SYSTEMS ENGINEERING · WEEK 13 · STATION V

Agents as Processes

What a language model is, in this class's terms. Sandboxing is virtualization. The loops are closed. The gate is what you built.

WHAT IT IS

A function from text to a distribution over the next token, with 10^{11} parameters, run in a loop.

Trained on most of the written record to predict the next piece of text; sampled one token at a time and fed its own output back. Nothing in it is a semantics: it does not run programs, check proofs, or consult the world. It produces notation that is likely, in the sense of resembling what was written before.

That makes it superb at everything that is a property of notation: fluency, style, translation, recall of patterns. And structurally unable to certify anything that is a property of meaning. A hallucination is a proof-shaped object that is not true; the machine cannot tell, because telling would require the semantics it does not have.

IN SYSTEMS TERMS

A program, running on someone's hardware, consuming a great deal of energy, whose output is other programs, commands and requests. Untrusted, by construction: not malicious, but unverified, and there is a whole class about what a kernel does with those.

SANDBOXING IS VIRTUALIZATION

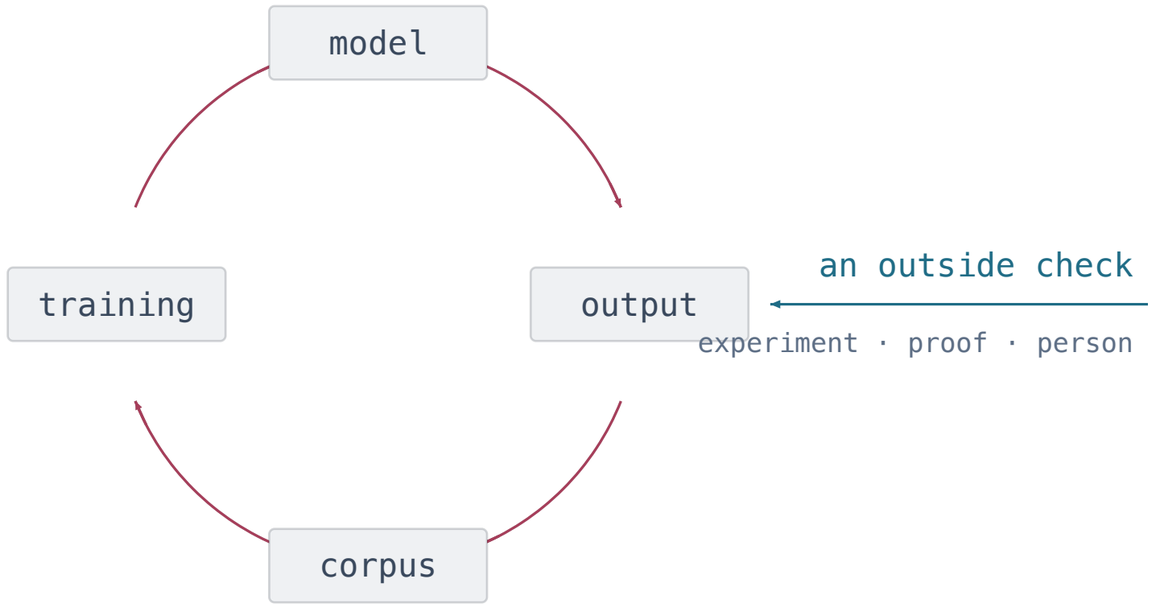
An agent runs generated code. Everything you built is what runs it safely.

THE AGENT NEEDS	THE KERNEL PROVIDES	WEEK
run code it did not verify	a process, isolated in space by paging	4, 7
code that may not stop	a timer; a bound in instructions, seconds, or tokens	5
a limited view of the world	the door: system calls, and the kernel says no	2, 7
many attempts, in parallel	fork; cheap contexts; copy-on-write	7
a budget	the profile: instructions, memory, joules	11
trust in the sandbox itself	a small, named TCB; a kernel beneath the kernel	6, 12

The sandbox around an agent is a virtual machine, and its designer faces the self-reference of week 6: the agent may try to reach the sandbox's controls, and the isolation of the controls is the whole question. Nothing new; harder stakes.

THE LOOPS ARE CLOSED

Models train on model output. Models grade models. Agents invoke themselves. And no system can certify itself.



model, output, corpus, training, and round again; the arrow from outside is the check

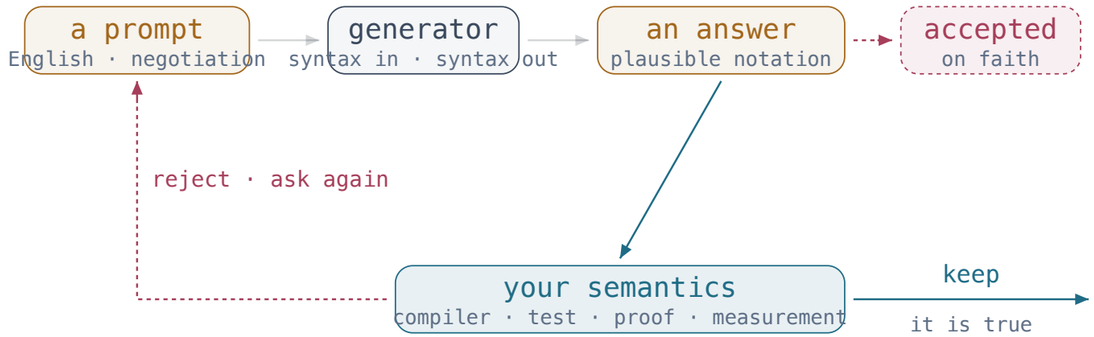
Self-reference, the guiding principle, in the machines: model collapse when the loop tightens, judge and candidate sharing blind spots, agents that call agents. The figure is week 6's ladder without a bottom.

Is this system safe? A semantic property; Rice says no general procedure. Can it explain itself? By Gödel II, an explanation is more output, not a certificate. Introspection is not audit. The kernel knew this: it never asks the process; it isolates and bounds and checks from outside.

So the effort goes where the arrow enters from outside: external, independent, bounded checks. Which is the Cost chapter, and the gate.

THE GATE · IN THIS CLASS

Ask a machine for a systems routine. Then mipster, your kernel, rotor.



the only part the machine does not supply

a generator supplies notation; the gate supplies the semantics

```
$ ./selfie -c generated.c -m 1
./selfie: … exception: invalid memory access at 0x…
$ ./selfie -c selfie.c -m 2 -c generated.c -y 1
$ ./rotor -c generated.c - 0 && tools/bitme.py -kmax 1000
… core-0-load-seg-fault — satisfiable · input …
```

Does it run? On mipster, isolated: the exception handler catches what the generator did not. Does it run under your kernel? Under hypster, time-shared with others, with your locks. Can it fail? Within a thousand steps, on any input: absence within a bound.

None of the three is available to the machine that wrote it. Cantor's object from outside the list, Gödel's stronger system, Thompson's second compiler, the check the generator cannot be: the fourth appearance of one theorem, and this class built all three checks.

WHAT CHANGED

Generation got cheap. Isolation, bounding and verification did not.

Before, producing a program was expensive and scarcity did the filtering. Now production is nearly free and unbounded, and the filter has to be built: a kernel to run it in, a bound to stop it, a check to accept it. Hard to find, easy to check, at consumer scale.

Value migrates to the ends the machine does not occupy: stating what should be true, and establishing that it is. For a systems engineer the second is the job description. Own the ends.

BEFORE NEXT WEEK

The Machines chapter's exercises 1, 2 and 6, with a systems routine instead of factorial. Bring the failing input rotor found, or the bound up to which none exists. Next week: what a system is.