IMA journal title (to be updated during typesetting) (2021) **00**, 1–23
https://doi.org/DOI HERE

An End-To-End Match-Build for Model-Simulate-Compute

SRIKAR BALASUBRAMANIAN

[Received on 24th July 2024]

Keywords: round; match; agent; agent system; strategy; outcome; co-independent; incorporation; card

1. Abstract

There is a lot of ongoing research in the area of game theory applicable towards temporal logic, artificial intelligence, fuzzy logic, information theory, and other areas, aimed at mapping out real-world applications of data. However, although the introduction of new ideas in these fields present emerging discoveries to be looked at, there has not been a focus on applications of such ideas in an simulator-based environment to showcase the growth potential of such methodologies proposed. By mapping out some sort of idea, flow, process, method, or any sort of philosophy in a simulation-based environment given system specifications, it then raises the possibility towards solving current or new real-time problems that occur, delve deeper into the growth potential of a field, or strengthen the current understanding of a particular topic or area given its progression.

2. Introduction

This paper will discuss a practical, yet comprehensive in depth approach towards maximizing a grounded, learning curve for a match-build.

A match is defined by the following set of steps: 1. Defining the Model Architecture. 2. Mapping out the Simulation Conditions. 3. Compute processing in batches or in full, depending on applicable format assignment.

During the ongoing simulation of the match, agents will be present in a round-by-round structure that is discussed in the later sections of the paper.

An agent, in this case, is an operational mechanism with extensive capabilities of learning data given and making predictions based upon that data to operate independently or concurrently in line with other agents that work to meet a strategical approach for the main intentions of maximizing the rewards obtained for a group of agents or for agents themselves.

A strategy can be defined as making a set number of appropriate moves that best meet the requirements towards improving the current positional states of the agent-system depending on the conditions defined to simulate the match in its entirety. To figure out a strategy on a round-by-round basis, agents can either work together in alignment to design a unified strategy or use learning data given to formulate a strategy of their own, leading to co-independent strategies designed for success in that particular round.

The paper will provide an example of a match played out to a final outcome, containing information per each round that can be extracted, analyzed, and optimized for future match builds.

An outcome is an output outlined by specific criteria, and evaluations of outcomes can be done per round and finalized for completion of a match.

A structure useful for reference in the latter parts of this paper will be a round file hierarchy shown below.

round_(roundNo)_(teamLabel)_compute
round_(roundNo)_(teamLabel)_selection_cast
round_(roundNo)_(teamLabel)_excess
round_(roundNo)_(teamLabel)_remaining

The above denotes a simplified, double agent-system structure showing the main components used for simulation of a round. An agent-system can consist of an agent or a multiple number of agents. The rest of the paper will assume a double agent-system structure acting concurrently on in parallel for many of the match-build examples. However, extensions can be made to many other types of agent-systems where possible forms of incorporation can be made between both agent-systems set to take part in a match. We only note the scaling of agent-systems in terms of the number of agents present, but the structure that remains the same is the concept of one system opposing one another, unless alliances are formed. This will be the model architecture followed for the rest of this paper, but requirements can be written differently if needed to look at new match-builds.

3. Model

A model is a representation of a knowledge base as a template for the simulation. It stands out as a fundamental component in a match-build for that knowledge to become exposed in a simulation. Depending on the requirements defined for model creation, it allows for understanding a model's growth potential and the end-goal it holds.

3.0.1. Components

The underlying pieces for construction of the model are the components. Any number of components built in place are to operate as gears (connected or unconnected) to begin the pipeline for binding. The term components stands out as a way for classification of parts, materials, and any form of functionality that presents a useful opportunity towards the construction of the model.

3.0.2. Bind

A binding is established to bind together each component. The process of the binding is stated by system specifications for purposes of deploying pipelines to piece together a convergence of components, leading to a justifiable model for improved performance once deployed into the arena. Alterations can be made to a model to improve the type of model deployed and to involve several opportunities for conducting experiments to examine a model's potential in depth.

4. Simulate

A model binded by components can be simulated in an environment defined by system specifications. Depending on environmental conditions imposed, it will impact the simulation process

of the model. The process for the simulation happens on a round-by-round structure basis. A round can be noted as a slice, divider, or a sort of incremental measure that stands out as an iteration for each round conducted in the match.

5. Compute

A round defined by Agent System 1 X Agent System 2, after simulation, will contain some form of metrics in relation to the performance of both agent systems during that simulation. Below represents the fundamental structures that compose a round simulated in a match. Note that other sub-structures can also be added as a layer within the fundamental structures that hold during the match process. These fundamental pillars are significant towards analysis of matches of similar variants and for match-builds that follow a different type of model, simulation, and compute than the one proposed above.

5.0.1. Backtrack

For Agent System 1 X Agent System 2, back-tracking is useful to retrace each agent's move from start to finish in that particular round to understand the knowledge learned by each of the agents. Looking at the success or failure of a strategy executed during that round can help identify patterns for minimizing errors or maximizing success for a strategy during future round simulations.

5.0.2. Description

The description of a round is important to establish a capture of information to write out not only the events taken place during the round, but also information about external metrics used to analyze these events.

5.0.3. DFA

This is a DFA, a deterministic finite automaton, part of computational theory, and is used in this round as an external metric to record two aspects: a state and a transition label. Deterministic finite automatons will have only one transition label that leads to the same state, and at least one transition label that leads to separate states. The states in this case, could be round-by-round iterations where the transition label is defined with a success or failure label to highlight two separate pathways for future round simulations. Determining the label type will require an algorithm evaluator for processing of each agent in the system. Depending on the use case of the DFAs, it serves as an important external metric for creating a diagram to look at the moves of a particular agent-system for each round simulated.

5.0.4. Documentation

The documentation during the round is an illustrative guide to establish a form of understanding the external and internal metrics used during the round simulation. By creating sections to reference these

metrics, it serves an important purpose towards looking at future match-builds carried out by having a strong foundation with the fundamentals.

5.0.5. Duration

The duration of a round is a measure of the time between Agent System 1 X Agent System 2 before the next round simulation. This external metric serves as an important evaluator to look into how long it takes to execute a strategy by each agent system.

5.0.6. Dynamic Grammar

A grammar is defined by a set of rules that will be important in the setup to describe the language used in communication between each of the agents and can also serve an important role towards the writing of the round description by using any relevant keywords from the language. In this case, the grammar is dynamic, meaning that depending on the data learned by the agents, the grammar will change mainly for satisfying some end-goal defined by the requirements for the agent system. This is taking into account that the grammar defined originally is fixed until it turns dynamic if an end-goal is stated.

5.0.7. Dynamic Language

A language is an identifier that puts the formulations in practice depending upon the rules defined by the grammar. Ongoing-communication between each of the agents during the round shows a demonstration of the language. In this case, the language is dynamic, meaning that different outputs defined by the grammar exist. Note that, a language can be dynamic with a fixed grammar if it leads to the generation of greater than 1 output.

5.0.8. Engagement

The engagement is a measure of the count of spectators present during the simulation of the round. For each round, it's possible to see a drop or rise in the number of spectators if they decide to stay, leave, or join back to view the match. The round duration serves as an important factor to count the number of spectators only after the round is completed. This is necessary to take into account specific spectators who decide to join back into the match after leaving.

5.0.9. Executable

The executable is a way to replay a round completed by Agent System 1 X Agent System 2. Although this will not show any information about metrics collected, it serves as a potential evaluator for establishing different variations of metrics for analysis of the round and as a access point for visual reference in the future.

5.0.10. Fixed Grammar

A grammar is defined by a set of rules that will be important in the setup to describe the language used in communication between each of the agents and can also serve an important role towards the writing of the round description by using any relevant keywords from the language. In this case, the grammar is fixed, meaning that the rules assigned will hold for any input passed.

5.0.11. Fixed Language

A language is an identifier that puts the formulations in practice depending upon the rules defined by the grammar. Ongoing-communication between each of the agents during the round shows a demonstration of the language. In this case, the language is fixed, meaning that only one output defined by the grammar exists.

5.0.12. Garbage Collector

A garbage collector plays an important role for any unprocessed data by the agents for the purposes of storage. Data of this type can include input entering the system or data unused during the simulation of the round. It can also store processed data, but data of this type will no longer be available in the system. In general, during the simulation of the round, data no longer needed will be sent to a garbage collector for further processing. Unless a specific use-case is defined to use the garbage collector differently, the above description will hold as the default use-case.

5.0.13. Generation

A generation of a round relates to an internal process that occurs for placing each participant of an agent system into their corresponding positions. This process is important for each round to ensure clarity over the entities involved in the simulation of a round and for any pre-processing or post-processing of data that needs to occur in between rounds.

5.0.14. Hierarchy

A round hierarchy is a representation to illustrate an ordering of external and internal metrics during the simulation of a round. Ranked by highest priority, the round hierarchy demonstrates the value that such metrics propose during the simulation of a round.

5.0.15. Incorporation Concept

The incorporation concept is the ideal that each agent system is working towards, though both systems are in direct opposition to one another unless some levels of incorporation are defined. The

ideal, defined in this sense is based by the general strategy set out by each agent system. The two pieces can then be constructed to coin a term for the type of match conducted.

5.0.16. Induce

The induce is a traceable path illustrating a set of actions by each agent-system during the simulation of a round. These moves are traced individually from one agent to another within their respective agent system. A traceable path for the agent-system holds knowledge to be transferred to another entity responsible for storage of the data.

5.0.17. Interpretation List

The interpretation list is a static holder to store path traces for each round conducted during the simulation. The path traces illustrate the time-step and the annotations applied to a given agent in the agent system. The PyReason software is used here as a tool for temporal logic tracing.

5.0.18. Interpreter

The interpreter for each round simulation reads in information about knowledge learned by each agent system. This will rinse and repeat until the round end number either specified by the system, user, or an output specification goal stated by system requirements is reached.

5.0.19. Jargon

The jargon relates to the language defined for the round. In this case, specific keywords, clauses, phrases, and sentences can serve as a template guide for the round description. This holds significance towards understanding the communication mechanisms used between agents during the simulation of a round. The documentation for the round will also be a useful reference point for the jargon learned. In general, though, defining the jargon takes into consideration contributing factors such as the context locations, the agents, and the agent system.

5.0.20. Logger

The logger for a round will log appropriate information such as the timestamp, the duration, the round number, and any other permissible data requested to validate round completion. This piece of functionality serves as a system verifier for each subsequent round completed until the round end number. These mechanisms can also be looked at a later time for any modifications to be made for how a round is validated, the process for logging a direct entry, making operational changes to the log system, or looking into other loggers that suit system requirements.

5.0.21. Merge Finalizer

The round merge finalizer takes into account the base components for simulation of a round. These are: the compute, the selection cast, the excess, and the remaining for each agent system. There is only a primary focus on these base components in particular to capture the main information that occurs during the simulation of a round.

5.0.22. NFA

This is an NFA, a non-deterministic finite automaton, part of computational theory, and is used in this round as an external metric to record two aspects: a state and a transition label. Non-Deterministic finite automatons will have at least one transition label that leads to the same state or to separate states. The states in this case, could be round-by-round iterations where the transition label is defined with a success or failure label for an agent move to highlight two separate pathways for future round simulations. Determining the label type will require an algorithm evaluator for processing of each agent in the system. Depending on the use case of the NFAs, it serves as an important external metric for creating a diagram to look at the moves of a particular agent-system for each round simulated.

5.0.23. Parser

The parser for a round involves the following structures: action, input-buffer, lexical analyzer, and a token. An action is a move operated by an agent within the agent system. The input-buffer is responsible for the processing of knowledge encoded and transferred by each agent. The knowledge in this case will be redirected to a lexical analyzer for further processing to set the token types of the knowledge contained within the buffer itself. A token is defined as a high-level annotation to define knowledge for the purpose of establishing semantics. Putting together all these components leads to a parser responsible for the ordering of knowledge representation for each agent-system. Any mismatches in the parser run will raise an error due to no match found for that particular block of knowledge.

5.0.24. Permutations

The permutations of a round involve looking at possible member-orderings for each agent-system. There is no set boundary on the number of permutations unless specifications are clearly made for extraction of unique entries only. The member-orderings serve as an important factor towards creating the best potential setup formation for a match success.

5.0.25. Process

The process includes the steps taken to complete the simulation for that round. In this case, these correspond to the following: the ordering applied by a permutation, the generation, the (selection,

excess, remaining), and knowledge processed for each agent system in conclusion of the round. The key internal metrics are used here to conduct the round process.

5.0.26. pyreason facts

A fact is a verifiable piece of information from an external source. The data can be directed towards a number of agents within their respective agent system. The diffusion of a fact is relevant based upon the graph structure defined for an agent system. A fact is expected to be applicable during simulation of the match. The system specifications are a place of reference for understanding the time variations of when and how long the fact will hold. The PyReason software allows for definition of facts through an identifier and the number of timesteps it is applicable for. There are some considerations to be noted for areas where a fact can be created.

The software referenced above can be accessed here: https://github.com/lab-v2/pyreason

5.0.27. pyreason graph

A graph is a way to represent data through a structured representation. The storage of data in this format is a container for the agent-system knowledge. The display of the data could happen directly as one main graph for the agent-system or as individual sub-graphs for each agent in the agent-system. The graph is an important point of reference towards the application of facts and rules during simulation of the match. The PyReason software defines graph types in the .graphml file format. The structure of the data will need to be aligned towards the acceptable format for the .graphml extension. Below illustrates an example of what a graph may look like for an agent system or an agent from the agent system. The semantics of the data representation depends upon system specifications.

The software referenced above can be accessed here: https://github.com/lab-v2/pyreason

5.0.28. pyreason rules

A rule is a way to establish a set of criteria to be met. The criteria can be specified by a number of clauses where each clause contains a number of conditions to be satisfied. The firing of a rule happens based by meeting each of the clauses defined. Depending on the number of rules, an interpretation can then be measured for a particular time-step. The rules stand out as key identifiers for processing the entities involved for a given interpretation of a time-step. The PyReason software defines a rule as a predicate through a number of collective observations that lead to a definite conclusion. There are some cases to consider for creation of rules during the simulation of a match.

The software referenced above can be accessed here: https://github.com/lab-v2/pyreason

5.0.29. pyreason interpretations

An interpretation is a process to evaluate the data. Depending on the entities affected for a timestep, it stands out as an important frame of measure towards looking at future applicable interpretations proportional to the number of timesteps. The PyReason software displays interpretations in the following format: [entity] [annotation]. The entity can be described by an identifier and the annotation is a markup feature relative to the entity. As an interpretation relies upon a number of rules and a applicable number of facts relative to the graph, these are some key areas to consider for understanding an interpretation's dataframe for a timestep.

The software referenced above can be accessed here: https://github.com/lab-v2/pyreason

5.0.30. Recorder

The recorder for a round is a display for the agent to make applications with their knowledge given encoded data. The knowledge set aside for each agent can stay the same or can vary depending upon the round number end regarding the types of data encoded. There are some limitations to recognize such as depletion of knowledge before the final round number and the halting of unused knowledge leading to the same display of data until the final round number. Such shortcomings provide cases to avoid for safer and secure applications of agent knowledge.

5.0.31. Relevance

The relevance of a round is measure of the impact it has on consequent rounds simulated. Extensions of this metric can also be made towards chosen rounds in particular or at random for observing the impact one round has on another until the round number end. Some measures to consider for the contribution a round provides towards the match are in the forms of percentages, annotations, keywords, labels, and real numbers. It is highly beneficial for how the use of such measures are applicable to be clarified in the system requirements.

5.0.32. Remaining

The round remaining is one of the main internal metrics used during simulation of a round. The display of an agent's encoded knowledge for usage means any knowledge contained within the agent after is what remains. The left over knowledge provides an area of analysis towards understanding any contributing factors towards why the knowledge exists within the agent-system rather than flushed out or why that particular branch of knowledge was not chosen for selective applications. As this stands out as a key internal metric, the information is collected by the system for post-processing of data after the simulation of a round is completed.

5.0.33. Rule List

A list of rules are defined for each agent system to abide by for each round simulated. The rules may change or stay the same depending on further observation of internal and external metrics measured for

both agent systems. An explicit definition of any number of rules work to constrain each agent system's capabilities to make appropriate applications with their knowledge. They also take into consideration requirements defined for the system and other extraneous factors looked into for the system build.

5.0.34. Selection

The round selection is one of the main internal metrics used during simulation of a round. The display of an agent's encoded knowledge for usage allows for an agent to create some form of value. In this case, using such knowledge can be tied towards building a unified or co-independent strategy within the agent system. The knowledge used for selection provides an area of analysis towards understanding any contributing factors towards why extensions were made with this knowledge. As this stands out as a key internal metric, the information is collected by the system for post-processing of data after the simulation of a round is completed.

5.0.35. Simulation

The round simulation utilizes the 3 main components (selection, remaining, excess) for completion of a round in order to put into perspective the set of moves made by each agent part of their respective systems. This structure provides a base foundation towards the writing of the round language, round description, and round documentation.

5.0.36. Singleton

The round singleton is a capture of internal and external metrics stored within an object store. It can described as an instance where the state of this information is securely encapsulated and access points are created for either analyzing, modifying, or receiving the information.

5.0.37. Solution

The round solution is a diffuse in the form of knowledge representation passed through to affect internal and external metric measures, including encoded data within both agent systems. Classifying a solution spread out depends upon the system requirements and how the knowledge is affected across both systems.

5.0.38. State

The round state is a confirmation message through a number that shows completion of that round before moving on to the next. One connection can be traced from one state to another until the final round number. The checkpoint stands out as an important receiver for any conditional logic to be checked before starting the next round.

5.0.39. Statistics

The round statistics are extractions of pieces of data from the knowledge contained within agent-systems or from internal and external metrics. Such information pulled for cross-examination provides a means to organize the information into a representative-format (table, figure, diagram, file) for understanding the main takeaways from the round simulated.

5.0.40. Strategy

The round strategy is a plan created by any number of agents in the agent system to persist knowledge contained within the system for longer periods of time. Depending upon the system requirements and factors such as the scalability and stability of the system, the knowledge will not stay stale due to increasing demands to make use of it during the round simulation.

5.0.41. Timestamp

The round timestamp is a confirmation of the round simulated before continuing into the next round. A timestamp stands out as a collective valuable that is a validation mechanism before simulation of the next round. Depending on system requirements, there holds a chance that some rounds are simulated, but do not hold a timestamp. This is in part due to a failure occurring during the system run, or a failure to meet conditions proposed by the system for any specified metric.

5.0.42. Transducer

The round transducer is a computation calculated by the knowledge contained within each agent system. Using the base components (selection, remaining, excess) to model each agent's move, an output can be derived. Depending upon system requirements, they serve as a reference guide to understand the output to be generated. The computation holds a strong correlation towards the plan devised within each agent system for execution (success, failure) of a strategy. The applications of such knowledge within each agent system signal the need to require a result for purposes of converging towards some sort of outcome.

6. Temporal Logic Formulas For Match Initialization

The following describes formulas for initiation of a match before it begins for both agent systems.

A temporal logic formula can be measured as a way to either see if the initial prediction was correct, serve as a useful point estimator for data collection converging to a general match type given sample data collected over time, or a way to establish boundaries affecting the structure of operations to be carried out.

We will look at 5 formulas classified by their roman numeral representations.

Type I: The setup of the arena requires a mapping of parameter initializations towards their corresponding positions.

For each agent in the agent system, map to a random position with labels as A, B, C, and so on. A position can be defined as a coordinate point as a representation for the agent location. Depending on the grid boundaries, the random position for the agent will be mapped towards its respective boundary.

Type II: The diffusion of knowledge from system requirements across each agent system for awareness of incoming contract obligations to adhere to.

Defining system requirements takes into account factors such as scalability, adaptability, and reliability. Due to such needs affecting the system's life-cycle, the extraction of knowledge sent over to agents within their agent system places limitations upon an agent's capabilities.

To define the process explicitly is as follows.

1. Create a knowledge base from understanding of system. 2. Break it down into sub-knowledge components for agent to process with above factors taken into consideration. 3. Hold from start to end of match. 4. Measure any occurrence for failure of agent to meet obligations.

Type III: Conditions specified on environment based by incoming domain knowledge known or forecasted.

The environment each agent system partakes in is an unknown variable and requires a level of domain knowledge for understanding or predicting miscellaneous factors affecting the operations carried out by each agent system.

The process for how such an event can occur is stated below.

1. An unknown event occurs based upon environmental conditions imposed. 2. The target can be directed towards both agent systems, affecting the knowledge-base structure. 3. Information collected by internal and external metrics during match simulation are not accurate.

Type IV: Conditions specified on agent to operate in the given environment location based by system requirements.

The formula stated from Type III is tied towards Type IV for purposes of structuring knowledge within each agent system. It stands out to maximize maintainability of an agent system's knowledge base.

A visualizer is shown below for the agent system initial placement.

1. Deploy each agent from agent system into environment location. 2. Structure agent system knowledge base accordingly given any environmental constraints. 3. Prepare for a number of restructures to occur if event of any scale distributes towards an agent system.

Type V: See a theory, proposal, or a document observed as a representative measure over the course of the match.

Classify theory, proposal or document as a formulation. Applications of this formulation is applied from the start to end of match.

The notation for a formulation is shown below.

1. Apply a conversion of knowledge obtained to an identifier for cross-checking an applicable match against an agent system's knowledge base. 2. Create a knowledge repository for any matches as learned information during the match simulation. 3. Store the learned information into a branch. 4. Annotate formulation with the corresponding branch of knowledge repository.

7. Match Process

Here is what a match could look like given the setting of the match initialization, the knowledge bases contained within each agent system, and the use of internal and external metrics during the simulation.

Below defines the model for each agent system as a base for a match-build by considering the specific areas of study the knowledge is applicable to.

1. Agent Knowledge Encoding (Given)

   Variable amount of knowledge $\rightarrow$agent's inborn traits.

   The traits for an agent are displayed below.

   Lifeline: A measure of the agent's sustenance.

   Movements: The number of moves completed by agent.

   Regeneration: The restoration of the agent's lifeline if in crisis.

   Transfer: Ability to send knowledge to various storage units.

   Backup: Functionality to diffuse knowledge into other sub-agents.

   Division: The sub-division of knowledge into other sub-agents.

State: A tracker for measuring usage of any agent component.

Path: A explicit tracing of any agent's component state.

Features: Distinct attributes or characteristics from agent.

2. Agent Knowledge Addition (Transferred)

A separate container of knowledge added to enhance an agent's capabilities.

Domain: A generalization to an area, field or branch of study.

Category: A subdivision of a generalization as labels.

Product: A label mapping to any number of units.

Extensions: Further insights into containerized knowledge.

Access Points: A storage of knowledge into some object form.

3. Agent Knowledge Connectors (Attached)

A variable number of ties from the given knowledge encoding and transferred addition.

Mechanisms: A representation of parts with a designated role.

Connections: A way to establish a trace (Point A to Point B)

Configurations: A finalizer applicable to a connection's operations.

Bind: A process for putting together a number of configurations.

7.0.1. Model →Simulation

For each round completed, showcase the agent knowledge encoding, agent knowledge addition, and agent knowledge connectors within the respective agent system.

Here is a display of a round in action with the above agent knowledge in use.

Round 1

Consider the agent knowledge store to be subdivided into separate sub-stores of knowledge.

The classification of a knowledge store for a match-build in this case is a deck. The knowledge store can be distributed across different knowledge sub-stores as cards. In the round sample shown, 1 store is used →7 sub-stores.

Listed below is the information on each sub-store. For each card, consider the following connotations to describe the usage of the card during the simulation of the match.

Unit Tag: Denoted by a label to identify unit

Unit Assembly # Of Steps: The number of iterations needed to build the unit

Unit Success Build Rate: A percentage measured for configuration of unit

Unit Industry Type: Domain knowledge for unit classification

Unit Supply Count: The amount of stock available for the unit

Unit Information: A description of the unit's purpose and end-goal

Consider the setup of the card to involve the connotations discussed previously as part of a category. The unit mapped out by the card, in this case, follows the boilerplate template starting from domain :: category :: product. This template is usable for any future builds executed as part of characterizing the agent knowledge addition. Below shows a depiction of a randomized set of cards shuffled from the container for a round. For the list of cards shown, consider the naming conventions for each unit configuration in this example to come from games of wizardry. However, depending on the knowledge base used, a new set of naming conventions can be defined for the connotations.

Card (Sub-Store 1):

Unit Tag: Tower Shield

Unit Assembly # Of Steps: 0

Unit Success Build Rate: 100%

Unit Industry Type: Utility

Unit Supply Count: 3

Unit Information: Applies a -50% Damage Ward

Card (Sub-Store 2):

Unit Tag: Guiding Light

Unit Assembly # Of Steps: 0

Unit Success Build Rate: 100%

Unit Industry Type: Utility

Unit Supply Count: 2

Unit Information: Applies a +35% Healing Charm

Card (Sub-Store 3):

Unit Tag: Tower Shield
Unit Assembly # Of Steps: 0
Unit Success Build Rate: 100%
Unit Industry Type: Utility
Unit Supply Count: 3
Unit Information: Applies a -50% Damage Ward

Card (Sub-Store 4):

Unit Tag: Cleanse Charm
Unit Assembly # Of Steps: 0
Unit Success Build Rate: 100%
Unit Industry Type: Utility
Unit Supply Count: 5
Unit Information: Remove a negative charm from ally

Card (Sub-Store 5):

Unit Tag: Weakness
Unit Assembly # Of Steps: 0
Unit Success Build Rate: 100%
Unit Industry Type: Utility
Unit Supply Count: 4
Unit Information: Applies a -25% Damage Charm

Card (Sub-Store 6):

Unit Tag: Stun Block
Unit Assembly # Of Steps: 0
Unit Success Build Rate: 100%
Unit Industry Type: Utility
Unit Supply Count: 2
Unit Information: Resists the next 2 incoming stun effects

Card (Sub-Store 7):

Unit Tag: Guiding Light

Unit Assembly # Of Steps: 0

Unit Success Build Rate: 100%

Unit Industry Type: Utility

Unit Supply Count: 2

Unit Information: Applies a +35% Healing Charm

Each card's designation denoted by the connotations can act as a sub-store within the agent container of knowledge. The data from a card can be made available to an access point for reference. Depending upon the system specifications, there are 2 types of access points to consider: local and central. A local access point allows for a navigable view from one card to another, excluding the visiting order of the cards. A central access point is a containerized view of the set of cards accessible during a particular round.

As this corresponds to one round conducted during the match simulation, an agent from the agent system is expected to make a card-selection, conforming to the base components required for simulation of a round.

Here is a sample scenario for a card an agent could select from their container of knowledge. The applications of their knowledge will indirectly affect the agent knowledge encoding for subsequent rounds of the match simulation. The details of the process are explained in depth, showing each of the agent's traits in action.

CARD SELECTED: Guiding Light (Agent Selection)

The lifeline of an agent is in crisis if circumstances warrant regeneration. The circumstances happen based by the rules of the match stated by system specifications. This is applicable towards both agent systems participating in the match simulation.

An agent can also allocate an amount of finite space for distribution of knowledge from a container. The knowledge can be separated across a number of sub-knowledge stores based upon knowledge learned during the match simulation. This piece of functionality is applicable for each agent system part of the match simulation. A state can be measured for changes that occur to the sub-stores over the course of the match. A path can be traced for each round until the end round number as an illustration for the changes persisted across the sub-stores overtime.

Depending on the how the round generation works, it drives the potential towards introducing sub-agents as entities for knowledge distribution for an agent part of a agent system. The creation of a backup as a way to store agent knowledge is an accessory for subsequent rounds completed by the agent system. The process for how the knowledge is distributed requires a division, or portions of agent knowledge copied into a sub-agent.

The movements of an agent are dependent upon the agent knowledge connectors. For a given number of binds on an agent, a movement can be traced as a step that leads towards an action completed by the agent. The process for movement creation relies on an understanding of how each bind operates. Depending on system specifications, an agent can make a number of movements for completion of an specific action defined, allowing for further agents within the agent system to do the same.

The above introduces a number of agent capabilities for management of knowledge during the match simulation. Now, given the set of cards listed above and the card selected, the agent has a number of tools available for maintenance of knowledge encoding and addition over the course of the match.

The main aspects of a round can now be described by the following: a set of cards assigned for each agent within the agent system, a card selection, and a set of tools available for management of agent knowledge encoding and addition.

8. Tables

Below is a table displaying information in a row X column format for each round simulated during the match. The table shown provides a level of intuition towards the extraction of each round as a feature to further understand the data collected. Parts of the data can be analyzed towards the simulation of future match-builds or for making changes to existing match-builds. This is dependent on the amount of data examined either for a round, a set of rounds, or for each round of the match. The table is listed below, following the match process from section VII, where each round corresponds to a row and the data associated for that round is placed within the respective column. The simulation of the match is performed until 5 rounds. Each round illustrates a possibility of agent-capabilities in action for each agent system, given the model used during simulation of the match. A possibility refers to a randomized generation of a set of cards from the container of knowledge, including a number of ways for an agent to manage their knowledge over the course of the match.

Listed below are the sample bindings applied onto the agent for a match run of 5 rounds. The number of movements is determined based upon the bindings applicable during the simulation of a round in the match. Let the lifeline of the agent to start at 100 hit-points. The table shows usage of the agent capabilities in action for each round conducted in the match. It is possible that for some rounds, an agent may exclude usage of a capability at any given round during the match simulation. The goal of each agent system is maximize the effectiveness of a number of strategies carried out over the course of the match. By taking this into account, it allows for agents within their respective agent system to evaluate the appropriate set of capabilities for optimal deployment of a strategy per round.

Consider the naming conventions for each bind to come from games of wizardry. In this case, an agent can be modeled as a wizard for each bind placed.

Agent 1 - Agent System 1

Mechanism: Abstract Aeon Helm
Connections: Abstract Aeon Helm →Agent Head
Configuration: Place Abstract Aeon Helm on Agent Head.

Mechanism: Abstract Aeon Cape
Connections: Abstract Aeon Cape →Agent Body
Configuration: Place Abstract Aeon Cape on Agent Body.

Mechanism: Abstract Aeon Boots
Connections: Abstract Aeon Boots →Agent Foot 1, 2
Configuration: Place Abstract Aeon Boots on Agent Foot 1, 2

Mechanism: Flamenco Tocador (Breed of Pet)
Connections: Flamenco Tocador →Agent Companion
Configuration: Incorporate Flamenco Tocador with Agent

Mechanism: Abstract Aeon Sword
Connections: Abstract Aeon Sword →Agent Hand
Configuration: Place Abstract Aeon Sword on Agent Hand.

Mechanism: Abstract Aeon Athame
Connections: Abstract Aeon Athame →Agent Tool-Kit
Configuration: Place Abstract Aeon Athame inside Agent Tool-Kit

Mechanism: Abstract Aeon Charm
Connections: Abstract Aeon Charm →Agent Collar
Configuration: Place Abstract Aeon Charm on Agent Collar.

Mechanism: Abstract Aeon Ring
Connections: Abstract Aeon Ring →Agent Index Finger
Configuration: Place Abstract Aeon Ring on Agent Index Finger.

Mechanism: Abstract Aeon Deck
Connections: Abstract Aeon Deck →Agent Container Of Knowledge
Configuration: Integrate Abstract Aeon Deck with Agent Container Of Knowledge.

The above shows the bind process of agent 1 of agent system 1 put together for 9 configurations.

Agent 1 - Agent System 2

Mechanism: Eternal Inspired Helm
Connections: Eternal Inspired Helm →Agent Head
Configuration: Place Eternal Inspired Helm on Agent Head.

Mechanism: Eternal Inspired Cape
Connections: Eternal Inspired Cape →Agent Body
Configuration: Place Eternal Inspired Cape on Agent Body.

Mechanism: Eternal Inspired Boots
Connections: Eternal Inspired Boots →Agent Foot 1, 2
Configuration: Place Eternal Inspired Boots on Agent Foot 1, 2

Mechanism: Decade Piggle (Breed of Pet)
Connections: Decade Piggle →Agent Companion
Configuration: Incorporate Decade Piggle with Agent

Mechanism: Inspired Aeon Sword
Connections: Inspired Aeon Sword →Agent Hand

Configuration: Place Inspired Aeon Sword on Agent Hand.

Mechanism: Inspired Aeon Athame
Connections: Inspired Aeon Athame →Agent Tool-Kit
Configuration: Place Inspired Aeon Athame inside Agent Tool-Kit

Mechanism: Inspired Aeon Charm
Connections: Inspired Aeon Charm →Agent Collar
Configuration: Place Inspired Aeon Charm on Agent Collar.

Mechanism: Inspired Aeon Ring
Connections: Inspired Aeon Ring →Agent Index Finger
Configuration: Place Inspired Aeon Ring on Agent Index Finger.

Mechanism: Inspired Aeon Deck
Connections: Inspired Aeon Deck →Agent Container Of Knowledge
Configuration: Integrate Inspired Aeon Deck with Agent Container Of Knowledge.

The above shows the bind process of agent 1 of agent system 2 put together for 9 configurations.

Match	Lifeline	Movements	Regeneration	Transfer	Backup	Division
Round 1 (Agent 1 - Agent System 1)	X	X	X	X	Data6	X
Round 1 (Agent 1 - Agent System 2)	X	X	X	X	X	Data13
Round 2 (Agent 1 - Agent System 1)	X	X	X	Data5	X	Data7
Round 2 (Agent 1 - Agent System 2)	X	X	X	Data11	Data12	Data13
Round 3 (Agent 1 - Agent System 1)	X	X	X	X	X	X
Round 3 (Agent 1 - Agent System 2)	X	X	X	Data11	X	Data13
Round 4 (Agent 1 - Agent System 1)	X	X	X	Data5	Data6	Data7
Round 4 (Agent 1 - Agent System 2)	X	X	X	X	Data12	Data13
Round 5 (Agent 1 - Agent System 1)	X	X	X	Data5	X	X
Round 5 (Agent 1 - Agent System 2)	X	X	X	Data11	Data12	X

TABLE 1 *Match Simulation for Round 1, 2, 3, 4, 5 given 2 agent systems each with 1 agent*

In the table shown above, the consistent traits of an agent in use for each round include the lifeline, the movements, and the regeneration. The regeneration may not be required for each round completed by the agent, but it warrants a check to be performed for securing an agent's lifeline. The other traits of an agent may require further considerations depending on the use case defined for it throughout the match. A use case helps to portray the potential that an agent trait can have to itself or within their respective agent system for each round simulated. The system specifications are an important place of reference for understanding particular use-cases of an agent's traits in action during the match simulation.

Displayed below is a sample system specifications doc for information of the consistent agent traits in action and the use cases written for any other agent traits used based upon the table for one round simulated of the match.

Round (Agent 1 - Agent System 1)

Lifeline: ✓

Agent # of Hit-Points: 100

Distribution of Hit-Points Randomized For Agent-Bind Process

Agent Head ← Abstract Aeon Helm (10 hit-points)

Agent Body ← Abstract Aeon Cape (15 hit-points)

Agent Foot 1, 2 ← Abstract Aeon Boots (10 hit-points)

Agent Companion ← Flamenco Tocador (Breed Of Pet) (10 hit-points)

Agent Hand ← Abstract Aeon Sword (5 hit-points)

Agent Tool-Kit ← Abstract Aeon Athame (10 hit-points)

Agent Collar ← Abstract Aeon Charm (10 hit-points)

Agent Index Finger ← Abstract Aeon Ring (10 hit-points)

Agent Container Of Knowledge ← Abstract Aeon Deck (20 hit-points)

Movements Of Agent → Action

The action for an agent is to select a card and cast it from their container of knowledge.

Movement Options

Agent Head:

Agent Body: ✓

Agent Foot 1, 2: ✓

Agent Hand: ✓

Agent Tool-Kit:

Agent Collar:

Agent Index Finger:

Agent Container Of Knowledge:

Number Of Total Movements: 3

Movements Visualization

Adjustment of the body leading towards a sharp movement by the feet in one direction and closing in on the hand to create a circular, rotational path converging in one central point.

Regeneration

Loss Of Hit-Points: 10

Gain In Hit-Points: 5

Net: Loss - Gain = 5 Hit-Points Retrieved

Round (Agent 1 - Agent System 2)

Lifeline: ✓

Agent # of Hit-Points: 100

Distribution of Hit-Points Randomized For Agent-Bind Process

Agent Head ← Eternal Inspired Helm (10 hit-points)

Agent Body ← Eternal Inspired Cape (15 hit-points)

Agent Foot 1, 2 ← Eternal Inspired Boots (10 hit-points)

Agent Companion ← Decade Piggle (Breed Of Pet) (10 hit-points)

Agent Hand ← Inspired Aeon Sword (10 hit-points)

Agent Tool-Kit ← Inspired Aeon Athame (10 hit-points)

Agent Collar ← Inspired Aeon Charm (10 hit-points)

Agent Index Finger ← Inspired Aeon Ring (10 hit-points)

Agent Container Of Knowledge ← Inspired Aeon Deck (10 hit-points)

Movements Of Agent → Action

The action for an agent is to select a card and cast it from their container of knowledge.

Movement Options

Agent Head: ✓

Agent Body: ✓

Agent Foot 1, 2: ✓

Agent Hand: ✓

Agent Tool-Kit:

Agent Collar:

Agent Index Finger:

Agent Container Of Knowledge:

Number Of Total Movements: 4

Movements Visualization

A surge spreading from the center and shifting across the body, leading towards a levitation of the head, the hands, and the feet.

Regeneration

Loss Of Hit-Points: 10

Gain In Hit-Points: 5

Net: Loss - Gain = 5 Hit-Points Retrieved

Criteria for movements →action 1. There should hold an area of relevance given each movement that leads toward the action. Each movement should allow for a mode of transport towards the next upcoming movement. 2. A requirement of stability should be satisfied for each movement. There should hold no potential errors that lead towards failure of the action. 3. A level of maintenance for the movements is necessary for future rounds in the match simulation, depending on system specifications, necessitating further usage of the movements to be completed by an agent.

The completion of each movement for visualizing the action demonstrated by the agent provision a new focus on inner distribution of knowledge within an agent towards a successful cast of the card displayed during the match simulation.

The variations for the movement are dependent upon system specifications for deploying of the card-cast within the agent system. A set of criteria for movement variations and a release of the action based upon the number of movements serves as a critical evaluator in this process and should be clarified within system specifications.

The regeneration is an important part for restoration of the agent through a point-based system. This allows for manual evaluations to be made for the lifetime of the agent during the simulation of the match.

9. Conclusion

This paper presented a comprehensive overview of event-based components required for a match-build. The external and internal metrics, including the agent knowledge, allow for match creation.