Chapter

Group-Assign: Type Theoretic Framework for Human AI Orchestration

Aik Beng Ng, Simon See, Zhangsheng Lai and Shaowei Lin

Abstract

In today's Information Age, we work under the constant drive to be more productive. Unsurprisingly, we progress towards being an AI-augmented workforce where we are augmented by AI assistants and collaborate with each other (and their AI assistants) at scale. In the context of humans, a human language suffices to describe and orchestrate our intents (and corresponding actions) with others. This, however, is clearly insufficient in the context of humans and machines. To achieve this, communication across a network of different humans and machines is crucial. With this objective, our research scope covers and presents a type theoretic framework and language built upon type theory (a branch of symbolic logic in mathematics), to enable the collaboration within a network of humans and AI assistants. While the idea of human-machine or human-computer collaboration is not new, to the best of our knowledge, we are one of the first to propose the use of type theory to orchestrate and describe human-machine collaboration. In our proposed work, we define a fundamental set of type theoretic rules and abstract functions *Group* and *Assign* to achieve the type theoretic description, composition and orchestration of *intents* and *implementations* for an AI-augmented workforce.

Keywords: AI Augmentation, Human Computation, Human AI Collaboration, Human AI Framework, Artificial Intelligence

1. Introduction

The nature of work is always transformed when automation is introduced. Looking back in history, automation has typically served to reduce or eliminate the need for manual work. From hand-delivered messages to telegraph, written communications in the past has required much manual labour. Today, email and a slew of instant messaging platforms get the same job done instantaneously and better. In recent years, highly advanced forms of automation involving AI are making headways into the mainstream workplace. Examples include manufacturing robots learning how to perform bin picking, robot patrol enforcing social distancing in midst of a virus situation [1] and many more. Automation therefore has an overall effect of moving human workers up the cognitive value chain, shifting towards increasingly managerial and strategic roles that are more knowledge-based. The reason for this is apparent as automation introduces characteristics such as being stronger and more tireless relative to human workers, and thereby allowing businesses to do more. Taking a step further, AI is also beginning to encroach into the

cognitive realm at work whereby as an instance, an insurance company reportedly replaced its employees with an AI system [2]. All things considered; it is understandable why the workforce is under a constant drive to do more.

Generally, however, automation is well-suited only for tasks that are repeatable within a fixed context. Contrast to this, the handling of work tasks across shifting contexts is not a good candidate for automation. As an example, in the face of a widespread consumer behaviour change due to a pandemic, AI models supporting sentiment analysis, fraud detection, marketing and inventory management operations no longer behaved as expected. The article [3] writes:

> *What's clear is that the pandemic has revealed how intertwined our lives are with AI, exposing a delicate codependence in which changes to our behavior change how AI works, and changes to how AI works change our behavior. This is also a reminder that human involvement in automated systems remains key.*

Though AI models are designed for robustness to changes in the incoming data, situations like these reveal the AI models' brittleness when there is a significant shift in input data distribution. This is termed as *out-of-distribution* (OOD). This means that the input data at point of inference is no longer the same as the training data's distribution that the AI model learnt from. This has the negative effect where the AI model not only potentially makes a mistake on OOD inputs, but even confidently classifying it as a known class. Clearly, this is undesirable and in critical deployments, the mistake can be costly. Clearly, this is undesirable and in critical deployments, the mistake can be costly.

To counter this, there are research efforts looking into OOD detection. To illustrate with an example, a deep generative model for OOD detection was trained using in-distribution genomic sequences [4], with the log-likelihoods plotted for both in-distribution and OOD inputs. We can see from the results (**Figure 1**) that the histogram of log-likelihood overlaps significantly for both in-distribution and OOD inputs, thus showing the model's inability to differentiate between in-distribution and OOD. The authors further note that their observations are not in isolation and are congruent with earlier works using image data.

Naturally, artificial general intelligence (AGI) comes to mind when we broach the topic of narrow AI (as afore discussed). We can think of AGI as the AI's ability being on par or exceeding that of a human's ability to learn, understand and perform intellectual tasks. To conceptualise the relationship of narrow AI and AGI, one may think of them as two sides of the same coin or rather, both ends of the same AI spectrum. Simply put, advancements in narrow AI will evolve towards AGI ultimately. Consensus indicates that AGI is not here today [5, 6] and predictions for the

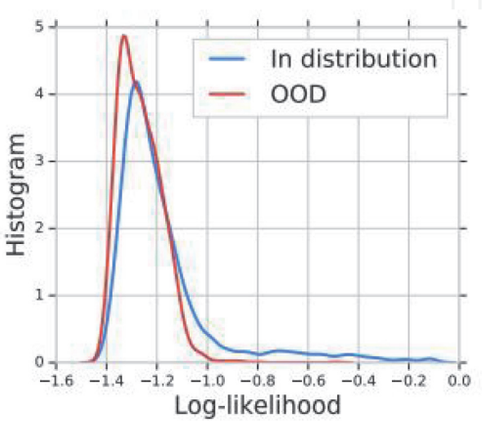

Figure 1.
Log-likelihood hardly separates in-distribution and OOD inputs, adopted from [4].

advent of AGI ranges widely anywhere from a few years to many decades away. Towards this goal, there are research directions on multiple fronts such as to progress deep learning from its current System 1-like abilities to be System 2-capable [7], improving language understanding through increasingly larger and complex language models such as BERT [8] (and its variants) and GPT-3 [9], etc. These are tangible indications that AGI is still some way off.

Meanwhile, we believe there is a complementary and parallel research direction to advancing narrow AI towards AGI. And that is human AI collaboration in which we exploit AI (be it narrow or AGI) to augment our natural abilities. In this sense, for as long as work is envisioned to involve humans, the pursuit for human AI collaboration remains a valuable research direction which will only be more relevant and strengthened further by AI's advancements and increasing pervasiveness at work. In the following sections, we will progressively present our proposed framework designed to enable the collaboration within a network of humans and AI. To begin, we will first discuss some background concepts relevant towards our proposed work.

2. Type theory: formal language of terms and types

Type theory is a branch of symbolic logic in mathematics. The theory of types was conceived to address Russell's paradox arising from naïve set theory, which says that any definable collection is a set. If we define a set of all sets that do not contain themselves, the paradox is illustrated when the set is not a member of itself and therefore needs to contain itself, which then leads to a contradiction of its own definition (Eq. (1)).

$$Let\ S = \{x \in Set | x \notin x\}. Then\ S \in S \Leftrightarrow S \notin S. \tag{1}$$

We can think of type theory as a formal language, complete with a set of rules that inform us on the construction and computations for strings of symbols which we will further introduce hereon the concept of *terms* and *types*.

$$a : A \tag{2}$$

We begin with a basic representation (Eq. (2)), more formally referred to as a judgement, often encountered in type theory, which simply means that a is a term of type A. We can also think of this as a is an element of A. To illustrate for further clarity, we define some examples of terms and types using some real-world objects:

- $red, green, yellow, orange : Colour$
- $mary, john, isaac, caleb : Name$
- $apple, strawberry, banana, orange : Fruit$
- $chicken, fish, beef, turkey : Meat$

In type theory, every term must have a type as seen in List 2. For ease of understanding, we can loosely correspond this to the set theoretic statement $a \in A$ where red, green, yellow and orange are all members of type Colour. Another correspondence can be thought of as *propositions as types* under the Curry-Howard isomorphism [10], where A is a proposition, then a validates the existence of A. To

provide the evidence, we will have to perform some mathematical operation to construct the object, that is the term inhabiting the type. Here we observe that *constructivism* is a foundational aspect of type theory, meaning we cannot just assume that objects exist through means such as "Suppose that there exists such an object … ." or by proving the existence of some object through deriving a contradiction from the assumption that object does not exist.

Like other concepts in mathematics, the construction of an object in type theory is governed by rules and we will focus on the introduction of the relevant concepts in the following sections.

2.1 Function type

To start, we look at the *function type*. Given types A and B, we can construct the type $A \to B$ of functions mapping from domain A to codomain B. If we define a function f of type $A \to B$ and apply to the term a of type A, we can obtain a term $f(a)$ of type B (also can be written as $f(a) : B$). For function types, the mapping between the domain to the codomain is constant and fixed. Let us look at a more relatable and practical example of a function type:

- We define a function *head* that returns the first element of a list.
- *head* belongs to the type $List\, a \to a$
- We apply *head* to a list $[1, 2, 3]$ and result is 1.
- If we apply *head* to a list, the result will be.

Hence, a function type will always map from some domain A to the codomain B.

2.2 Dependent product type

Next, we introduce the *dependent product type* for which its terms are functions whose type of codomain varies depending on the term of the domain that the function is applied to. It is also referred to as a *dependent function type* or $\prod$*-type*. Given a type A and a family of types $B : A \to U$, we can construct the type of dependent products $\prod_{x:A} B(x) : U$ where U is known as *universe* whose elements are types. The dependent product type is often used in type theory, and we can think of it as a more generalised form of the function type. The main difference lies in B being a constant family in the function type, such that $\prod_{x:A} B \equiv A \to B$.

To illustrate, let us define f as a dependent function of type $\prod_{x:A} B(x)$ and apply term a of type A. The result is such that we obtain a term $f(a)$ of type $B(a)$ (also can be written as $f(a) : B(a)$). We further provide a more relatable example of a dependent product type as follow:

- We define a dependent function *intOrString* of type $\prod_{x:Boolean} intOrString(x)$ that returns an integer or string depending on the input $true, false : Boolean$.
- We further define terms and types as $11 : intOrString(true)$ and *'hello' intOrString* $(false)$.
- If we apply *intOrString* to the term *true* of type *Boolean*, an integer 11 is returned.

- If we apply *intOrString* to the term *true* of type *Boolean*, a string *'hello'* is returned.

- Note that $intOrString(true)$ and $intOrString(false)$ are different types.

Hence, a dependent product type will map to a different codomain depending on the input term.

2.3 Propositions as types

Earlier on, we briefly talked about the correspondence referred to as *propositions as types*. To validate the truth about a proposition, it means that the corresponding type needs to be inhabited by some term and this is the *evidence* (or *witness*) to the proposition. Generally, the evidence will not be constructed explicitly but rather, translated from proofs into a term of a type and in this sense, it feels like classical set theory reasoning. However, a proposition in type theory goes beyond being true or false, to being a collection of all possible evidence towards the proposition's truth. This mirrors much of our real-world work scenarios, in the sense that there is often more than one correct (true) way of fulfilling a task.

Furthermore, the correspondence between type theoretic and logic operations (**Table 1**) allows us to syntactically construct a type theoretical operation with the semantics of the corresponding logical operation. This is significant because with the ability to correspond between type theoretical and logical operations, the evidence (or proofs) are therefore first-class mathematical objects instead of being just a means for communicating mathematics.

Although it may not be immediately apparent, what we just discussed has impactful implications, mainly:

- Logical operations are integrated within the type theoretical operations, thus combining semantics and syntax. Hence, under the paradigm of propositions as types, a proposition is true and valid when we provide a term to the type. In other words, the type (proposition) is now inhabited by a term (evidence).

- As we are operating with first-class mathematical objects within type theory, this introduces an important aspect: Computability. This gives rise to a further correspondence which is termed as evidence (or proofs) as programs.

- Due to the constructivism nature of type theory, terms are constructed through a set of rules introduced earlier within Section 2. This introduces another important aspect: Explainability.

Logical	**Type Theoretic**
True	1
False	0
Not A	$A \to 0$
A and B	$A \times B$
A or B	$A + B$
A implies B	$A \to B$
A if and only if B	$(A \to B) \times (B \to A)$

Table 1.
Correspondence of logical and type theoretical operations.

2.4 Reasoning through structured types

Type theory can be viewed as a mathematical formalisation for a programming language. Examples of such programming languages include Agda, Coq, Haskell and more. One notable usage is in proof assistants, that resulted in a verifiable proof of the four colour theorem [11] well over a century after its introduction in 1852. Another notable usage is in formal program verification, which is a software programming paradigm that ensures that the resulting computer program has the rigour of a mathematical proof. This is achieved through specifying how a program should behave and ensuring that it works as specified, which is synonymous with the creation and proof of a mathematical model. Beyond the guarantee of the program's correctness, this has significant implications on cyber security in our highly connected digital society.

Though dependently typed functional programming is not mainstream at the point of writing, it is on the rise and initiatives such as CompCert [12] are active in taking functional programming forward. In concluding Section 2, we find the following quote [13] useful as a succinct summary of type theory:

In type theory, unlike set theory, objects are classified using a primitive notion of type, similar to the data-types used in programming languages. These elaborately structured types can be used to express detailed specifications of the objects classified, giving rise to principles of reasoning about these objects.

3. Group-assign: type theoretic framework for human AI orchestration

Having discussed the background and relevant concepts, we will describe our proposed work hereon. We will first start off with a summary of what the framework is and what it does in Section 3.1. Following this, we will further describe the framework details, methodology and associated terminologies over the subsequent sections. We will also openly discuss about the design considerations that influence the current version of our proposed framework. This is done with the key purpose for sharing our research thoughts through the journey of developing our framework, to better inform future interested parties on how they can leverage and further our proposed work.

3.1 Framework overview and contribution

While the idea of human-machine or human-computer collaboration is not new and different ideas have been proposed. To the best of our knowledge, we are one of the first to propose the use of type theory as a language to orchestrate and describe human-machine collaboration. In our proposed framework, we define a fundamental set of type theoretic rules:

- Base form of intent representation.
- An intent can be applied to different data.
- An intent may result in any number of possible implementations.
- An intent may be composed of one or more constituent intents.

We also define abstract functions for *Group* and *Assign* as base methodologies within the framework to handle data and assigned towards associated implementations.

As an implementation to the type theoretic framework, we develop a prototype using Python that allows us to orchestrate independent declaration of intent(s) and

instantiating the *intent(s)* with associated data and implementations, visualised as a simple directed graph that can be recursively built upon a *intent-data-implementation* pattern. Collectively, this graph represents a work plan (e.g. Running a fast food restaurant) in the real world. Each node symbolises some real-world human intent, data group, implementation. For example, "Cook Burger Patty" is an intent that can be instantiated with "Chicken", "Beef" as data groups associated to "Ten steps to cook a burger patty" as an implementation.

3.2 It all begins with an intent

In the context of our proposed work, we define intent simply as "the desire to do something (carry out an implementation)". It is beyond the scope, however, to discuss or quantify intent from a philosophical or psychological view. Before we do something, we first have the intent to do so and the intent does not always necessarily lead to any tangible implementation. Here, we introduce the distinction between intent and implementation.

Work tasks often involve multiple actions and, in this sense, can be considered complex. In undertaking the task, we form an overall intent (which is to complete the task) comprising of constituent intents, which together represents an abstract plan to manage the task. For example, to set up a meeting, we will need to check for the meeting room availability, attendees' availability and then determine the best common time slot. The overall intent in this example is "set up a meeting" with the rest being constituent intents. While "check for meeting room availability" and "check for attendees' availability" can be independent, "determine best common time slot" will depend on these two constituent intents. A constituent intent may depend on one or more other constituent intents or it may also be independent from (existing alongside) other constituent intents. Here, we introduce the notion of a hierarchy of constituent intents within the context of an overall intent.

It can be challenging when we talk about intents. Horizontally across a company, different people in the similar job tiers can have different views about the same thing. Vertically, people across the job tiers will see things at different granularity. Using the same illustration of setting up a meeting, a manager may just instruct the team to set up a meeting. The team member in charge will probably add more constituent intents such as checking for meeting room and attendees' availability because "set up a meeting" is insufficient to fulfil the task. This is an example of vertical granularity differences. Given if another team member is put in charge, he/she may also handle it differently and perhaps add "Cater for coffee and tea" as a constituent intent. This is an example of horizontal diversity. Therefore, to achieve the overall intent (some collective goal), it is important to have the ability to connect diverse and distributed intents in a robust manner.

With these design considerations in mind, framework design principles are summarised as follow:

- Intent and implementation are distinct.
- An intent (within a context) can contain a hierarchy of constituent intents.
- An intent may depend on other constituent intents.
- An intent may have one or more possible implementations.
- Intents should be connected in a robust manner, enabling explanability.

3.3 What is the language for connecting intents?

We earlier discussed about the importance of connecting intents. And by connecting intents, we are composing some work plan. More generally, we are composing a structure and examples abound as we live in a world filled with structures. Examples of structures exist in buildings, deoxyribonucleic acid (DNA), literature, music and many more. The principle of compositionality [14] states that:

> *For every complex expression e in language L, the meaning of e in L is determined by the structure of e in L and the meanings of the constituents of e in L.*

Reasoning is not monolithic and whether as an individual or a team, reasoning is compositional in nature. As we saw earlier, works in AI (both symbolic and neural) are also looking to emulate this behaviour within their AI models. However, intents are intangible and formless. We cannot know what another's intent is unless it is expressed. From a human to human perspective, we compose expressions using some language (e.g. English, Chinese, French, German, etc.) to convey our intents to each another, and the success of it depends on both parties understanding the language as well as whether the expression is well-formed. This takes place so commonly in our daily lives that most of us likely have taken for granted the underlying significance. Hence, an expression is a proxy of our intent and the language is what enables the connection of intents.

Progressing into a future where humans and AI collaborate, will a human language suffice? The answer is clearly no. This is where we believe type theory will serve a suitable and important role in our proposed framework as the language (syntax) that allows users of the framework to define and connect intents and associated implementations (semantics) in a principled way.

3.4 Intents as types can be understood by machines

Humans are not precise and often ambiguous in expressing our intents. Clearly, there is no metric for compatibility and level of abstraction when it comes to human intents. The level and the type of details we deem important and sufficient vary accordingly based on our experience. But with machines, precision and non-ambiguity is critical for things to work.

We believe type theory serves a key role in our proposed framework as the language (syntax) that bridges humans and machines. By embedding human-expressed intents within our type theoretic framework, we posit that the expressiveness (and ambiguity) of humans can be preserved while simultaneously having the precision that machines require in order to function.

This allows users of our framework to define and connect human intents and associated machine implementations (semantics) in a principled and precise manner that also allows for diversity and distributed contributions from multiple parties as is reflective of real world conditions.

Earlier, we presented the idea of correspondence in type theory such as "propositions as types" and "proofs as programs". In our proposed framework, we further introduce a correspondence termed *intents as types* (**Figure 2**).

$$implementation : Intent \tag{3}$$

Recall we established that an intent is distinct from its implementation and an intent may have one or more implementations. This structure is a natural correspondence to the basic representation of term and type (Eq. (2)) earlier introduced

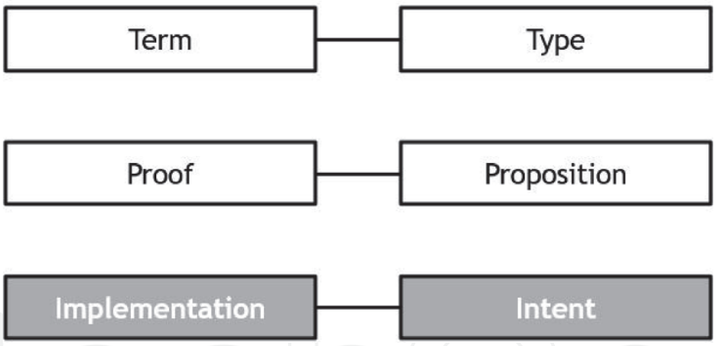

Figure 2.
Intents as types.

and we will represent the base form of an intent in a similar type theoretic manner (Eq. (3)). Hence, we can easily understand this correspondence as:

- Intents as types
- Implementations as terms

By establishing intents as types, we have effectively laid down the foundation of our proposed framework from which we will further extend its functionalities.

3.5 Separation of intent and implementation

In our proposed framework, there is a profound significance underlying intents as types. And this is because it allows for the separation of intents and implementations, which we believe to be critical towards enabling collaborations. Today, intent and implementation are intertwined which can be seen from how systems often specify and dictate how we carry out tasks in fulfilment of some intent. However, in context of the knowledge workplace, this is probably too draconian and rigid where the creativity and autonomy of individuals are especially valued. Furthermore, in reality, intents are often separate from implementations and may even be contributed by different people. Particularly, collaboration at scale is complex and we cannot reasonably expect it to be well-defined or pre-defined from the onset. On contrary, we can expect that for any collaboration:

- Multiple parties are involved.
- Coordination needs to happen vertically and horizontally within the organisation's hierarchy.
- People's ideas and ways of doing things can be fluid, dynamic and diverse.

Essentially, we need to flexibly handle the division of interdependent labour, interconnected intents, and diverse methods of handling the task at hand. Therefore, the question is: how can we tie people's collaboration together - managing the flow of information, etc. - allowing each person to define what they can do for others without overtly constraining their implementation, then allowing each task with input data to be broken into groups of data which can be handled with different implementations?

To achieve this, we believe that there needs to be a separation of intent from the implementation using type theory, enabled through our proposed framework.

3.6 Framework axioms

Next, we derive the rules (axioms) of our proposed framework which will govern the operations in a type theoretic manner.

Given some data $x : X$ of type X and an intent $G(x)$, the output is some implementation $g(x)$. We represent this statement type-theoretically in Eq. (4), which relates an implementation (term) to its intent (type).

$$g(x) : G(x) \tag{4}$$

Alternatively, we can write

$$g : \Pi_{x:X} G(x) \tag{5}$$

where we view g as a term of a dependent product type.
To fulfil an intent $G(x)$:

- there may exist data groups or subtypes $X_1, X_2, \ldots, X_k \subseteq X$ where the intent former G can be applied. This means that for different data $x_1 : X_1, x_2 : X_2, \ldots, x_k : X_k$, we could form different intents.

$$G(x_1), G(x_2), \ldots, G(xk) \tag{6}$$

- Given some data $x_i : X_i$, the intent $G(x_i)$ may have one or more implementations. Moreover, there may exists any number of possible strategies or implementation formers $g_1, g_2, \ldots, g_m$ for constructing implementations for $G(x_i)$.

$$g_1, g_2, \ldots, g_m : G(x_i) \tag{7}$$

- Each implementation former g_j for $G(x_i)$ may consume the outputs from one or more constituent intents $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$ and associated implementations may receive inputs from one or more constituent intents (Eq. (8)). This means that an intent may contain its own hierarchy of constituent intents.

$$\begin{aligned} \gamma_1 : \Gamma_1(x_i) &\rightarrow \gamma_2 : \Gamma_2(x_i, \gamma_1) \\ &\rightarrow \ldots \\ &\rightarrow \gamma_n : \Gamma_n(x_i, \gamma_1, \ldots, \gamma_{n-1}) \\ &\rightarrow g_j(x_i, \gamma_1, \gamma_2, \ldots, \gamma_n) : G(x_i) \end{aligned} \tag{8}$$

In summary, these framework rules allow us to:

- Define and construct intents.
- Connect intent to constituent intents.
- Associate an intent with its implementation(s) and related data.

3.7 Group and assign

Next, to complete the framework, we will introduce two algorithms, Group and Assign, as abstract methods respectively described in Algorithm 1 and Algorithm 2.

The Group function (Eq. (9)) is defined as: For every intent, we have a set of data that can be further grouped into smaller groups based on some grouping criteria J.

$$Group(G, [x]) \rightarrow (G, [(x, j)]) \tag{9}$$

Algorithm 1 Group.

1: **Input**: G, $[x]$ where $x \in X$, J where $j \in J$
2: **Output**: $(G, [(x, j)])$
3: **Initialise**:
4: $L \leftarrow \emptyset$ where L is a placeholder list to collate all (x, j) pairs
5: **for** each j in J **do**
6: **if** x matches criteria j **then**
7: $L \leftarrow (x, j)$
8: **end if**
9: **end for**
10: $(G, [(x, j)]) = (G, L)$
11: **return** $(G, [(x, j)])$

The Assign function (Eq. (10:)) is defined as: For each group belonging to an intent G, some implementation g is defined and applied to the group.

$$Assign\left(G, \left[(x, j]\right) \rightarrow \left(G, \left[x, j, g_j(x, \gamma_1, \gamma_2, \ldots, \gamma_m)\right]\right)\right. \tag{10}$$

Algorithm 2 Assign.

1: **Input**: (G, [(x, j)])
2: **Output**: $\left(G, \left[\left(x, j, g_j\right)\right]\right)$
3: **for** each (x, j) in $[(x, j)]$ **do**
4: **if** some g exists for (x, j) **then**
5: $\left(x, j, g_j\right) \leftarrow (x, j)$.append g_j
6: **end if**
7: **end for**
8: **return** $\left(G, \left[\left(x, j, g_j\right)\right]\right)$

Our proposed framework is collectively formed by the framework rules (Section 3.6), Group and Assign. This completes the description of our framework and in the following sections, we will discuss the evaluation strategies and findings for our proposed framework.

3.8 Evaluation approach

Our proposed work brings together type theory, type theoretic framework axioms and associated functionalities as a human AI intent orchestration framework intended for real world application. To the best of our knowledge, this is a novel effort and uniquely positioned idea to introduce capabiltiies for enabling a

collaborative human AI future. This concurrently presents a challenge for us in determining how best to provide an evaluation of the proposed framework as widespread adoption and understanding will require more time and effort beyond our scope of research, as is reasonably expected given that the collaborative human AI society has yet to be a norm at the point of writing.

Going into the future, we intend to utilise our proposed framework and progress beyond our preliminary evaluation efforts to progressively identify and engage external parties for further evaluation through joint collaborations. Nevertheless, we endeavour to provide an evaluation of our proposed framework here and therefore consider the following:

- What is the closest and most relevant domain for our proposed framework?
- In reference to this domain, what are some useful evaluation strategies?

We believe that evaluation strategies for a toolkit (which we liken as comparable to our proposed framework) from the domain of human computer interaction [15] is relevant and suitable. We also note that the authors have expressed that "The problem is that toolkit evaluation is challenging, as it is often unclear what 'evaluating' a toolkit means and what methods are appropriate.", which speaks of a similar challenge for us and is still commonly faced till date by researchers of toolkits. Concretely, we reference and adopt two well-established evaluation strategies for the purpose of our evaluation, namely:

- Demonstration: As the name suggests, this evaluation strategy shows what the framework can support and how users might potentially use the framework through means such as using examples to illustrate a variety of possible applications. And it helps with the question of "What can be done with the framework". For our proposed work, we conduct this in the form of a "How To" scenario which is a technique of the demonstration evaluation strategy. Essentially, it is a walkthrough on a step-by-step breakdown of the workflow into individual steps and its associated results. Following this method, we demonstrate using a real-world context and step-by-step breakdown of how the framework orchestrates the intents of multiple users (Section 3.9).
- Usage: This particular evaluation strategy involves a user group in how they work with the framework, which helps with verifying aspects such as conceptual clarity, ease of use, value as perceived to the target user group, etc. And it helps with the question of "Who can use the framework". In practice, this is complementary and often combined with demonstrations. For our proposed work, we conduct this in the form of a walkthrough which is a technique of the usage evaluation strategy and gather the users' overall impressions. Using this method, we show our proposed framework to our potential users for their feedback and impressions. As a further enhancement, we also made this an interactive walkthrough where the potential users participated under the context of a work task in our team (Section 3.10).

3.9 "How to" scenario

To illustrate the walkthrough, we utilise our prototype software library and further implement a demo application built on top to illustrate plausibility and practical usability of our proposed framework. **Figures 3–6** are screenshots taken from the demo application.

DOI: http://dx.doi.org/10.5772/intechopen.96739

To begin, let us suppose the scenario where we are planning for a fast food restaurant operations. Serving meals to our customers would be core to our business. In this context, "Serve Meal "would therefore be an overall intent from a management perspective. Another person on the team might look at this and suggest that we offer nuggets. Another then contributes that selling burgers will be a great idea too. Now, we have three intents altogether (**Figure 3**): "Serve Meal" and two constituent intents, "Serve Nugget" and "Serve Burger".

Subsequently, another team member points out that a meal would only be complete with a drink and further contributes "Serve Drink" (**Figure 4**). Here, we see that the framework is flexible to handle contributions of intents from different parties in a distributed manner. We could go on and add more intents, but this will suffice in this walkthrough for now.

At this point, we start to have a semblance of a plan on our food menu strategy and at work, this is what is often referred to as a "high level" view. However, it clearly still lacks further granularity such as:

- What type of burger?
- What type of drinks?
- What type of nuggets?
- What are the types of meal combinations we want to offer in our food menu?

We decide then that what we serve will depend on our inventory except we will not serve beef nuggets because it is not a norm. Looking through our inventory, we have chicken and beef in our raw meat inventory and coca-cola in our drinks. So, we will make beef burgers, chicken nuggets and coca-cola drinks. In doing this, we have effectively created groups of data based on some criteria and associated them with the corresponding implementation and intent (**Figure 5**).

Finally, we decide to offer 2 types of meal: Beef burger with coca-cola and chicken nuggets with coca-cola. This leads us to having a complete plan (**Figure 6**): We know what we want to do (Intent), we know how to do it (Implementation) and we have the ingredients (Data).

It is clear that different levels of details (i.e. intents, implementations) are often being handled by different workers horizontally as well as vertically within a company hierarchy. Any work operations of some scale will necessitate this division of labour. This is where disconnects will happen because there are no guarantees of higher level details connecting with more granular details, especially more so when the big picture is contributed by many different workers.

Figure 3.
Constructing intents in the context of menu planning for a fast food restaurant.

Figure 4.
Adding an intent "serve drink".

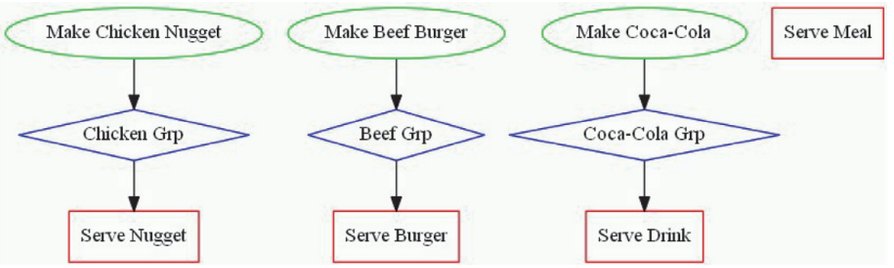

Figure 5.
Associating intent-data-implementation. Intents are represented as red rectangles, data is represented as blue diamonds and implementations are represented as green ovals.

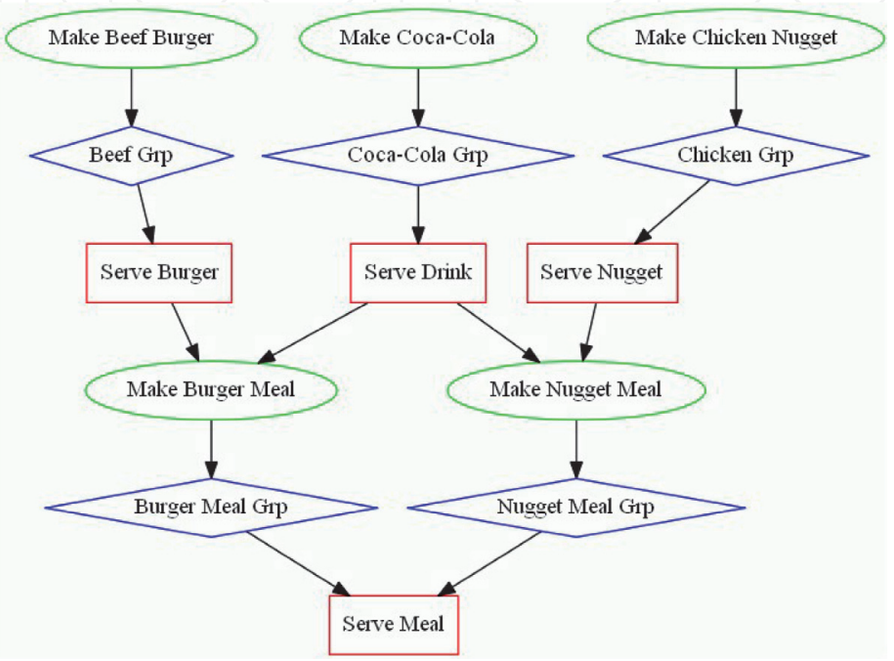

Figure 6.
Completed plan for offering fast food meals.

3.10 Usage feedback

To collect usage feedback, we conducted an interactive walkthrough on a work planning task for strategising our business unit growth, where we collaboratively develop a workplan by using our proposed framework principles. For confidentiality reasons, the actual content of the workplan will not be documented here. However, the process is similar in nature as our detailed step-by-step breakdown in Section 3.9. Following the conclusion of our interactive walkthrough, the user feedback for our proposed framework are summarised as follow:

- Structured yet flexible: The intent-data-implementation pattern for building up a workplan provided a clear structure for interdependent labour, interconnected intents, and diverse methods of handling the task at hand. However, there is still much flexibility for expressing intents and implementations, and the structure does not inhibit this flexibility.

- Clarity and precision: It is apparent from the workplan to see which intents are unfulfillable due to the inability to provide implementations or data for. While

this feels rigid initially, it is subsequently well-accepted as there is a recognition that we cannot simply speak about our intents at work without the means to make it happen.

- In summary, we like to highlight some key points:

- The workplan can be easily visualised as a simple directed graph that is recursively constructed through primitive blocks comprising of 3 types of nodes: Intent, Implementation and Data.

- Beneath the apparent simplicity, the workplan is type theoretic and built upon the rules and algorithms (Group and Assign) described respectively in Sections 3.6 and 3.7. This confers any workplan built upon this framework with the desirable type theoretic properties discussed in Section 2.

From our evaluations, we demonstrate the plausibility of our proposed framework towards its intended goal for orchestrating work plans across a heterogeneous network of human intents associated with AI/human implementations and data.

4. Conclusions

In summary, we can see that the framework orchestrates the intents and associated implementations from different people while keeping the intent and implementation separate.

- This allows for each person to define what they can do for others in a distributed manner, while also enabling them the flexibility and freedom to provide their implementations they deem best.

- More so, by utilising primitive blocks of intent-data-implementation, the workplan is built up in a type theoretic manner where dynamic dependencies can be captured and represented clearly.

- This essentially makes the workplan a mathematical model which can be fully described using type theoretic expressions and hence is computable and constructive in nature.

4.1 Challenges and future directions

At the beginning of the chapter, we established the significance of human AI collaboration, and proceeded to share about our proposed work and its intended contributions towards this goal. In ending this chapter, it is apt to candidly discuss about its potential challenges and associated future directions. To do this, let us expand our view of human AI collaboration beyond the technological lens. Again, what then is human AI collaboration? It is a relationship, fundamentally. Like any successful relationship, trust and communication are crucial. Let us discuss each of these factors:

- Trust. This represents our confidence level in relying on the output from our AI counterpart. How do we ensure that the AI is performing as it should? Is there transparency in the way the AI operates? Are we able to interpret and understand why the AI does what it does? For example, standard-setting

organisations define criteria for many technologies to ensure that compliance guarantees quality, digital security protocol such as SSL ensures the security for internet communications, well-documented manuals aid us in product troubleshooting and maintenance, etc. Every new technology introduced would eventually face questions such as these. Going forward, an area of potential interest lies in the framework integration with proof assistant capabilities. Work plans built upon our proposed framework are type theoretic in nature. The broad idea is therefore to treat every work plan as a theorem to be proven. By proving the theorem (work plan), the strong implication is that the work plan is verified to be working as intended. The ability to frame real world work plans as a mathematical model has desirable benefits in terms of trust, and this is an area which we hope to investigate more deeply.

- Communication. This represents how human and AI convey and exchange information. While our proposed work contributes towards a facet of human AI collaboration to enable the description and orchestration of intents across a network of humans and machines, it is targeted and focused as is the nature of research. As an analogy, while networking protocols enable the exchange of information over the Internet, it does not inherently make information easily searchable by users. For this, technologies such as search engine come into play. Switching back to our context here, an interesting aspect of communication (beyond our proposed work) would be to consider how these intents (along with its associated implementations and data) can be made discoverable and reusable by others.

Naturally, our discussion here is by no means exhaustive. It is our intent and hope that our proposed work and discussion contributes towards and catalyse future discussions in the research community for the continued advancements of human AI collaboration and ultimately, towards the future of a collaborative human AI society.

Acknowledgements

In making this work possible, I gratefully acknowledge the support of NVIDIA and research funding from Singapore Economic Development Board that enabled the opportunity for conducting this work.

DOI: http://dx.doi.org/10.5772/intechopen.96739

Author details

Aik Beng Ng[1*], Simon See[1], Zhangsheng Lai[1] and Shaowei Lin[2]

1 NVIDIA Corporation, San Tomas Expressway, Santa Clara, CA

2 Singapore University of Technology and Design

*Address all correspondence to: aikbengn@nvidia.com

References

[1] CNA. Meet the robot dog promoting safe distancing in Singapore's parks [Internet]. 2020. Available from: https://www.channelnewsasia.com/news/singapore/meet-the-robot-dog-promoting-safe-distancing-in-singapore-s-12716544 [Accessed: 2020-05-09]

[2] BBC. Japanese insurance firm replaces 34 staff with AI [Internet]. 2017. Available from: https://www.bbc.com/news/world-asia-38521403 [Accessed: 2017-01-05]

[3] MITTR. Our weird behavior during the pandemic is messing with AI models [Internet]. 2020. Available from: https://www.technologyreview.com/2020/05/11/1001563/covid-pandemic-broken-ai-machine-learning-amazon-retail-fraud-humans-in-the-loop/ [Accessed: 2020-05-11]

[4] Jie Ren and Peter J. Liu and Emily Fertig and Jasper Snoek and Ryan Poplin and Mark A. DePristo and Joshua V. Dillon and Balaji Lakshminarayanan. Likelihood Ratios for Out-of-Distribution Detection. In: Advances in Neural Information Processing Systems (NEURIPS '19); 2019. p. 14707–14718

[5] Cem Dilmegani. 995 experts opinion: AGI / singularity by 2060 [Internet]. 2021. Available from: https://research.aimultiple.com/artificial-general-intelligence-singularity-timing [Accessed: 2021-02-08]

[6] Federico Berruti and Pieter Nel and Rob Whiteman. An executive primer on artificial general intelligence [Internet]. 2021. Available from: https://www.mckinsey.com/business-functions/operations/our-insights/an-executive-primer-on-artificial-general-intelligence [Accessed: 2021-02-08]

[7] NIPS. From System 1 Deep Learning to System 2 Deep Learning [Internet]. 2019. Available from: https://nips.cc/Conferences/2019/ScheduleMultitrack?event=15488 [Accessed: 2019-12-11]

[8] Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers); 2019. p. 4171–4186

[9] T. Brown and B. Mann and Nick Ryder and Melanie Subbiah and J. Kaplan and Prafulla Dhariwal and Arvind Neelakantan and Pranav Shyam and Girish Sastry and Amanda Askell and Sandhini Agarwal and Ariel Herbert-Voss and G. Krüger and T. Henighan and R. Child and Aditya Ramesh and D. Ziegler and Jeffrey Wu and Clemens Winter and Christopher Hesse and Mark Chen and E. Sigler and Mateusz Litwin and Scott Gray and Benjamin Chess and J. Clark and Christopher Berner and Sam McCandlish and A. Radford and Ilya Sutskever and Dario Amodei. Language Models are Few-Shot Learners. In: 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada; 2020.

[10] Sørensen, Morten Heine and Urzyczyn, Pawel. Lectures on the Curry-Howard isomorphism. 1st ed. Elsevier Science; 2006. 456 p. Hardcover ISBN: 9780444520777

[11] Gonthier, Georges. A Computer-Checked Proof of the Four Colour Theorem [thesis]. Microsoft Research Cambridge; 2005.

[12] Inria. COMPCERT [Internet]. 2008. Available from: http://compcert.inria.fr/ [Accessed: 2020-12-03]

[13] The Univalent Foundations Program. Homotopy Type Theory: Univalent Foundations of Mathematics [Internet]. 2020. Available from: https://homotopytypetheory.org/book [Accessed: 2020-12-03]

[14] Szabó, Zoltán Gendler. The Stanford Encylopedia of Philosophy [Internet]. 2017. Available from: https://plato.stanford.edu/archives/sum2017/entries/compositionality/ [Accessed: 2020-12-03]

[15] Ledo, David and Houben, Steven and Vermeulen, Jo and Marquardt, Nicolai and Oehlberg, Lora and Greenberg, Saul. Evaluation Strategies for HCI Toolkit Research. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18); 2018. p. 1–17