CONTENTS

ai expressions

An AGI API

Erick O. Oduniyi
Intelligent-Interfaces Group, USA

1 ENGINES, FUNCTIONS, AND OPERATIONS 2 APPLICATIONS

When we talk about ai expressions we must first talk about **engines**, **functions**, and **operations**.

1.1 Engines

1.1.1 Synthesis Engines

A process or sequence of functions that creates or adds information[1].

$$E_S : f_1 \rightarrow \ ... \ \rightarrow f_n \in \{S_n, S_1 \mid H(S_n) > H(S_1)\} \quad (1)$$

1.1.2 Analysis Engines

A process or sequence of functions that removes or subtracts information[2].

$$E_A : f_1 \rightarrow \ ... \ \rightarrow f_n \in \{S_n, S_1 \mid H(S_n) < H(S_1)\} \quad (2)$$

1.1.3 Lossless Transformations

A process or sequence of functions that don't add or remove any information.

$$T : f_1 \rightarrow \ ... \ \rightarrow f_n \in \{S_n, S_1 \mid H(S_n) = H(S_1)\} \quad (3)$$

1.2 Functions (flows)

Any mathematical function.

1.3 Operations

A collection of unary, binary, and/or ternary operations.

[1] Engines with function names like **generate, create, add, synthesize, build, compose, mutate** are strong candidates for being synthesis engines.

[2] Engines with function names like **remove, subtract, decimate, analysis, break, decompose, mutate** are strong candidates for being analysis engines.

Once you have defined engines, functions, and operations we can begin to compose larger computational structures. The most important composition of these fundamental concepts is an **application** — *a system that utilizes both synthesis and analysis engines*. Necessarily, these engines are composed of functions and operations within and over them.

2.0.1 A case in synthesis: leaf.engine

The **leaf.engine** created by Erick Oduniyi is an example of a synthesis engine: adds and formats files to **hyper-pages**, which uses a function called $generate()$ — *a mathematical function that takes as input a set of files and outputs those files bounded to a hyper-page*. As a reminder, hyper-page(s) are defined as:

$$\textit{hyper-page} = hy_{frame} +_b hy_{content} +_b \ hy_{border} \quad (4)$$

Where hy_{frame} (**hyper-frame**) and hy_{border} (**hyper-border**) are rectangles and the operation $+_b$ is called **superimposed addition** — *objects are placed behind or in-front of one another*. And, in between the frame and border is content, or **hyper-content**, $hy_{content}$

2.0.2 A case in analysis: image recognition, harmonic analysis, physical weathering

These cases (examples) are left to be imagined by the reader[3].

2.0.3 Application cases: echo.state, Instagram, Facebook

In the 21st Century, companies like **Instagram** and **Facebook** develop and maintain applications (the most prominent ones bearing the same name): adding, removing, sharing stories. The **echo.state**[4] system created by the **intelligent-interfaces Group** (ii^G) is also an example of an application: echo.state makes use of both synthesis and analysis engines. For example, the echo.state

[3] Hint: analysis engines remove information.

[4] Not to be confused with echo state networks

application uses a function called *mix() — a mathematical function that collects (transform), appends (synthesis), or removes (analysis) sets of hyper-pages* and/or **booklets**, *booklet*:

$$mix(Library) = \textit{hyper-page}_1 \ldots \{+, -, \cap, \ldots\} \ldots$$
$$\ldots \textit{hyper-page}_n \ldots \{+, -, \cap, \ldots\} \ldots$$
$$\ldots \text{booklet}_1 \ldots \{+, -, \cap, \ldots\} \ldots$$
$$\ldots \text{booklet}_n \tag{5}$$

Where $Library = \{booklet_1, \ldots, booklet_n\}$ (all of the publicly available digital booklets on echo.state). Furthermore, the mix function can perform any set or logical operation over hyper-pages and booklets, and thus is also an application. To be more precise, the mix function is an application embedded in the echo.state application! Moreover, echo.state also contains the library process, purchase process, and other engines to make it possible for digital booklets produced with the leaf.engine() to be interfaced with on both digital and physical platforms.

3 ENVIRONMENTS

Environments run applications. Unlike the previous computational structures, environments are not theoretical machines. We use ai expressions to describe them, but in reality they are resource dependent systems. We will start by giving a few real-world examples of environments: *Solar Systems, Planets, Continents (climates/environments).*

4 EXPRESSIONS

4.1 Verbal and non-verbal expressions

- Math
- Logic
- Program
- Language
- Body Movement

4.2 Shape Shifting

4.3 Mass Synthesis

4.4 Semantic Syncing

Not unlike the previous chapters and sections, this chapter is about a tool. However, here we discuss perhaps our lightest and most violent tool. Though, it is a tool you've come to know intimately (consciously or not). And while you may be familiar, there are aspects that we believe are often taken for granted (including by the authors). We want to highlight those aspects here. From there, we move forward faster than ever before.

Now, it is with great care, we discuss and deploy the backbone of AI (ai expressions), **semantic syncing**:

- Semantic — *relating to meaning in language or logic*
- Syncing — *the act of getting two elements into harmony*

4.5 Shape Shifting, Syncing, & Synthesis

In Japanese "Ai" [/ay/] means **love**:

$$\textit{Assume}\ \text{Ai} = \text{“love”} \tag{6}$$
$$\textit{Let}\ \text{Ai} = \text{ai} \tag{7}$$
$$\textit{Let}\ \text{ai} = \text{AI} \tag{8}$$
$$\tag{9}$$

The future of **artificial intelligence** (AI):

$$\textit{Assume}\ \text{AI} = \text{"artificial intelligence"} \tag{10}$$
$$\textit{Then}\ \text{"love"} = \text{“artificial intelligence”} \tag{11}$$
$$\textit{Then}\ \text{"love" expressions} = \text{"artificial intelligence" expressions} \tag{12}$$
$$\textit{Then}\ \text{ai expressions} = \text{AI expressions} \tag{13}$$

"Love languages" was already taken.