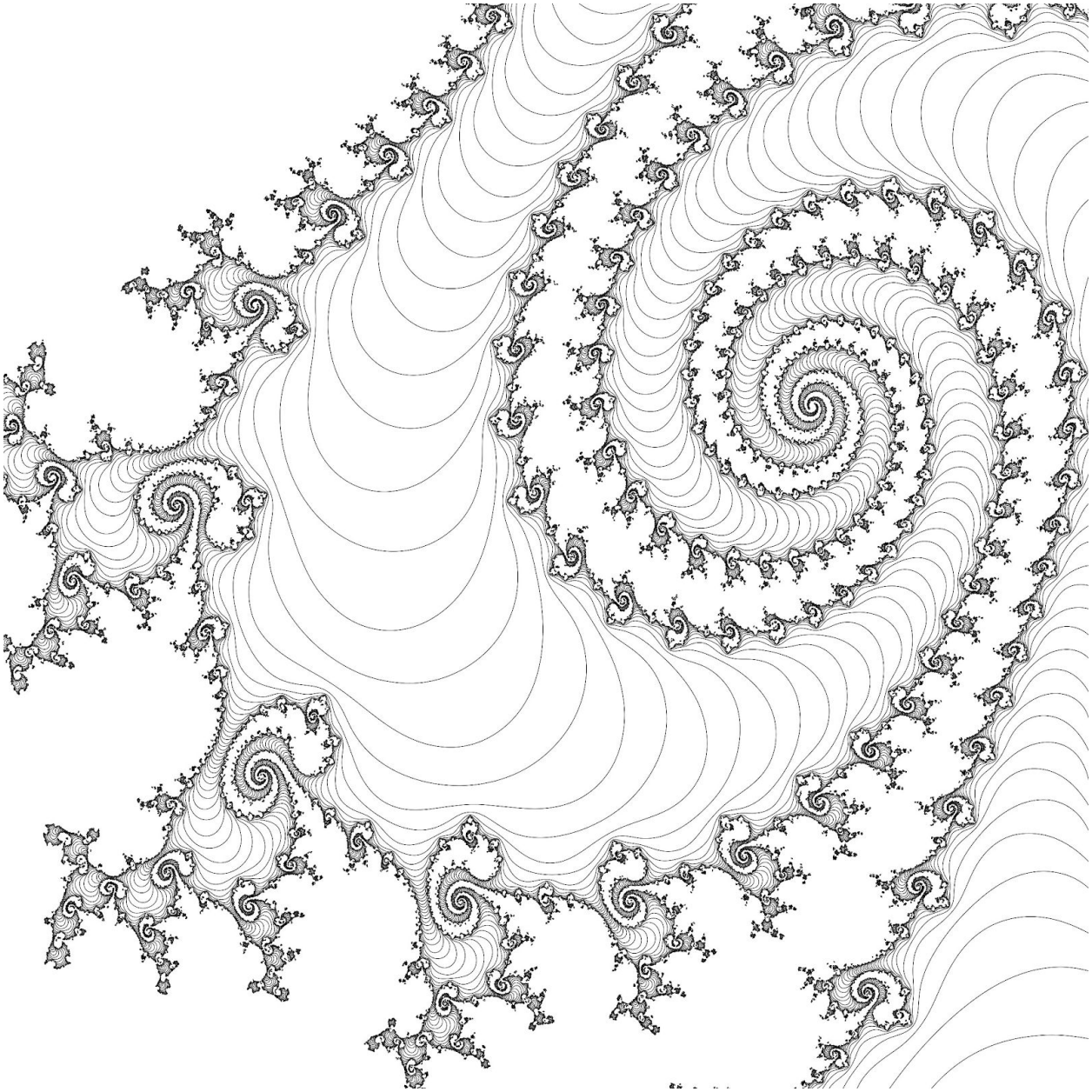
- Abstraction
- Analogies
- Robust modeling
- Strong extrapolation along multiple axes

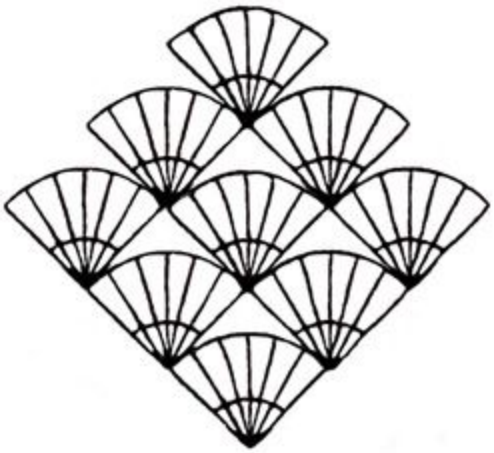
Ideal source of data: reservoir of interesting patterns (rich enough that cannot be hacked by hardcoding priors) that can’t be hacked with human semantic priors (e.g. wikipedia/etc is

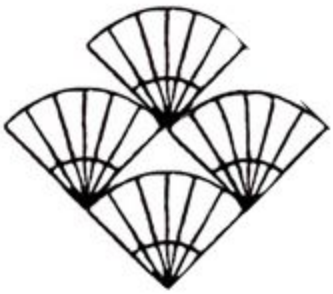
problematic), meaning external data would typically not help. Possibilities: code, math, physics, biology, nature. Genome?

- Program A generates data grid
    - Pixel grid? Dna? Bytecode? Ascii/utf8? Prime numbers?
    - Fractals have interesting recursive structures
    - Snowflake designs, sashiko designs
- Program B extracts information from the grid, producing a few example
    - Maps with patterns of interests to the human mind
    - Should be intuitively doable by humans
    - Sufficient depth of reasoning to be interesting (in particular allow for recursion)
    - Example programs:
        - Recognize “classes” of things in input
        - Correct anomaly in emergent pattern
        - Inverse of detecting anomalies: recognizing invariants
        - Complete series (doable in one shot)
        - Classify pattern given a couple positive/negative examples
            - == Bongard problem
        - Extract count of pattern / character
        - “How many triangles in this image” type problems
        - Segmenting grid into sections
- Must generate B from a small number of input/output pairs
- Yet, strong mapping to useful tasks

Something that’s intuitive for humans for near-impossible for machines







?