Concept Paper

Intent-Driven Modeling and Management of Complex Adaptive Systems – Proposed Approach and Research Agenda

Nanjangud Narendra * and Nithin Nagaraj

Posted Date: 6 May 2026

doi: 10.20944/preprints202605.0367.v1

Keywords: complex adaptive systems; knowledge graphs; compositional reasoning; intent-driven management; intent decomposition

 doi:10.20944/preprints202605.0367.v1

Concept Paper

Intent-Driven Modeling and Management of Complex Adaptive Systems – Proposed Approach and Research Agenda

Nanjangud Narendra * and Nithin Nagaraj

Complex Systems Programme, National Institute of Advanced Studies (NIAS), IISc Campus, Bangalore 560 012, India
* Correspondence: ncnaren@gmail.com

Abstract

Complex adaptive systems (CAS) have two defining characteristics. First, they are complex, i.e., composed of several interacting parts. Second, they are adaptive, i.e., their behavior can be changed in response to external stimuli and changes in the external environment. Due to this, managing such systems is quite challenging. Traditional approaches have involved defining policies that determine the behavior of any CAS under particular circumstances. However, such approaches are rigid and inflexible, since they are dependent on pre-specified policies. To that end, in this position paper, we describe an *intent-driven* approach to modeling and managing CAS. This would be a more flexible approach, not dependent on any specific policies, but which can be customized based on the context in which the CAS is functioning. We describe the various components of our approach, which include compositional reasoning to decompose the intent into sub-intents as per the context; mapping the sub-intents onto the execution model which will satisfy the intent; and feeding back the results of the execution to facilitate continual learning and continuous improvement in managing the CAS. In particular, one aspect that we highlight is the application of *neurochaos learning*, which uses chaos theory to facilitate rapid continual learning that would help improve the overall efficiency of our approach. For each component of our approach, we also present several research questions that need to be addressed before intent-driven management of CAS can become a reality.

Keywords: complex adaptive systems; knowledge graphs; compositional reasoning; intent-driven management; intent decomposition

---

1. Introduction

Complex Adaptive Systems (CAS) [1], are systems that have two key characteristics: (1) they are composed of several interacting components, and (2) their behavior can change in response to both external stimuli and internal changes. Examples of CAS abound, e.g., telecommunication systems, transportation systems, manufacturing networks, supply chain networks, etc.

Typical management and handling of CAS has involved defining policies and rules that determine its behavior under particular circumstances. However, since these are already prespecified and difficult to change on the fly, this makes CAS management rigid and cumbersome. To that end, recent work on *intent-driven management* [2] promises to address this issue. Intents are goals that are at the same time rigorous and also can be specified in a manner understandable by humans. They can then be dynamically translated into policies, rules and actions that meet the intents. So far, most applications of intent-driven management have been in the telecommunications area, e.g., [3], but it is being applied in other areas such as cloud network management [4] and supply chain management [5].

To that end, in this position paper, we present our approach for intent-driven modeling and management of CAS. It encompasses a complete intent management lifecycle, from intent modeling, intent decomposition, mapping to an implementation model, and reporting the implementation

 doi:10.20944/preprints202605.0367.v1

results to help further improve intent decomposition and implementation model mapping in the future. In so doing, we present our proposed techniques that would need to be incorporated, such as, (a) compositional reasoning [6] for intent decomposition, (b) goal-oriented derivation [7] for implementation model mapping, (c) chaotic continual learning [8] to improve the quality of intent decomposition and implementation mapping in the future. In addition, for each technique, we present a set of research questions that need to be addressed in order to make intent-driven management of CAS a reality.

The rest of our paper is as follows. We present background material in the next Section. In Section 4, we present an illustrative example from the logistics domain. In Section 5, we present our intent, context and execution models. In Section 6 we present our proposed approach for automated mapping of intent models to execution models as per the given context model. Section 7 presents the details of our approach, including the various agents needed to automate the fulfillment of the intent. In Section 8, we list out the key research questions that are needed to be addressed to make intent-driven management of CAS a reality. Finally our paper ends in Section 9 with suggestions for future work.

2. Background

2.1. Knowledge Graphs

A *Knowledge Graph* (KG) is a graph whose nodes correspond to entities and edge correspond to relations among the entities [9]. Each entity could belong to an *ontology* [10], which defines to which semantic class the entity belongs, and its position in the semantic class. For example, an entity that represents a person could belong to the category of users of a computer system, and the users could themselves be organized in a hierarchy which defines the position of the person in question in the ontology, viz., whether the person in question is a worker or manager. The entity could therefore possess certain attributes (e.g., name, ID, position in the hierarchy, department to which the person belongs) which can then be used to define *features* for the node representing the entity. These features can be grouped together as a *feature vector* representing the node in question.

Similarly, an edge in the KG represents a relation between the two entities that it links together. Examples of relations could be sibling, colleague, manager, subordinate, etc. This relation can also be attributed just like an entity.

Knowledge graphs (KG) [11] have emerged as the key knowledge representation method, especially for linked data, i.e., data that can be represented in the form of a graph. There has been extensive research so far on using knowledge graphs in a manner suitable for enabling machine learning [12]. Typically, this usage has involved generating node and/or edge embeddings of the KG and feeding those embeddings to a neural network to solve problems such as classification and link prediction [13].

2.2. Intent-Driven Management

An early definition of intent has been developed by the TeleManagement Forum [14], thus: an intent "is the formal specification of all expectations including requirements, goals, and constraints given to a technical system". This expectation would cover functional and non-functional requirements, examples of the latter being quality requirements. For example, transportation of a product from seller to buyer would be a functional requirement, whereas delivery of the product within a time limit would be a non-functional requirement.

Intents are not only intuitively understandable by humans, but they can also be rigorously specified to make them computer-readable. They can also be decomposed into sub-intents, in order to ease intent management. Finally, the lowest layer of sub-intents can be mapped onto execution models that are typically represented as process models and implemented as such [7].

In our approach, we represent intent models as KGs in a manner similar to that presented in [14], and execution models as dynamic process KGs in a manner similar to that presented in [6]. Hence our approach comprises the following two key actions: decomposing intents into sub-intents

 doi:10.20944/preprints202605.0367.v1

via compositional reasoning [6], and mapping intent models onto execution models via contextual analogical reasoning (CAR) [15**?**].

As described in [6], compositional reasoning is the task of breaking down complex problems into more manageable subproblems [16]. To augment this, rule-guided compositional learning [17] can also be added to it. (A similar model for the telecom domain has been presented in [18].) This is highly essential for complex adaptive systems (CAS), which possess the two key properties mentioned above: (a) composed of several interacting parts, and (b) constantly adapt as per user requirements and/or environmental conditions.

Compositional reasoning also involves CAR, which is the task of determining the "closest" solution to a problem or sub-problem from existing solutions that have been successful in the past. That is, some sort of "semantic similarity/comparative reasoning in the given context" [19] would need to be developed.

And to make this entire effort sustainable over the long run, this would also need *continual learning* [8]. This will ensure improved performance of the compositional reasoning system over time. It will also enable better adaptiveness of the CAS itself.

Neurochaos Learning (NL) [20] is a novel machine learning paradigm inspired by the chaotic firing patterns observed in biological neurons. Unlike conventional neural networks that rely on gradient-based optimization, NL leverages deterministic chaos — specifically, the sensitive dependence on initial conditions exhibited by chaotic maps such as the Bernoulli shift or the skew tent map — to generate rich, high-dimensional feature representations from input data. In NL, each input value is used as an initial condition for a chaotic map, and the resulting chaotic trajectory is harvested as a feature vector. These features are then fed into a lightweight classifier (e.g., cosine similarity based classifier), making NL computationally efficient while remaining highly expressive.

In the context of continual learning, NL offers a particularly attractive property: because chaotic maps are deterministic and parameter-free, the feature extraction layer does not suffer from catastrophic forgetting — a key limitation of standard neural networks when learning from sequential data streams. This makes NL well-suited for the dynamic, evolving nature of CAS, where new contexts and intents must be accommodated over time without discarding prior knowledge.

NL is also relevant to contextual analogical reasoning (CAR). The chaotic feature representations produced by NL preserve subtle distinctions between inputs, enabling fine-grained similarity comparisons between past and current execution contexts. This can enhance CAR by providing a richer similarity substrate on which to base the retrieval and adaptation of prior execution models. Together, these properties make NL a promising foundational technique for several components of the intent management lifecycle described in this paper.

3. Overview of Our Approach

Our overall method is depicted in Figure 1 below.

 doi:10.20944/preprints202605.0367.v1

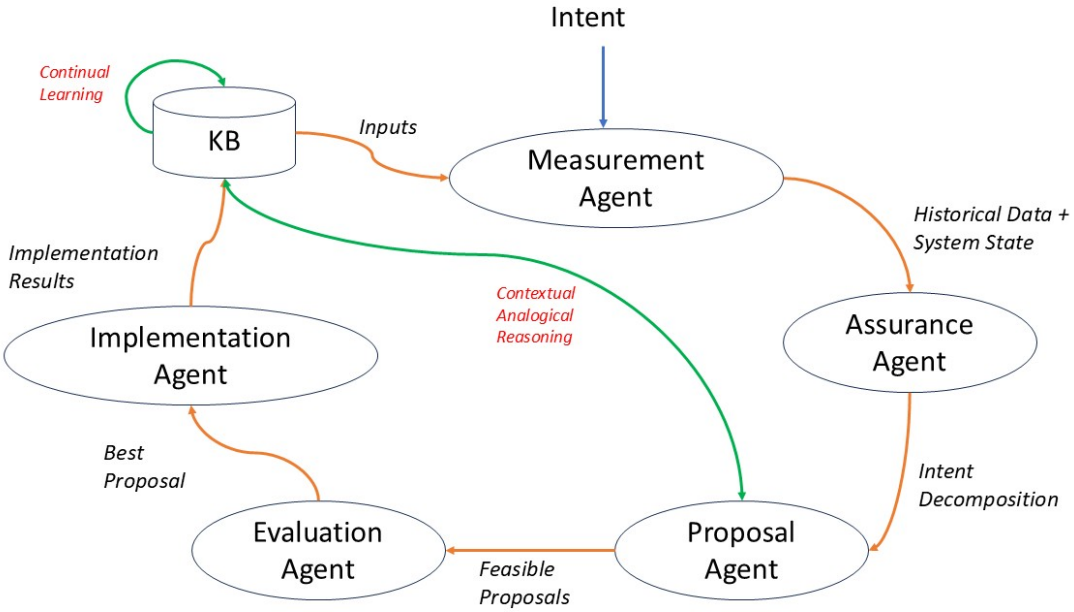

Figure 1. Our Approach.

The key models that are needed for the method of Figure 1 to function, are the following:

1. Intent model - expressed as a set of requirements and quality parameters
2. Context model - expressed as a set of $< key, value >$ pairs, where the *key* represents a context element (e.g., storage, compute, location, any kind of environmental variable external to the CAS), and *value* represents either its value or a range within which that context is relevant.
3. Execution model - expressed as a dynamic process KG [6,21], the eventual implementation of which is supposed to fulfill the intents expressed in the intent model
4. Mapping between intent model and execution model in a given context

Based on the above models, the following actions will have to be implemented as per Figure 1:

1. Determining state of the underlying system as recorded in the Knowledge Base (KB), by Measurement agent
   (a) This will be based on the inputs from the KB.
   (b) This will also contain the results of past executions and the "plan vs actual" loss measures as determined by the loss functions, which will be the result of continual learning [8,22].
   (c) This would involve implementing classification and link prediction algorithms as applicable, which is where Neurochaos Learning (NL) [20] would be useful.
2. Decomposing the intent into sub-intents for ease of execution
   (a) This will be implemented by the Assurance agent, which can employ techniques such as those described in [6,23] to decompose the intent to sub-intents, each of which can then be mapped onto execution model candidates.
3. Determining the execution model candidates by Proposal agent
   (a) Implementing reinforcement learning (RL) along with contextual analogical reasoning (CAR) [15**?**] – using the decomposed sub-intents supplied by Assurance agent.
      i. Based on this, determining the feasible set of proposals, i.e., feasible set of execution models given the context.
      ii. This will involve comparing the context of the current intent against the contexts that were recorded in previous cases in the KB, and then implementing contextual analogical reasoning (CAR) by "mixing and matching" the contexts.

 doi:10.20944/preprints202605.0367.v1

iii. Using this to determine the proposed execution models that can meet the intent - by Proposal agent.

4. Determining the "best" candidate by Evaluation agent, among those supplied by the Proposal agent
5. Implementation of the "best" candidate by Implementation agent, which also involves the following:
   (a) Calculating and reporting extent of fulfillment/violation after implementation.
   (b) Using that to implement continual learning to update the results into the KB, which can then be picked up appropriately using CAR, as depicted in Figure 1. The continual learning can be implemented using models such as Temporal Difference.
   (c) Here too, NL would be useful, in particular, to incorporate chaotic continual learning [8], continual reinforcement learning [22] and dynamic reinforcment learning [24].

Neurochaos Learning (NL) has the potential to contribute meaningfully at multiple stages of the intent management lifecycle depicted in Figure 1, beyond its role in chaotic continual learning.

At the **Measurement Agent stage**, NL can assist in classifying the current system state by mapping observed system metrics onto chaotic feature representations, which can then be compared against historical states stored in the KB. This is particularly useful when the system state space is high-dimensional and evolving, as is typical in CAS. NL-based classification can also support link prediction in the KB's knowledge graph, helping to infer missing or implicit relationships between past intents, contexts, and their outcomes.

At the **Assurance Agent** stage, NL can support intent decomposition by rapidly exploring the space of candidate sub-intent combinations. The sensitive dependence on initial conditions inherent in chaotic maps means that even small differences in intent specifications can be mapped to well-separated regions in the chaotic feature space, making it easier to distinguish between semantically distinct decomposition candidates and converge on the most suitable one.

At the **Proposal Agent** stage, NL can enhance contextual analogical reasoning (CAR) by providing chaotic feature embeddings of both the current context and historical execution contexts. Since these embeddings preserve fine-grained distinctions between contexts, NL can improve the precision of similarity-based retrieval of past execution models, leading to better-quality proposals for the Evaluation agent.

At the **Evaluation Agent** stage, NL-based regression [25] can be used to estimate the likely extent of intent fulfillment for each candidate execution model, prior to actual implementation. This provides a lightweight simulation capability that can help the Evaluation agent rank proposals without incurring the cost of full execution.

Finally, at the **Implementation Agent** stage, NL supports chaotic continual learning [8] by encoding implementation outcomes as chaotic trajectories that are stored back into the KB. Over successive intents, this enables the system to progressively refine its decomposition and mapping strategies, improving overall intent fulfillment rates across the lifecycle.

4. Illustrative Example

Our illustrative example is in the logistics network domain. A logistics network is a typical example of a CAS, since it comprises multiple interacting entities such as vehicles, operators, inventory managers, suppliers, etc., and it is highly dynamic, e.g., routes and delivery schedules would many times need to be adapted to changing conditions and circumstances.

We assume a city where goods have to be delivered from a source to several destinations. Delivering such goods would require fulfilling the intents of goods delivery in a particular context, which would involve determining the appropriate route plan with the relevant intermediate stops, taking the context into account.

Hence one example of an intent $I1$ could be: "deliver these parcels from Point A to Points B, C and D". The expectations from the intents could be: (E1) overall deadline is 1 hour, (E2) Parcel for Point B

 doi:10.20944/preprints202605.0367.v1

to be delivered within 20 minutes since that is a preferred customer, (E3) overall fuel consumption to be within 1 liter. Some contextual conditions to consider, would be: (a) partial blockage on Road #1 due to civil works, (b) slush and potholes on Road #2 due to rain, (c) traffic jam on Road #3.

For ease of implementation, the intent $I1$ can be decomposed into sub-intents $I1_1$ and $I1_2$ as follows:

- $I1_1$: Deliver to Point B first
- $I1_2$: Deliver to Points C and D after delivering to Point B

The sub-intents can then be mapped onto an execution model that models the route that the delivery executive has to take, which will fulfill the sub-intents and fulfill expectations, while taking the contextual conditions into account.

Our example is illustrated in Figure 2. As can be seen here, there are several possible routes that could fulfill the intent and its sub-intents, such as: $A - P1 - B - D - C$, $A - P4 - B - D - C$, $A - P1 - B - P4 - P5 - P6 - P7 - P8 - C - D$, and $A - P4 - B - P4 - P5 - P6 - P7 - P8 - C - D$. Each such route can be modeled as a knowledge graph (KG), with nodes being the points and edges being the roads between the points. (The points in this example could refer to either intersections of roads, or specific buildings.)

The best solution among the above possibilities could depend on the context. For example, would it be preferable to tolerate the traffic jam on Road #3 (the first of the possible routes listed above), or the slush and potholes on Road #2 (the third of the possible routes listed above)? The fulfillment of the expectations of the intent would therefore depend on making the appropriate tradeoff in order to ensure that the expectations (E1) through (E3) are met. In particular, fuel consumption would increase if the delivery vehicle had to periodically "stop and go" on roads experiencing potholes and/or traffic jams, rendering fulfilling expectation (E3) particularly difficult.

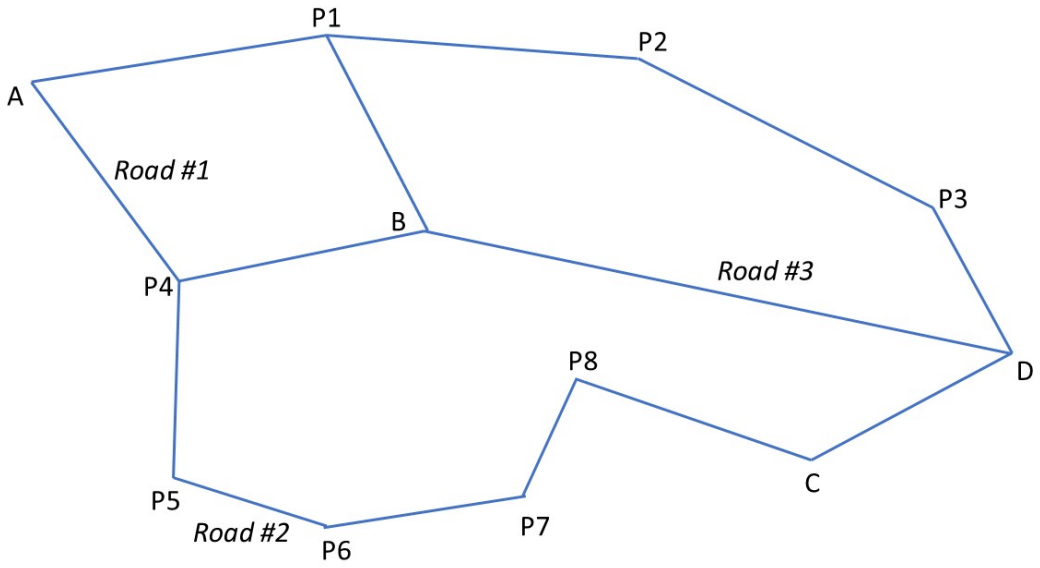

Figure 2. Illustrative Example.

This naturally raises the following questions: (1) how would intents, expectations and contexts be formally modeled; (2) how would intents be correctly decomposed into the sub-intents which collectively add up to the intent; (3) how would the execution model be modeled; (4) how would the sub-intents then be mapped onto the execution model to ensure their implementation; (5) what would be the “best” execution model that can fulfill the intent and its expectations, given the context; and (6) how would one learn from past history in order to determine the best solution the next time this intent needs to be fulfilled, given the contexts? As a corollary to question (6), another question would be: (7)

 doi:10.20944/preprints202605.0367.v1

how would one find the "best" solution if the context was changed, e.g., if there was an additional partial blockage on the road connecting $P7$ and $P8$ in Figure 2?

In the rest of this paper, we will discuss the above questions and present our proposed approach. In so doing, we will also present several key research questions that need to be addressed before intent-based management of CAS can become a practical reality.

5. Models

5.1. Intent Model

A strawman KG representing an intent model is depicted in Figure 3.

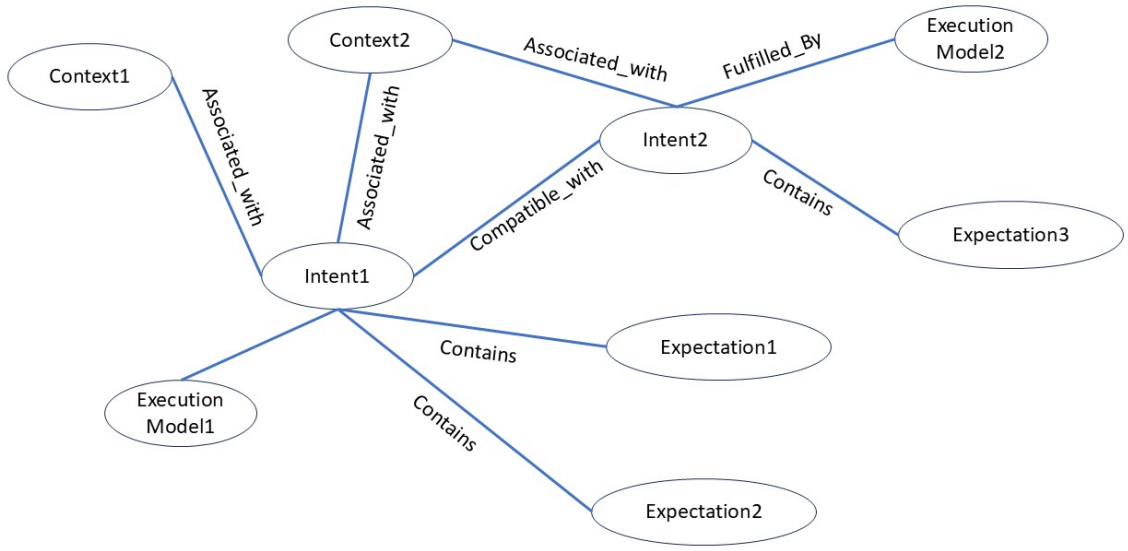

Figure 3. Intent Model.

We model an intent as a state of the system that needs to be achieved, i.e., the functional requirements that need to be met for the intent. Each intent also contains one or more *expectations*; an expectation is a non-functional requirement expressed in the form of performance metrics.

Intents can also be associated with each other via one of the following relations: *derivation* and *compatibility*. Derivation relation between intents that signifies that the other, as illustrated in Section 4, where $I1_1$ is a sub-intent of $I1$.

Compatibility relation between intents signifies whether they are compatible or not. Two intents are mutually compatible if the fulfillment of one does not conflict with the fulfillment of the other, else they are incompatible. Depending on the expectations of each intent, the extent of incompatibility, if any, can also be modeled. Intents that are neither compatible nor incompatible with each other, will not have any relation among each other.

Many times, for ease of implementation, intents need to be decomposed into sub-intents, which can then be mapped onto execution actions to fulfill the sub-intents. The key research issue therefore, is how best to decompose the intents so that the sub-intents add up to the intent, and how would "best" be defined here? One possible approach could be via *compositional reasoning* [6,26] on KGs, given that our intent model is represented as a KG. Compositional reasoning helps to break a complex problem into sub-problems, and a method for compositional reasoning for dynamic multimodal process KGs has been presented in [6]. In particular, the citation [6] shows how a given recipe can be analyzed and broken down into steps, based on specific requirements for tweaking the recipe to suit specific eating habits, e.g., to cater to vegetarians.

 doi:10.20944/preprints202605.0367.v1

In our earlier work [7] we showed how a goal can be decomposed into sub-goals until the semantics of the sub-goal can be mapped onto that of a business process task. Techniques such as this and the one presented in [6] can be investigated for our intent decomposition purpose.

Several intent decomposition approaches in the telecom domain are presented in [23], of which the graph traversal approach appears to be the most relevant for us. In that approach, each intent is subdivided into its candidate sub-intents based on past history stored in the Knowledge Base (KB). Each candidate sub-intent is then subdivided into its own candidate sub-intents, and so on, until no more decomposition is possible. A multi-partite graph is constructed, where the nodes of each partition of the graph are sub-intents at the same level, while edges are only between sub-intents at different partitions. Traversal of these edges will result in several paths, each of which would be a potential intent decomposition. The decomposition that can best meet the expectations is the one selected.

Effective intent decomposition could therefore be an integration of the techniques presented in [6,7,23].

A key challenge in intent decomposition is the combinatorial nature of the search space: as the number of sub-intents grows, the number of possible decompositions grows exponentially, making exhaustive search impractical. NL offers a promising approach to navigating this search space efficiently.

Specifically, the chaotic feature representations generated by NL can be used to embed each candidate decomposition into a high-dimensional feature space. Because chaotic maps are ergodic — meaning they eventually visit all regions of their state space — NL-based search can achieve broad coverage of the decomposition space without requiring an exhaustive enumeration of all candidates. This is analogous to how chaotic dynamical systems explore their phase space more efficiently than random walks, a property that has been leveraged in optimization contexts.

Furthermore, NL can be combined with the graph traversal approach from [23] to prioritize which branches of the multi-partite decomposition graph to explore first. By assigning chaotic feature scores to candidate sub-intents based on their similarity to historically successful decompositions stored in the KB, the search can be guided toward promising regions of the decomposition space early, reducing the number of iterations needed to arrive at a high-quality decomposition.

In this way, NL complements compositional reasoning [6] by acting as a fast, lightweight heuristic for pruning the decomposition search space, while the graph traversal and goal-driven techniques handle the structural correctness of the decomposition. Together, they provide a more efficient and robust intent decomposition mechanism suited to the dynamic demands of CAS.

5.2. Context Model

We define as a *context* as a part of the overall state of the CAS which is relevant to at least one intent [27]. Examples of contexts are those listed in Section 4 pertaining to the roads Road #1 through #3. Contexts can be represented as boolean conditions, e.g., "Traffic jam on Road #3 = TRUE". More specific contexts can also be defined, e.g., "20 minute average delay on Road #3".

An intent can have multiple contexts associated with it, as long as they are consistent. It is also possible for an intent to be context-free; for example, if the Roads #1 through #3 did not suffer from the issues listed above in Section 4, the intent $I1$ would be context-free.

Contexts are not only useful in describing the state of the CAS, but in determining the mapping of intents to execution models, since the mapping will differ based on the contexts. For example, the route plans described in Section 4 are based on the three contexts mentioned therein; if the context related to Road #1 were removed, the route $A - P4 - B - P4 - P5 - P6 - P7 - P8 - C - D$ could be the best solution, otherwise $A - P1 - B - P4 - P5 - P6 - P7 - P8 - C - D$ could be the best solution.

5.3. Execution Model

Based on the above discussion, we represent our execution model as a dynamic process KG (DPKG), where each node is an activity that is implemented, and edge between nodes represent

 doi:10.20944/preprints202605.0367.v1

handoffs between activities. As a DPKG, it can also contain so-called "back edges", i.e., edges that represent loops in the process. One such example of a back edge is $B - P4$ in the process $A - P4 - B - P4 - P5 - P6 - P7 - P8 - C - D$. For our illustrative example of Figure 2, the DPKG that mirrors the possible route plans, is depicted in Figure 4.

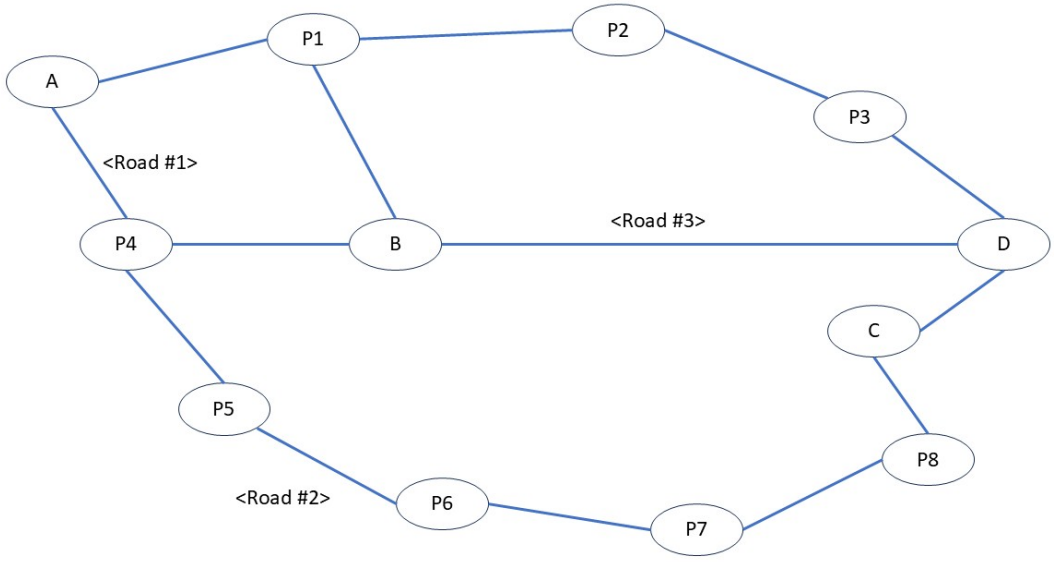

Figure 4. Execution Model.

Hence in a DPKG representing a route plan, the nodes are intersections or specific points in the route such as designated houses or offices. And the edges are routes or roads between the nodes. Hence leveraging from [28], node attributes could be address; GPS location; specific characteristics of the node such as intersecting roads, details of houses/offices at the node, etc. And edge attributes could be length and width of the road; characteristics of the road such as one-way or two-way, nature of road (whether residential or commercial, i.e., whether it admits larger vehicles such as buses or trucks), etc. Suitable embeddings of nodes and edges can be generated to help implement mapping of the intent KG to this DPKG.

6. Mapping Between Intent and Execution Models

In order for the decomposed sub-intents to be fulfilled, they need to be mapped onto an execution model that can fulfill them. This would require the following tasks: (a) mapping each sub-intent onto a part of the execution model, i.e., a sub-graph of the overall DPKG that models the execution model; (b) integrating the sub-graphs to produce the overall DPKG, while verifying that the overall DPKG does meet all the sub-intents.

In our earlier work [7] we presented a method for goal-driven business process derivation. The method keeps deriving sub-goals from parent goals and building a goal tree, until the leaf goals of the tree can be semantically mapped onto actual business process tasks. A similar approach for translating intents into policies and then finite state machines to execute the policies, has also been presented in [29].

However, for better accuracy, it would be beneficial to integrate contextual analogical reasoning (CAR) [15**?**] into this intent $\rightarrow$ mapping. The key idea of CAR is to determine the most "similar" mapping from those stored in the KBs, and then to modify it that mapping to suit the currently specified intent.

 doi:10.20944/preprints202605.0367.v1

7. Intent Management Lifecycle

In this Section, we will describe the intent management lifecycle as depicted in Figure 1, with emphasis on the various agents that implement the lifecycle. In so doing, we will also describe the functioning of the Knowledge Base (KB), which stores the various models and enables continual learning.

7.1. Measurement Agent

The main input to the Measurement agent will be the intents specified by the user. The role of the Measurement agent will be the following:

- Analyze the intents.
- Correlate them against the historical data in the KB, in particular, the prior records of intent fulfillment (or the lack thereof) of similar intents.
- Send the results of this correlation to the Assurance agent.

Referring to our illustrative example from Section 4, the inputs to the Measurement agent would be the intent $I1$, the expectations (E1) through (E3) and the contextual conditions (a) through (c). The outputs of the Measurement agent will therefore be the following, and would be derived from the continual learning implemented in the KB:

- Results of implementation of the intent $I1$ in the past, with same or different expectations and/or contexts. The result will be in the form of a *loss function* that measures the difference between the proposed execution model by the Evaluation agent, and the actual performance as reported into the KB. Loss functions can be modeled using measures such as the Kullback-Leibler (KL) divergence or mean squared error loss [30].
- Similarity between the past implementation scenarios and the current scenario. This can be a combination of similarity metrics [31] such as Euclidean distance, Minkowski distance, etc. And these will be of several types, i.e., among intents, among expectations and among contexts.

For example, referring to the example from Section 4, a past similar example could be a similar but different intent $I1'$, which would be about delivery to only points C and D, but with a time expectation of 30 minutes. And the loss function would measure a delay of 4 minutes over the 30 minutes over several instances of the route plan developed to satisfy intent $I1'$, signifying an average delay of 4 minutes. This information is then fed to the Assurance agent, whose task is to develop the appropriate decomposition of intent $I1$, taking into account the past history.

7.2. Assurance Agent

The role of the Assurance agent is to implement intent decomposition based on the information provided by the Measurement agent, viz., intents correlated against historical data.

Hence extending the illustration given above in Section 7.1, let us assume that $I1'$ had been decomposed into the following sub-intents:

- $I1'_1$: deliver to point C first
- $I1'_2$: then deliver to point D

If we were to suitably adapt the graph traversal technique from [23], and incorporating the prior learning from the KB as distilled and provided by the Measurement agent, the Assurance agent could arrive at the intent decomposition as described in Section 4. It could also arrive at a slightly different decomposition where point D would have to be serviced after point B but before point C.

7.3. Proposal Agent

The decomposed sub-intents, along with the historical information gleaned by the Measurement agent, are then fed to the Proposal agent, whose task is to develop the intent $\rightarrow$ execution model mapping. This can be done via contextual analogical reasoning [15**?**] to determine the best mapping based on the combination of the sub-intents and historical data.

 doi:10.20944/preprints202605.0367.v1

Referring to our illustrative example from Section 4, it can be argued that the execution model that implemented delivery to D before C would result in a delay, and therefore a non-zero loss function, hence the candidate route plans presented in Section 4 would be better options that can be presented to the Evaluation agent.

To augment the Proposal Agent's ability to determine candidate execution models, an NL-based approach can be incorporated that combines chaotic feature representations with a machine learning classifier. Specifically, each candidate execution model can be represented as a feature vector derived from its constituent sub-intents and associated context, and a chaotic map — such as the skew tent map or the Bernoulli shift map — can be applied to these features to generate a rich, high-dimensional chaotic representation. A lightweight classifier trained on these chaotic representations can then rank the candidate execution models by their likelihood of fulfilling the decomposed sub-intents, based on historical data stored in the KB.

Beyond classification, a chaos-based variant of reinforcement learning (RL) can also be incorporated into the Proposal Agent. Standard RL approaches explore the action space — in this case, the space of candidate execution models — using stochastic policies, which can be slow to converge in large, high-dimensional spaces typical of CAS. By replacing the stochastic exploration mechanism with a chaotic one, drawing on work such as [32], the Proposal Agent can achieve more systematic and efficient exploration of the candidate execution model space. Chaotic exploration has been shown to offer better coverage of the search space compared to purely random exploration, while remaining deterministic and therefore reproducible — a desirable property in intent management systems where auditability is important.

The combination of NL-based classification and chaos-based RL therefore provides the Proposal Agent with two complementary capabilities: a fast, history-informed ranking of candidate execution models via NL classification; and a principled, efficient exploration of novel execution model candidates via chaos-based RL. Together, these can improve both the quality and the diversity of the proposals forwarded to the Evaluation Agent, ultimately enhancing the overall intent fulfillment rate of the CAS.

7.4. Evaluation Agent

The Evaluation agent receives the various candidate execution models and then evaluates which of them would be the best. The evaluation of "best" would therefore be done via an aggregation-based approach that would model the various steps of the execution model as edges in the DPKG representing the execution model.

Each such edge is analyzed to determine the sub-intent that it would fulfill, and the extent of fulfillment is then determined, based on the combination of the following: the properties of the edge as defined by its features; and the past history as represented in the historical data gleaned from the KB by the Measurement agent. The overall fulfillment is then determined as a combination of those of the individual edges, for each execution model.

The execution model that produces the lowest possible loss function is then selected by the Evaluation agent for implementation.

7.5. Implementation Agent

The Implementation agent implements the "best" execution model provided by the Evaluation agent. It then reports the results of execution into the KB, which include the following:

- Execution details of each edge of the DPKG that is exercised as part of intent fulfillment. These will be modeled as part of the features of the edges, which will be reported into the KB.
- Extent of intent fulfillment (or the lack thereof) for each edge vis-a-vis the sub-intent that it has attempted to fulfill.

7.6. Knowledge Base (KB)

The KB is the datastore for the CAS, which stores the following:

 doi:10.20944/preprints202605.0367.v1

- Intent models as KGs, representing intents, their expectations, and their associated contexts.
- Execution models represented as KGs, with mappings to the appropriate intents and contexts, i.e., those intents fulfilled by the execution model in question, and the context in which the intent is fulfilled. These mappings can be represented as hypergraphs as presented in [33].
  - These mappings will also contain the results of the contextual analogical reasoning that resulted in the mappings. These will be represented as features of the edges that model the mappings.
- Implementation results stored along with each execution model, which provide details of the execution, along with the loss functions that model the extent of fulfillment (or lack thereof), linked to the contexts. These will be useful during continual learning [8]. In addition, techniques such as deep reinforcement learning for dynamic KG reasoning [34] and multi-hop KG reasoning with reinforcement learning [35], can also be considered for incorporation into continual learning.

The above data in the KB is what is needed for the Measurement agent in Section 7.1 to determine the appropriate historical data for the user-specified intent, as well as the system state information needed for the Assurance agent to implement intent decomposition.

8. Key Research Questions

The above description therefore raises several key research questions.

System State Determination: this is related to determining the system state by the Measurement agent and the KB.

- **RQ1**: How to represent the collected data in the KB so that it is both semantically meaningful, and also amenable to accurate retrieval? In this paper, we discussed a possible approach using a hypergraph representation [33], but that would still need to be tailored and possible enhanced, to accommodate intent, context and execution models.
- **RQ2**: Recent work in data management has centered on the concept of *data spaces* [36], which are shared logical partitions in databases to which selected access to data in prespecified formats is provided. This also includes a proposed reference architecture model. How can this be suitably tailored to develop a suitable model for our KB?

Intent Models and Intent Decomposition: this is related to intent modeling and decomposition by the Assurance agent.

- **RQ3**: how to represent intent models as KGs? Initial work on KG-based representation of intents in the telecom domain has been reported in [14,37], but this needs to be generalized so that it can apply to all domains. In particular, a more generic schema for intent models needs to be developed.
- **RQ4**: how to effectively decompose the intent into sub-intents? In the telecom domain [18], intent-driven decomposition assumes the existence of intent management functions (IMF) at lower levels that propose how best to perform this decomposition. But how would this be done in the absence of these IMFs? In this paper, we have highlighted the graph traversal approach from [23], but that would need to be augmented with techniques from goal-driven requirements engineering [38]. Here too, a domain-agnostic approach would be needed.
- **RQ5**: Given **RQ2**, how to represent sub-intents as derived KGs from the original KG that represent the parent intents, so that this can be used to facilitate intent-driven decomposition?

Intent Model $\rightarrow$ Execution Model Mapping by Proposal agent:

- **RQ6**: What would be the best method to implement contextual analogical reasoning for intent $\rightarrow$ execution model mapping? Of particular interest would be the determination of a suitable

 doi:10.20944/preprints202605.0367.v1

similarity metric which can adequately calculate similarities between past executions and the current one. What would complicate this calculation, is that this metric would have to take into account the differing contexts of the executions.

- **RQ7**: How would NL [20] help in determining the optimal mapping? That is, what would be the best NL-based technique to determine the optimal mapping based on past data? This could incorporate the aforementioned similarity metric, but it would be more of an empirical deep learning based technique to develop the optimal mapping.

Evaluation of proposals by Evaluation agent:

- **RQ8**: How to simulate the determination of extent of intent fulfillment by the candidate execution model so that its efficacy can be calculated? Would regression modeling based on NL [25] help here, and if so, how?
- **RQ9**: How would one automatically generate variants of execution models that could fulfill the intents, even partially? This would help generate several alternative partial solutions, when complete solutions are not possible, since there may be instances when it would not be possible to generate a perfect solution. Techniques such as generative AI, perhaps using Large Language Models (LLMs) [39], would need to be explored for this.
- **RQ10**: How to evaluate which of the execution models is the "best" one? Considering that any execution model has to satisfy multiple intents, this translates into a multi-criteria decision making problem. Hence the suitable machine learning techniques to solve such a problem [40] need to be developed.
- **RQ11**: As a corollary of **RQ10** above, how would imperfect execution models, i.e., those that only partially satisfy the intents, be evaluated to determine which would be the "best" one, in the absence of a perfect solution?

Implementation results reporting by Implementation agent, and continual learning:

- **RQ12**: How can chaotic continual learning [8] be extended to do the following: record the implementation results generated by the Implementation agent in the form of loss functions; and feed back into the chaotic continual learning approach to help generate better solutions to future user-specified intents?
- **RQ13**: How can active inference [41,42], a technique derived from causal inference that mimics the brain's ability to act in any situation so as to continuously reduce the deviation of planned solutions versus the reality, be incorporated to enhancing chaotic continual learning [8]?

9. Conclusions and Future Work

In this position paper, we have introduced our concept of Complex Adaptive System (CAS), which is a system of interacting parts whose behavior can change continuously. We have shown how such systems can be run and managed in an intent-driven manner, via an intent management lifecycle. We have also proposed specific agents for each phase of the lifeycle, along with methods by which the intents can be decomposed into sub-intents for convenience of implementation, and can subsequently be mapped onto execution models as per the context of the intents. In so doing, we have analyzed the problem space of intent-driven management of CAS, and we have posed several key research questions that need to be addressed if intent-driven management of CAS is to become a reality.

Our future work will involve addressing each of the research questions, and presenting and demonstrating solutions to solve them.

Conflicts of Interest: The authors declare no conflicts of interest.

 doi:10.20944/preprints202605.0367.v1

References

1. Carmichael, T.; Hadžikadić, M. The fundamentals of complex adaptive systems. In *Complex adaptive systems: Views from the physical, natural, and social sciences*; Springer, 2019; pp. 1–16.
2. Szilágyi, P. I2BN: Intelligent Intent Based Networks. *Journal of ICT Standardization* **2021**, *9*, 159–200. https://doi.org/10.13052/jicts2245-800X.926.
3. Niemöller, J.; Silvander, J.; Stjernholm, P.; Angelin, L.; Eriksson, U. Autonomous Networks with Multi-Layer, Intent-Based Operation. *Ericsson Technology Review* **2023**, *2023*, 2–13.
4. Sabour, S.; Ebrahimzadeh, A.; Soualhia, M.; Wuhib, F.; Glitho, R.H. Intent-Based Service Graph Selection for Cost-Effective Cloud Deployment. In Proceedings of the 2025 28th Conference on Innovation in Clouds, Internet and Networks (ICIN). IEEE, 2025, pp. 140–147.
5. Bensalem, M.; Dizdarević, J.; Carpio, F.; Jukan, A. The role of intent-based networking in ict supply chains. In Proceedings of the 2021 IEEE 22nd international conference on high performance switching and routing (HPSR). IEEE, 2021, pp. 1–6.
6. Venkataramanan, R.; Shyalika, C.; Sheth, A.P. Dynamic Multimodal Process Knowledge Graphs: A Neurosymbolic Framework for Compositional Reasoning. *IEEE Internet Computing* **2025**, *29*, 86–92.
7. Ghose, A.K.; Narendra, N.C.; Ponnalagu, K.; Panda, A.; Gohad, A. Goal-driven business process derivation. In Proceedings of the International Conference on Service-Oriented Computing. Springer, 2011, pp. 467–476.
8. Laleh, T.; Faramarzi, M.; Rish, I.; Chandar, S. Chaotic continual learning. In Proceedings of the 4th Lifelong Machine Learning Workshop at ICML 2020, 2020.
9. Ehrlinger, L.; Wöß, W. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)* **2016**, *48*, 2.
10. Guarino, N.; Oberle, D.; Staab, S. What is an ontology? *Handbook on ontologies* **2009**, pp. 1–17.
11. Hogan, A.; Blomqvist, E.; Cochez, M.; d'Amato, C.; Melo, G.D.; Gutierrez, C.; Kirrane, S.; Gayo, J.E.L.; Navigli, R.; Neumaier, S.; et al. Knowledge graphs. *ACM Computing Surveys (Csur)* **2021**, *54*, 1–37.
12. Tiddi, I.; Schlobach, S. Knowledge graphs as tools for explainable machine learning: A survey. *Artificial Intelligence* **2022**, *302*, 103627.
13. Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; Yu, P.S. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* **2021**, *33*, 494–514.
14. Mehmood, K.; Kralevska, K.; Palma, D. Intent-driven autonomous network and service management in future cellular networks: A structured literature review. *Computer Networks* **2023**, *220*, 109477.
15. Walliser, B.; Zwirn, D.; Zwirn, H. Analogical reasoning as an inference scheme. *Dialogue: Canadian Philosophical Review/Revue canadienne de philosophie* **2022**, *61*, 203–223.
16. Cui, W.; Zhang, L. Modeling knowledge graphs with composite reasoning. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2024, Vol. 38, pp. 8338–8345.
17. Niu, G.; Zhang, Y.; Li, B.; Cui, P.; Liu, S.; Li, J.; Zhang, X. Rule-guided compositional representation learning on knowledge graphs. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2020, Vol. 34, pp. 2950–2958.
18. Narendra, N.C.; Kanthaliya, R.; Akumalla, V. Intent-based Meta-Scheduling in Programmable Networks: A Research Agenda. *arXiv preprint arXiv:2412.04232* **2024**.
19. Liu, L.; Du, B.; Fung, Y.R.; Ji, H.; Xu, J.; Tong, H. Kompare: A knowledge graph comparative reasoning system. In Proceedings of the Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, 2021, pp. 3308–3318.
20. Harikrishnan, N.; Nagaraj, N. Neurochaos inspired hybrid machine learning architecture for classification. In Proceedings of the 2020 International Conference on Signal Processing and Communications (SPCOM). IEEE, 2020, pp. 1–5.
21. Garimella, R.; Yip, H.Y.; Venkataramanan, R.; Sheth, A.P. Building Multimodal Knowledge Graphs: Automation for Enterprise Integration. *IEEE Internet Computing* **2025**, *29*, 76–84.
22. Pan, C.; Yang, X.; Li, Y.; Wei, W.; Li, T.; An, B.; Liang, J. A Survey of Continual Reinforcement Learning. *arXiv preprint arXiv:2506.21872* **2025**.
23. Kattepur, A.; Das, S.; Daroui, D.; Mohalik, S.; Orlic, M.; Ertas, S. DETROIT: Decomposition techniques for a hierarchy of 6G network intent management functions. *Computer Networks* **2025**, p. 111657.
24. Khandelwal, V.; Yip, H.Y.; Sheth, A. Toward Neurosymbolic Reinforcement Learning via Editable Specifications. *Association for the Advancement of Artificial Intelligence* **2026**.
25. Henry, A.; Nagaraj, N. Augmented regression models using neurochaos learning. *Chaos, Solitons & Fractals* **2025**, *201*, 117213.

 doi:10.20944/preprints202605.0367.v1

26. Sinha, S.; Premsri, T.; Kordjamshidi, P. A survey on compositional learning of AI models: Theoretical and experimental practices. *arXiv preprint arXiv:2406.08787* **2024**.
27. Narendra, N.C.; Deb, N.; Das, S. Dynamic Contextual Goal Management in IoT-Based Systems. *IEEE Internet of Things Journal* **2020**, *7*, 10708–10718.
28. Tan, J.; Qiu, Q.; Guo, W.; Li, T. Research on the construction of a knowledge graph and knowledge reasoning model in the field of urban traffic. *Sustainability* **2021**, *13*, 3191.
29. Dzeparoska, K.; Tizghadam, A.; Leon-Garcia, A. Emergence: An intent fulfillment system. *IEEE Communications Magazine* **2024**, *62*, 36–41.
30. Kim, T.; Oh, J.; Kim, N.; Cho, S.; Yun, S.Y. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. *arXiv preprint arXiv:2105.08919* **2021**.
31. Ali, S.K.; Aydam, Z.M.; Rashed, B.M. Similarity metrics for classification: A Review. In Proceedings of the IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2020, Vol. 928, p. 032052.
32. Perepu, S.K.; Martins, J.P.; Dey, K.; et al. Multi-agent reinforcement learning for intent-based service assurance in cellular networks. *arXiv preprint arXiv:2208.03740* **2022**.
33. Masmoudi, M.; Ben Abdallah Ben Lamine, S.; Karray, M.H.; Archimede, B.; Baazaoui Zghal, H. Semantic data integration and querying: a survey and challenges. *ACM Computing Surveys* **2024**, *56*, 1–35.
34. Liu, H.; Zhou, S.; Chen, C.; Gao, T.; Xu, J.; Shu, M. Dynamic knowledge graph reasoning based on deep reinforcement learning. *Knowledge-based systems* **2022**, *241*, 108235.
35. Zhu, A.; Ouyang, D.; Liang, S.; Shao, J. Step by step: A hierarchical framework for multi-hop knowledge graph reasoning with reinforcement learning. *Knowledge-Based Systems* **2022**, *248*, 108843.
36. Bacco, M.; Kocian, A.; Chessa, S.; Crivello, A.; Barsocchi, P. What are data spaces? Systematic survey and future outlook. *Data in Brief* **2024**, *57*, 110969.
37. Mehmood, K.; Kralevska, K.; Palma, D. Knowledge-based Intent Modeling for Next Generation Cellular Networks. In Proceedings of the 2023 IEEE International Mediterranean Conference on Communications and Networking (MeditCom). IEEE, 2023, pp. 181–186.
38. Green, S. Goal-driven approaches to requirements engineering. *Imperial College, University of London, London, Technical Report* **1994**.
39. Dzeparoska, K.; Lin, J.; Tizghadam, A.; Leon-Garcia, A. LLM-based policy generation for intent-based management of applications. In Proceedings of the 2023 19th International Conference on Network and Service Management (CNSM). IEEE, 2023, pp. 1–7.
40. Liao, H.; He, Y.; Wu, X.; Wu, Z.; Bausys, R. Reimagining multi-criterion decision making by data-driven methods based on machine learning: A literature review. *Information Fusion* **2023**, *100*, 101970.
41. Prakki, R. Demonstrating the Continual Learning Capabilities and Practical Application of Discrete-Time Active Inference. *arXiv preprint arXiv:2410.00240* **2024**.
42. Friston, K.; FitzGerald, T.; Rigoli, F.; Schwartenbeck, P.; Pezzulo, G.; et al. Active inference and learning. *Neuroscience & Biobehavioral Reviews* **2016**, *68*, 862–879.