Measuring the intelligence of an idealized mechanical knowing agent

Samuel Alexander
The U.S. Securities and Exchange Commission
alexander@math.ohio-state.edu

For the full paper behind these slides, see:
https://philpapers.org/archive/ALEMTI-2.pdf

Measuring intelligence by what is known

- By a “Strong AI”, or “mechanical knowing agent”, I envision an agent like *HAL* from *2001: A Space Odyssey*: A mechanical agent who can reason like a human.

- No-one knows the right way to formally define a Strong AI.

- Whatever a Strong AI is, such an agent presumably *knows* certain mathematical facts.
  - Can we formally define an intelligence measure for Strong AI’s based on the mathematical facts they know?

A Guiding Question

- Question: Suppose $A_1, A_2, A_3, \ldots$ are Strong AIs, each *significantly more intelligent* than the previous. If B is another Strong AI, does it necessarily follow that there exists some k such that A_k is more intelligent than B?

A Guiding Question

- Question: Suppose $A_1, A_2, A_3, \ldots$ are Strong AIs, each *significantly more intelligent* than the previous. If B is another Strong AI, does it necessarily follow that there exists some k such that A_k is more intelligent than B?

- If intelligence is measured using real numbers, and "significantly more intelligent" implies "at least $+1$ more intelligent", then the answer is automatically *YES*. Not because of anything to do with intelligence, but just because of the structure of the real numbers.

A Guiding Question

- Question: Suppose $A_1, A_2, A_3, \ldots$ are Strong AIs, each *significantly more intelligent* than the previous. If B is another Strong AI, does it necessarily follow that there exists some k such that A_k is more intelligent than B?

- If intelligence is measured using real numbers, and "significantly more intelligent" implies "at least $+1$ more intelligent", then the answer is automatically *YES*. Not because of anything to do with intelligence, but just because of the structure of the real numbers.

- Scandal! The above answer is too easy.
- This leads us to investigate alternatives to the real numbers.

An Intuitive Ordinal Notation System

Definition: (See examples on the following slides) We define the *ordinal notations* to be the smallest set of computer programs such that:

- For every computer program P, if, when P is run, every output of P is an ordinal notation, then P is an ordinal notation.

For any ordinal notation P, let $|P|$ be the smallest ordinal α such that $\alpha > |Q|$ for every ordinal notation Q which P outputs. We say that P *notates* the ordinal $|P|$.

Example: 0

Let P_0 be the program "End", which immediately ends, outputting nothing.

Vacuously, every output of P_0 is an ordinal notation. Therefore, P_0 is an ordinal notation. It notates the ordinal $|P_0| = 0$.

Example: 1

Let P_1 be the program “Print(‘End’)”, which outputs “End” and then stops.

P_1 outputs nothing except ordinal notations (namely P_0), therefore P_1 is an ordinal notation. It notates $|P_1| = 1$.

Example: 2

Let P_2 be the program “Print(‘Print(‘End’)’)”

P_2 outputs “Print(‘End’)” [i.e., it outputs P_1], and then stops.

Since P_2 outputs nothing except ordinal notations, P_2 is an ordinal notation. It notates $|P_2| = 2$.

Detour: Components of intelligence

Whatever intelligence is, it surely involves core components like:

- pattern-matching
- creativity
- the ability to generalize
- the ability to abstract

Ordinal Notation Examples: $0, 1, 2$

- Ordinal notation for 0: “End”
- Ordinal notation for 1: “Print(‘End’)”
- Ordinal notation for 2: “Print(‘Print(‘End’)’)”

Ordinal Notation Examples: $0, 1, 2$

- Ordinal notation for 0: “End”
- Ordinal notation for 1: “Print(‘End’)”
- Ordinal notation for 2: “Print(‘Print(‘End’)’)”

Using your *intelligence* (your *pattern matching*, *creativity*, *ability to generalize*, and *ability to abstract*), can you figure out the pattern? Can you express the pattern in a single program?

Example: ω (the first infinite ordinal)

Let P_ω be the following program:

Let X = “End”; While(True) { Print(X); Let X = “Print(‘” + X + “’)” }

P_ω outputs “End”, “Print(‘End’)”, “Print(‘Print(‘End’)’)”, and so on forever.

P_ω outputs nothing except ordinal notations for $0, 1, 2, \ldots$, so P_ω is an ordinal notation. It notates ω, the smallest infinite ordinal.

Example: $\omega + 1$

Let $P_{\omega+1}$ be the program "Print(P_ω)" (where P_ω is the program from the previous example).

$P_{\omega+1}$ outputs nothing except the ordinal notation P_ω, so $P_{\omega+1}$ is an ordinal notation. It notates $\omega + 1$.

Example: $\omega + 2$

Let $P_{\omega+2}$ be the program "Print($P_{\omega+1}$)" (where $P_{\omega+1}$ is the program from the previous example).

$P_{\omega+2}$ outputs nothing except the ordinal notation $P_{\omega+1}$, so $P_{\omega+2}$ is an ordinal notation. It notates $\omega + 2$.

Examples: $\omega + 1, \omega + 2, \omega + 3$

- $P_{\omega+1}$= "Print(P_{ω})"
- $P_{\omega+2}$= "Print($P_{\omega+1}$)"
- $P_{\omega+3}$= "Print($P_{\omega+2}$)"

Using your *intelligence* (your *pattern matching*, *creativity*, *ability to generalize*, and *ability to abstract*), can you figure out the pattern? Can you express the pattern in a single program?

Example: $\omega \cdot 2$

Let $P_{\omega \cdot 2}$ be the following program:

Let X = P_{ω}; While(True) { Print(X); Let X = “Print(‘” + X + “’)” }

$P_{\omega \cdot 2}$ outputs nothing except P_{ω}, “Print(P_{ω})”, “Print(‘Print(P_{ω})’)”, etc., i.e., nothing except ordinal notations for $\omega, \omega + 1, \omega + 2, \ldots$

So $P_{\omega \cdot 2}$ is an ordinal notation for $\omega + \omega = \omega \cdot 2$.

Examples: ω, $\omega \cdot 2$, $\omega \cdot 3$

P_{ω}:

Let X = “End”; While(True) { Print(X); Let X = “Print(“” + X + “”)” }

$P_{\omega \cdot 2}$:

Let X = P_{ω}; While(True) { Print(X); Let X = “Print(“” + X + “”)” }

$P_{\omega \cdot 3}$:

Let X = $P_{\omega \cdot 2}$; While(True) { Print(X); Let X = “Print(“” + X + “”)” }

Using your *intelligence* (your *pattern matching*, *creativity*, *ability to generalize*, and *ability to abstract*), can you figure out the pattern? Can you express the pattern in a single program?

Let P_{ω^2} be the following program:

```
Let Left = ⌜ Let X = “ ⌝;
Let Right =⌜ ”; While(True) {Print(X); Let X = “Print(“” + X + “”)” } ⌝;
Let X = ‘End’;
While(True) {
    Let X = Left + X + Right;
    Print(X)
}
```

P_{ω^2} outputs nothing except notations for P_{ω}, $P_{\omega \cdot 2}$, $P_{\omega \cdot 3}$, ...

So P_{ω^2} is an ordinal notation for $\omega \cdot \omega = \omega^2$.

Exercise

1. Write ordinal notations for $\omega^3, \omega^4, \ldots$
2. Use your intelligence (pattern-matching, creativity, etc.) to find the pattern above and express it in a single program, to obtain an ordinal notation for ω^ω.
3. Write ordinal notations for ω^{ω^ω}, $\omega^{\omega^{\omega^\omega}}$, ...
4. Use your intelligence to find the pattern above and express it in a single program, to obtain an ordinal notation for $\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$.
5. Contemplate creative ways to go far beyond ε_0 .

Meta-Example: P_{Me}

- Imagine if instead of 25 minutes, I had infinity minutes, and I kept notating bigger and bigger ordinal numbers.

- Using your intelligence, could you figure out the pattern? Could you express the pattern in a single program?

Main Definition

Definition: For any truthful mechanical agent X, the *intelligence* of X is the smallest ordinal α such that $\alpha > |Q|$ for every computer program Q such that X knows that Q is an ordinal notation.

(By *truthful*, I mean an agent whose knowledge contains no falsehoods.)

Main Definition (Alternate Formulation)

If X is a truthful mechanical agent, we could give X the command:

> "Until further notice, do nothing except output as many ordinal notations as you can think of."

Let L be the resulting list of outputs. Since X is mechanical, there is some computer program P that outputs precisely L. If X is truthful, then P is an ordinal notation (because it outputs nothing except ordinal notations). The intelligence of X is defined to be $|P|$.

Detour: Application to Neuroscience?

Someday in the future, we could observe which neurons fire in the human brain as the human is tasked with writing ordinal notations for large ordinals.

It would be interesting to see the relationship between:

- the neurons which fire.
- the size of the ordinal which the subject manages to notate.

Philosophical Application: Singularity Skepticism

"Sons are seldom as good men as their fathers; they are generally worse, not better."–Homer (The Odyssey)

Theorem: If truthful mechanical agent X creates truthful mechanical agent Y, and if X knows the sourcecode of Y (because X created Y), and if X knows the truthfulness of Y (because why would X create an untrue child?), then X is more intelligent than Y.

Singularity Skepticism

Theorem: If truthful mechanical agent X creates truthful mechanical agent Y, and if X knows the sourcecode of Y (because X created Y), and if X knows the truthfulness of Y (because why would X create an untrue child?), then X is more intelligent than Y.

Proof: Let P be a computer program outputting exactly what Y would output if commanded, "Enumerate all the ordinal notations you can think of." Since X knows Y's sourcecode, X can infer P. Since X knows Y's truthfulness, X knows P is an ordinal notation. By definition, X's intelligence is the smallest ordinal α such that $\alpha > |Q|$ whenever X knows Q is an ordinal notation. So $\alpha > |P|$.

Objection!

Theorem: If truthful mechanical agent X creates truthful mechanical agent Y, and if X knows the sourcecode of Y (because X created Y), and if X knows the truthfulness of Y (because why would X create an untrue child?), then X is more intelligent than Y.

Q: What if Y is prohibitively expensive for X to simulate, doesn't the theorem break down in that case?

A: No, the theorem has nothing to do with X simulating or emulating Y. A person can reason about and manipulate sourcecode, without ever actually running that sourcecode. In the same way, X can reason about and manipulate Y's sourcecode even if it would be prohibitively expensive for X to actually execute that sourcecode.

Full paper

For the full paper behind these slides, see:
https://philpapers.org/archive/ALEMTI-2.pdf

Cite the paper:

```
@conference{alexander2019measuring,
  title={Measuring the intelligence of an idealized mechanical knowing agent},
  year={2019},
  author={Alexander, Samuel},
  booktitle={Cognition, Interdisciplinary Foundations, Models, and Applications (CIFMA))},
  publisher={Springer}
}
```