Synthetic Cognitive Agent Situational Awareness Components

Sanford T. Freedman
Julie A. Adams
Department of Electrical Engineering and Computer Science
Vanderbilt University
Nashville, Tennessee 37240

July 2008

Abstract

Synthetic cognitive agents often require an understanding of the environment's state and the ability to predict the effects of actions on the environment and self. Bestowing agents with situational awareness may increase understanding of domain dynamism, action selection, and action execution. While contemporary architectures have made considerable strides, present architectures have yet to achieve human levels of situational awareness. If future architectures are to realize high levels of situational awareness, agent designs may necessitate certain features. This paper presents an initial exploration into the features and design components required to realize cognitive agent situational awareness.

1   Introduction

Synthetic cognitive agents (SCA) often require an understanding of the current domain state and the ability to predict how their actions will affect the world and their internal status. Bestowing SCAs with Situational Awareness (SA) may increase their understanding of the dynamism within a domain, action selection, and action execution. SA may benefit all SCAs to varying degrees, but the largest benefit may be seen by robotic agents and agents within high fidelity simulations due to their domains' real-time dynamic nature.

Current SCAs are ill-equipped to generate and maintain SA. Cognitive architectures exist, but none specifically generate and foster SA. These architectures roughly fall into two paradigms [Hawes et al., 2007]. The first aims to model human cognition, examples include Soar [Lehman et al., 2006] and ACT-R [Anderson et al., 2004]. These architectures do not inherently incorporate sensor and actuator reasoning, are

not amenable to combining diverse inference methods, and real-time performance is not a primary design goal. The second, including SSS [Connell, 1992] and AuRA [Arkin and Balch, 1997], seeks to realize intelligent robotic systems. Contemporary robots are generally designed for a particular task or domain and often do not have the requisite knowledge and capabilities required to handle errors and situations that were not envisioned. An exception are agents designed with minimal to no prior knowledge and abilities that are expected to learn all required skills and abilities. Unfortunately, these designs only manage initial progress before learning halts. It is theorized that these agents fail to continue developing due to a lack of sufficient types of knowledge representation and mental machinery [Minsky et al., 2004] [Minsky, 2006].

In order to design systems capable of achieving SA, the phenomena must be understood. SA, a concept receiving considerable attention in the human factors community since the late 1980's, considers humans' ability to understand and make decisions within complex dynamic environments. Endsley defines SA as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future" [Endsley, 1988].

While often misinterpreted as a process or a set of processes, SA groups processes working to form the human cognition [Flach, 1995]. Portions of these processes handling incoming data and computing higher level constructs can be grouped together by SA. Endsley partitioned SA into three levels. Level 1 acquires raw information from the environment. Level 2 is the comprehension of level 1 data. The final level utilizes level 1 and 2 constructs to predict future environmental states and potential actions [Endsley, 1988].

SA benefits humans operating within complex, dynamic domains in numerous ways. SA may increase a human's understanding of an environment, evolution of that environment, and how actions affect task progress. SCAs may be developed to operate in domains and perform tasks with similar challenges. SCAs may replace humans for particularly dangerous or unpleasant domain/task combinations. SA capabilities may assist SCAs in the same manner as humans.

Similar to humans, SCAs may require a comprehensive understanding of the present state of their tasks, goals, and environment. This awareness includes comprehension of dynamic obstacles, events, and other agents within the given environment, and even their own internal mechanical and cognitive state. This understanding may not be realizable at design time, but requires the fusion of current sensor readings with a priori information. These sensor readings may be provided by a diverse collection of sensors, lack consistency, or provide erroneous information. Akin to humans, SCAs will be required to process and understand these sensor readings in real-time. Despite this difficult cognitive task, actions will be required, if only to provide better sensor readings.

Action selection within these conditions can be challenging, even for humans. With the combination of environmental dynamism, non-ideal sensors, and domain complexity, a complete domain understanding may be infeasible as the domain may be too complex to model completely. Models can approximate the environment, but may require constant updating as dynamic objects mutate the environment and the domain evolves. This maintenance may include previously unavailable information and corrections to erroneous model sections. Inherently incomplete and constantly requiring updating, models should include the most important environmental aspects, as they

relate to the SCA so that efficient and productive action selection can be performed.

Selecting and performing actions may generate errors. Error sources include flawed models, erroneous actuation, and unexpected dynamic events. SCAs need to be able to detect and understand these errors so that remediation may be performed. In order to improve efficiency and lessen the chance for destructive consequences, errors need to be predicted and prevented. This requires the ability to detect situations where errors frequently occur.

This paper describes a non-exhaustive list of agent architecture features required to achieve SA. A list of design mechanisms is also provided that may be utilized to achieve the features required for the generation of SCA SA.

1.1 Agent architecture features for achieving SA

Domains requiring SA often involve information rich environments with numerous dynamic obstacles. As real-time performance is often paramount, humans have a difficult time achieving and maintaining SA [Endsley, 1988] and SCAs may find the domains equally challenging. At least seven primary features are required by both humans and SCAs in order to achieve and maintain SA. This section describes each of these features.

Information filtering aids SCAs in navigating the voluminous data gathered. Non-ideal sensors coupled with imprecise actuators often require SCAs to maintain multiple concurrently operating sensors, each potentially producing copious amounts of data. Additionally, as SCAs increase in complexity, internal self monitoring of both physical and cognitive subsystems may grow in importance for successful task completion. Significant portions of incoming data may be extraneous or of relatively low importance, requiring filtering to ensure relevancy. Information can be further partitioned into data deemed beneficial to each SCA subsystem. Information may also be presented with differing time scales and complexity. Low level sensor data may be available at several hertz or faster while complex symbolic constructs may only be introduced on the order of seconds. Different components will require more processing and data must be filtered so that inference methods are aligned with the frequency and complexity of incoming data.

Error detection and correction are critical for operating within complex, dynamic domains as perfect modeling may be infeasible. Dynamic objects operate unexpectedly, invalidating models and causing inappropriate actions to be taken. Additionally, SCAs generally possess non-ideal sensors and actuators that produce erroneous information and fail to execute perfectly. Errors need to be detected and understood for prevention and correction.

Future SCAs may be placed in non-uniform domains, tasked with multi-faceted goals, each benefiting from different capabilities. In order to generate high levels of performance and efficiency, SCAs may need to *adjust* their configurations to align with given tasks. Additionally, tasks may have varying priority levels and the capabilities best equipped to complete the highest priority tasks should be provided with additional resources. Accomplishing this alteration requires the SCA to be adjustable. Depending on the particular domain, task, and capabilities, adjustments may occur at design time or while situated within the field.

A *diverse array of capabilities and data representations* may improve a SCA's ability to exist and operate in complex domains. SCAs may need to observe the environment from differing Realms of Thought [Minsky, 2006]. Required SCA capabilities may include: voice recognition, natural language processing, vision, object recognition, physical manipulation, and symbolic cognitive processing. Within each realm, numerous algorithms and implementations exist, each potentially optimal for different situations. As SCAs and their tasks escalate in complexity, the need for a wider capability coverage may grow. SCAs may also require redundant mechanisms for parallel processing of similar data and minimization of errors and uncertainty via comparing results from different inference methods.

Modular SCAs may require efficient and effective *inter-module communication*. Complex domains may require specialized functionality, but modules must communicate. Results from multi-modal sensing and inference are required for the production of composite level 2 and 3 SA constructs. Due to the volume of available data, uncertainty of diverse domains, and non-ideal sensors, information must be effectively and efficiently communicated between modules. Some information and generated constructs may be of little use, unless combined with constructs from another module. Many constructs may require serial processing. When redundant modules are operating concurrently, communication is required to effectively merge and utilize their outputs.

Information available to SCAs can quickly and frequently become stale. Achieving and maintaining SA may require *storing volumes of information and purging of stale data*. The rate at which data grows stale may be modality and task dependent, but SCAs must detect stale data and utilize architectural designs that encourage or force the removal of stale information, including a reduction of redundant information.

1.2 Agent architecture mechanisms

A number of mechanisms may help realize the required features for developing SA within SCAs. These features may be difficult to implement directly, but they may emerge from the interactions of these mechanisms. A non-exhaustive list of mechanisms are presented in this section.

Non-fixed designs are based on the premise that no single framework, inference method, or data representation is capable of achieving broad levels of intelligence [Minsky et al., 2004]. Non-fixed designs allow for a wide range of potentially diverse capabilities to be incorporated into one coherent system. Such designs frequently possess the ability for multiple instances of various resources to be present in the architecture. As non-fixed designs often do not mandate that all components be permanently active, these architectures do not process a fixed control path, but adjust control depending on the architecture's current configuration.

Hierarchical architecture design requires a layered structure. This modular design generally places the lowest hierarchical levels among low level sensor data and less complex cognitive concepts. At each successive hierarchical level, cognitive constructs grow in complexity that often merge constructs from lower levels. Often, hierarchies abstract time, with the slowest updating processes at the summit and faster processes at the base of the hierarchy [Connell, 1992].

Binding represents two critical forms of functionality, combining results from multiple modules or sub-systems and generating high level representations or constructs. The SCA produces comprehensive information that is more complete and beneficial by using information from multiple sources. The fused results represent higher level constructs, that enable higher level processes to operate more efficiently. Some binding systems utilize references to the original data used to form the resultant constructs. These data pointers can provide grounding, creating a relation between a concept and the entity it represents [Hawes et al., 2007].

Software design can be classified into two distinct extremes, modular and labyrinthine. Modular designs generally have closed modules and fixed APIs, with internal mechanisms and data representation not externally available. *Labyrinthine designs* reveal their inner data structures and processes and may accept wider ranges of inputs, providing a more flexible system [Sloman, 1989]. SCAs developed with a labyrinthine mindset may allow for tighter integration between currently disjoint technologies.

Embodiment represents architectures with a physical form [Feil-Seifer and Matarić, 2008] and an understanding of that form. A physical form endows a SCA with the ability to affect its environment and directly sense the status of the domain. Some systems may incorporate a physical form, but true embodiment requires cognitive capabilities to both realize that the form exists and understand the effects of its environmental interactions.

Metacognition, oft referred to as the "cognition of cognition" [Cox, 2005], differs from "regular" cognition as it deals with reasoning over one's mental models [Minsky, 1986], as opposed to the phenomena represented by those models. The term Metacognition has been utilized to refer to two associated, but separate capabilities, reasoning over one's cognitive functions and regulating cognition [Cox, 2005].

Modern software and SCAs can complete complex tasks in difficult and dangerous domains, yet fail at the simplest tasks that a young child can perform. This phenomenon results from a lack of *commonsense*, knowledge humans are expected to know and understand due to its importance and everyday nature [Minsky, 2006]. Endowing SCAs with commonsense may improve their ability to acquire and maintain SA. Commonsense may allow SCAs to better understand tasks and goals, possess a set of default actions and thoughts, and increase understanding of situational data requirements. While not receiving considerable attention (Minsky etal.2004), commonsense reasoning has been attempted via large knowledgebases, see [Liu and Singh, 2004] for a comparison of Cyc, WorldNet, and ConceptNet.

2 Required Features

Features required for SCAs to achieve human level SA can be difficult to implement. The following section details how different mechanisms may be used to realize those features.

2.1 Information Filtering

Information filtering can be achieved via several architectural components, including the components listed above. These components provide their benefits by diverse

means, making SCAs robust to domain and task changes.

Hierarchical designs filter information through bottom-up processing. Low levels filter, classify, group, and symbolize large volumes of data points via algorithms requiring minimal computation per datum. Increasingly computationally demanding tasks are then performed on the reduced data set. Higher level processes can take advantage of the lower level classification and filtering to utilize optimized algorithms, eliminating the need for computationally expensive exception case code. Filtering data in this manner is called *level-banding* [Minsky, 1986] and allows processes to operate at optimal levels of detail. SCAs in dynamic, complex environments, may be incapable of operating in real-time if level-banding is not employed to partition and organize the data. Hierarchies and level-banding allow SCAs to operate on multiple time scales when processing data [Connell, 1992]. Lower level processes operate at a higher frequency, processing large volumes of data while higher levels run at a slower frequency performing more computationally expensive operations. Higher level processes require the results from lower levels to be effectively merged.

Binding, combining low level results into higher level constructs, compliments hierarchical designs. Binding results are processed by higher level processes, affording inference over multiple perspectives and realms of thought while minimizing data volume. Binding may reduce processing redundancy. Reasoning over higher level concepts frequently requires access to a concept's constituent pieces, which may have undergone multiple transformations. Via data pointers, higher levels can access previous forms of the data, reducing the need for redundant transformations.

Data pointers require at least a partially labyrinthic design. Processes within the paradigm expose internal data structures via binding's data pointers. Higher level processes may minimize redundant processing by utilizing these intermediate results. Increased data processing efficiency may be realized though labyrinthine designs by improving the information processing pipeline. Increased parallelization may be possible by concurrently executing multiple low level processes and combining their intermediate results.

Intermediate results may assist information filtering, but SCAs require efficient inferencing. Non-fixed designs allow capabilities to be incorporated based on the current tasks and domain, optimizing information processing and filtering. Design choices may reduce computational demands and increase process output quality and reliability. Depending on the task and data, optimality of different inference methods and data representations may vary. Non-fixed designs allow for incorporation of a diverse array of options, even if those options were not inherently incorporated or supplied by the architecture. Non-fixed designs may also allow for multiple instances of the same inference method, allowing concurrent execution of similar information processing.

Non-fixed designs permit a diverse array of inferencing, but SCAs within complex, dynamic domains will rarely reside in stable states. Priorities fluctuate, inference methods start and stop, and SCAs' goals change. An instance of the traditional algorithm selection problem [Cox, 2005], metacognition may aid in selecting currently optimal inference methods while determining if current information processing and filtering are aligned with current goals and priorities.

Effective inference selection requires domain and task knowledge and commonsense reasoning generates defaults, preconceptions, and expectations providing proto-

typical understanding of the situation. Commonsense assists information filtering and processing via prototypical responses for time critical decisions. While potentially requiring searching of large databases, under some circumstances, database queries may be more efficient than processing sensor data. If accuracy is high or error penalties are acceptable, then utilizing commonsense reasoning may allow reduced processing. Additionally, commonsense results may allow for directing attention and optimizing selection of mental models. Through directing attention, acquiring necessary information may be quicker and reduce the search for requisite data. Optimal mental model selection may increase incoming data processing efficiency, requiring fewer sensor readings.

Further reducing sensing demands, embodied SCAs reduce environmental search and data storage. SCAs that understand their sensors and environments may acquire information on demand, reducing the volume of incoming data for processing. Furthermore, by understanding sensors and actuators and how they operate within the environment, SCAs may require less data to determine their current state. As complexity grows, tasks become more involved, and environments more difficult, emergent interactions between the SCA and the environment will increase. Hardware understanding may allow for better detection and recognition of emergent situations, as opposed to inferring them via processing volumes of sensor readings.

SCAs working toward SA must process volumes of data. Information filtering improvements may increase SCA data stream processing efficiency, allow incorporation of additional sensing modalities, thus increasing SCA SA.

2.2 Error detection and correction

Achieving and maintaining SA requires both sensing and inference error detection while minimizing and correcting those errors. Errors derive from sources, such as, missing information and entities taking unexpected actions.

Inference methods and data representations must support error correction via modification and removal of erroneous knowledge. Depending on the domain and task, error correction methods may change, requiring non-fixed designs. Non-fixed designs allow inference methods to be activated or incorporated that align with existing and future error correction techniques. Non-fixed designs may also increase error detection. Redundant inferencing methods may permit comparison of multiple independent data sources. When differences arise, the likelihood of an error may be elevated. A SCA may be incapable of determining which result is in error, but realizing the possible presence of an error will permit additional error detection and analysis techniques.

Commonsense reasoning can provide default values and expectations for particular situations. Defaults can be combined with other inference methods to assist in error detection. When deviations from more traditional methods occur, either special case situations have arisen or errors have occurred. Comparing defaults may increase error awareness, triggering more in-depth error analysis. Default values may also prevent errors. Commonsense reasoning may generate lists of possible common errors occurring in particular situations. Using these predictions, SCAs can either take direct actions to prevent these errors or simply utilize more conservative inference and sensing strategies.

Defaults may aid metacognition in detecting and preventing errors. Attention can be directed towards areas within the environment or self that contain a high likelihood of generating erroneous sensor information. By directing attention, errors may be either prevented or detected quicker. Along with directing attention, metacognition can activate processes best equipped to detect and handle predicted errors. This includes not only selecting the most effective method, but activating a range of processes covering several "Ways to Think" [Minsky, 2006], generating a more comprehensive situational understanding. Metacognition may capitalize on uncertainty regarding facts and knowledge of previous performance to predict errors. While possibly task and domain dependent, learning may aid in detecting and correcting errors regarding environmental status, dynamic object states, and internal configurations.

Embodiment may aid hardware error detection as current sensors and actuators are non-ideal. Breakdowns, erroneous readings, and imprecise execution generate uncertainty that can result in misinterpretation and incorrectly updated belief states. Embodiment may enable a better understanding of errors, allowing SCAs to adjust belief states or utilize additional error testing mechanisms. In addition to detecting and correcting sensing errors, an increased understanding of how particular sensors operate within a domain may allow minimization of additional error cases. Attention may be directed to various points to acquire information pertaining to a particular fact or increased inferencing can be performed to understand possible emergent interactions.

Errors may further be reduced with binding. Reasoning over composite constructs may allow higher level processes to evaluate the consistency of lower level data. Lower level processes may verify that their results are successfully being incorporated into higher level constructs. When this incorporation fails, the likelihood of an error may increase.

Via data pointers, labyrinthine designs may assist SCAs in error minimization by reducing stale information. Stale data and unavailability of relevant information may force a SCA to miss beneficial opportunities within a dynamic, complex environment and reduce the amount of time available to side-step impeding error conditions. Intermediate forms of knowledge made available via labyrinthine designs may reduce errors by providing additional options for comparing potentially faulty inference results.

Inference and data processing errors impair a SCA's decision making within complex domains. SCAs with SA must be able to detect, prevent, and correct these errors if human level intelligence is to be achieved.

2.3   Adjustability

Accomplishing tasks within complex, dynamic domains may require diverse capabilities. Mandatory capabilities may change as tasks and domains change or progress is made toward tasks. SCAs need to adapt to current challenges and SA may provide some of that flexibility.

The diverse range of capabilities and resources a SCA may need requires a non-fixed design. Tasks may benefit from parallelization, requiring multiple instances of particular capabilities. Diverse capabilities may be required, but SCAs may be incapable of simultaneously supporting multiple inference methods and data representations. Non-fixed designs allow the incorporation of inference methods and data rep-

resentations required for the current task while minimizing the number and degree of modifications to the system. Reconfigurations may occur at run-time by activating and deactivating various modules or at design time with hardware and software modules being inserted or removed.

Diverse capabilities will be ineffective without control mechanisms as simultaneous activation of all capabilities may be impossible or counter-productive. Metacognition may provide activation and deactivation abilities, while reasoning over module interactions. Metacognition can also modify critical parameters to better align with the current domain state, present tasks, and belief state uncertainty. Priorities can also be modified to best reflect the current situation. Metacognition may enable higher levels of cognitive thought, allowing for reflection over current goals, ideals, and performance. SCAs can utilize selective and directed attention to influence processing demands, which may have an indirect effect on responsiveness and the quality of SA. Adjusting SA generation responsiveness may be affected by error probability and severity. Some domains require a slow, cautious pace. As multiple concurrent tasks are often required, SCAs may need to perform satificing, working towards a global maximum instead of several local maxima. Metacognition may also dynamically alter the ratio of bottom-up and top-down processing being performed.

Hierarchies may assist metacognition by altering bottom-up and top-down processing ratios. Designs utilizing multiple modules at each hierarchical level may have multiple bottom-up to top-down ratios realizing finer grained control. Further, priority can be moved between levels to influence level-banding and time-scales. When numerous rapid events occur, increased priority can be placed on lower level, fast time scale processes, but when small numbers of complex problems exist, a greater priority may be placed on slower, higher level inference methods. Hierarchical designs also allow changes at one level to have minimal impact on other modules and levels, if the interface remains the same.

Binding may increase multiple modules interoperability. Data pointers may allow modules in the middle of an inference chain to be modified with minimal effect on other modules. This allows for multiple inference methods and data representations to be used with minimal impact on the overall system.

Labyrinthine designs assist binding's data pointers ability to present data to other modules, as final output and intermediate data can be presented. With inference method adjustments, requirements on data format and contents may change and allowing access to intermediate data may provide already correctly formatted data or assist in reducing unnecessary and extraneous transformations. Along with potentially increased compatibility, labyrinthic modules may accept multiple inputs allowing the same module to be utilized for multiple similar functions and adjustment to the most appropriate available data sources.

Hardware must be adjustable to changing domain and task demands. Embodied SCAs, possessing comprehensive knowledge of their hardware, may understand these demands and adjust settings and modes in real time to meet active sensing or actuation requirements. A sense of embodiment may enable a SCA to understand how its hardware configuration interacts with its software based interfaces and adjust those modules as needed to best manipulate the environment and direct sensors to acquire and process the most pertinent domain information. As domains, tasks, and SCA hardware con-

figurations become more complex and intricate, emergent interaction increases and the SCA must be able to understand these interactions and predict them.

Commonsense may aid generation of predictions via defaults and expectations regarding the current situation. These concepts may help SCAs understand and predict emergent interactions and effects resulting from changing modalities, via hardware or software modifications, which may generate several benefits. SCAs may be better equipped to weigh the benefits and costs of making configuration modifications. While complex, commonsense may provide heuristics for making selections and enable reasoning over a diverse array of "Ways to Think" [Minsky, 2006]. This wide array of "Ways to Think" may minimize configuration changes benefiting one form of processing but unknowingly hindering others. Commonsense may also assist in directing attention. As domains and tasks change, SCAs should stay flexible to variations in sensor reading availability, accuracy, and applicability. Flexibility allows SCAs to utilize the most appropriate and effective forms of available information, instead of following possibly inappropriate schemas.

SCAs operating within complex and dynamic domains will experience changing demands on both sensing and inference. SCAs that are capable of dynamically modifying data and control flow may be better equipped to handle current demands and maintain higher SA levels.

2.4   Diverse capabilities and knowledge

Future SCA tasks may require increased capabilities. Challenges may be diverse, requiring equally broad but effectively managed skill sets. These abilities may be achieved through a number of different means.

Unless tailored to a specific, narrow domain, SCA demands may be diverse. Unified intelligence theories may lack the requisite diversity to realize SA due to constrained capabilities [Minsky et al., 2004], but non-fixed designs may permit diverse inferencing and data representations. SCAs may be capable of incorporating more appropriate and applicable inference methods for a domain while utilizing a greater array of capabilities. Architectures provide certain benefits while forcing certain restrictions, but non-fixed designs may possess fewer restrictions as they require fewer assumptions regarding methods and data representations. Care is required to ensure that a large percentage of the inference methods can execute simultaneously and not require constant switching between individually limiting modes. Non-fixed designs may further assist SCAs by allowing for the inclusion and simultaneous activation of multiple instances of the same inference methods.

Support for an infinite number of inference methods and data representations may only provide minimal benefit if they cannot be controlled. Metacognition may provide a portion of this moderation by maintaining activation of the most appropriate processes. Metacognition may also predict the performance of individual inference methods and select the most appropriate ones. After selection, processes must be monitored for failures and performance. If errors occur, inference methods may require assistance or different inference methods may be activated. Even without errors, processes must be monitored to ensure progress towards current goals and tasks while remaining within the SCA's set of values. Goal progress may be estimated to ensure

optimality of activated process selection. Each process requires some form of performance metric, which may determine when the process should be replaced or modified. The metacognitive capabilities performing this monitoring must also determine causes for performance gains and losses. Some processes may detect abnormal performance levels resulting from an abnormal situation, and decisions over the actively selected processes must take this into account. Metacognition can select the most appropriate process for each current subtask, while ensuring that the selected processes function additively and do not degrade performance. Metacognition may require locally suboptimal performance in order to achieve globally optimal performance and may also adjust modes or operating parameters within the system.

Optimizing process selection with metacognition may be difficult, but hierarchical designs may simplify the task via partitioning, reducing the problem into manageable sub-problems. Benefiting from reasoning over different levels within a hierarchy, metacognitive processes may operate at the optimal levels of detail and time scales. Metacognition may be added in a hierarchical fashion providing numerous benefits, including problem space partitioning.

Combining binding with metacognition and hierarchical design may simplify the problem. Binding provides interface protocols dictating how potentially radically different inference methods and data representations interact. This may simplify selection of active processes as it increases the independence of each process. Further, binding may lessen context switching costs. When internal data representations can be exported in some form or are no longer needed, binding can assist in changing currently active inference methods since the new methods can interface to the current binding proxies, as if the process had been running the entire time.

Context switching costs may be reduced via labyrinthic design. Internal representations may be made globally available and temporarily archived. Once a new process is activated, the archived data may be imported. Formatting the archived data may be required, but may be more efficient than starting from scratch. Furthermore, if labyrinthic modules can accept multiple input forms, then they may be useable by multiple forms of processing [Sloman, 1989].

Additional data may consist of hardware status, as embodied SCAs may possess heightened sensor and actuator understanding, allowing for additional inferencing. Along with hardware reasoning, inferencing may consider hardware, firmware, and software interactions to increase awareness of internal state and realize finer grained sensor control, thus improving sensing and data interpretation.

Reasoning over data from diverse hardware sources may require numerous realms of thought. Temporal, spatial, or logical inferencing may be required to fully utilize incoming data. Commonsense may provide reasoning about physics, dominion, and many other areas required for operating in complex domains. Depending on the implementation and realization, commonsense may allow reasoning with multiple realms of thought simultaneously. This parallel processing may provide a greater understanding of the environment.

SCAs operating within complex and dynamic domains may require multimodal sensing and inference. As architectures increase in the number and diversity of supported capabilities, SCAs may be able to gain a broader understanding of their environment and realize improved SA.

2.5 Inter-module communication

Unless designed as a monolithic system, SCAs possess multiple modules that must communicate. Improving communication may consist of removing unnecessary communications and improving communicating efficiency and quality while minimizing processing requirements.

Metacognition may generate increased domain understanding by optimally selecting communication modes. As tasks change, different inference methods may produce greater efficiency and output quality. Selecting modules for a given situation may minimize excessive information processing while increasing the flow of critical information.

Metacognition may be more effective when combined with a non-fixed design and a diverse array of inference methods and data representations. This abundance of possible configurations may result in improved communication quality. Often while executing a particular task, the entire set of available functionality may not be required. Unnecessary modules may be deactivated or assigned a low priority either at design time or runtime. Supporting modules may also be changed or replaced depending on the information requirements of the primary modules being utilized. Proper selection of supporting modules (modules that primarily post-process information) may allow for more concise information and improved information filtering.

Along with deactivating modules, SCAs may further improve inter-module communication via labyrinthine designs. Providing intermediate data and accepting multiple input forms, labyrinthic modules may minimize data formatting while communicating normally unavailable information.

Combining labyrinthic designs with binding may aid inter-module communication. Data pointers allow modules to acquire low level data without constant, expensive information forwarding. As higher level constructs are iteratively constructed, data forwarding may be further suppressed. Combining data pointers, higher level constructs, and level-banding may permit inter-module communication across multiple hierarchy levels. Low level sensing can monitor the environment while providing slower, higher level processes with stable symbolic constructs. Once deliberation has finished, additional lower level modules may carry out actions. These modules can query previous low level modules, via data pointers, for current information. As long as high level constructs do not require modification, higher level module decisions may be combined with low level sensing information, enabling efficient and effective actuation on the environment or self [Hawes et al., 2007].

Binding and labyrinthic designs may further improve communication when paired with a hierarchical design. The parent-child relationships in a hierarchy may minimize the number of modules affected by a particular mode of inter-module communication. These relations may further isolate communication channels. Depending on the communication mechanisms, the parent and child may share information without bombarding others with unnecessary data.

Understanding sensor and actuator interactions may improve inter-module communication as this knowledge may help acquire and filter data. Many architectures not inherently supporting embodiment only allow environmental information to be available to cognitive processes via restrictive interfaces often treated as "black boxes". These

interfaces often require unnecessary formatting and may strip critical information from incoming data. Embodiment can also filter sensor readings. An understanding of sensor operation and the environment's status may reduce the need for sensing, especially if an error tolerance for a particular task exists. A SCA can begin to sense on demand by understanding how sensing is performed instead of constantly sensing. While only applicable for certain domains and sensing, this understanding may reduce the volume of data processing.

Akin to embodiment aiding search within the environment, commonsense may direct attention within the environment and the SCA's internals. This directed attention may minimize irrelevant sensor readings and provide data of increased importance to higher level processes. Commonsense's defaults and expectations may aid selection and configuration of mental models. Improved mental model selection may permit better understanding of communication requirements enabling improved self-reconfiguration.

SCAs utilizing a hierarchical or modular design may improve error detection and prevention via binding, which often utilizes probabilistic inference to combine lower and higher level constructs. When these constructs are associated with high uncertainty, or cannot be generated, then there may be an increased likelihood of erroneous or incompatible data within the system. SCAs can then direct attention to lower level processes, increasing the priority of the affected systems. The resultant higher level constructs may also modify belief levels. As bindings are generated, lower level processes, or higher level processes that feedback into lower level inference methods, can glean additional information from the bindings and modify their belief levels or their output values. This strategy may increase the ability for multiple inference forms and data representations to assist one another, providing comparisons and alternate results.

As SCAs grow in complexity, the volume of multimodal data may increase. In order for SCAs to capitalize on this data, inter-module communication must be present to ensure that the SCA maintains a broad environmental understanding that may increase SA breadth.

2.6   Storing volumes of data and purging stale data

SCAs seeking SA need to ensure that their knowledge of self and the environment remains current. Stale data prevents effective operation within real-time dynamic environments, as opportunities will be missed and dangers may be unavoided. Each proposed design method can function together to acquire current information and purge stale data.

Environmental data may change rapidly, but not all changes occur at similar frequencies. SCAs utilizing non-fixed designs may adjust to global and local environmental fluctuations. A naive approach of setting all modules operating to maximum frequency may not improve performance, as information filtering may degrade. Processes should operate at their natural speed. Conversely, sensing and inference cannot occur too slowly or data may become stale and erroneous. Data may be amenable to replacement with incoming sensor readings, but other data may require decay. Data structures and algorithms may require selection based on support for information decay, as not all methods provide this capability. The time since some datum was recorded

may provide a decay metric, along with time since last use, for selecting which data points to remove or decay.

While time and ranking can be used, domains and tasks may require advanced decay routines. Metacognitive capabilities may allow for dynamic deletion strategies by identifying stale and erroneous information. With metacognition, deletion of information does not have to be based solely on time and data vacancy, but also importance and ease of updating and reacquiring. The ability to direct attention may impact the SCA's ability to reacquire information. Additionally, metacognitive reasoning can determine which information should be dropped or requires immediate updating. These decisions may be dependent on information priorities resulting from the SCA's goals, tasks, and status. Since SCAs may be unable to maintain a perfect environmental and self understanding, satificing may determine when the SCA's knowledge is good enough. This quality metric may include a breadth of information coverage and depth within critical areas of cognitive processing. Information decay and updating may be influenced by errors. Metacognitive abilities may determine if the present errors resulted from an execution or an inference error, and to determine which component is responsible for the unsatisfactory execution.

Metacognition may be most effective when paired with hierarchies as dynamic reconfiguration may reduce stale information and promote data validity. Coupled with a non-fixed design, metacognitive control may increase update frequencies of components, temporarily inhibit poorly working components, and increase the priority of critical components. Hierarchies partition components by function, and levels of abstraction, allowing for fine tuning of data management and performance. Level-banding may reduce stale information, since the flow and detail level of data traversing the hierarchy may be modified within some bounds, providing "just in time" modulating of information filtering.

Binding assists in generating level-banding; ensuring minimization of stale data, especially when hierarchical and modular designs are utilized. Repeated copying and transferring volumes of data degrades performance by congesting data busses. Further, synchronization of copied data can be difficult. Binding's data pointers may alleviate many of these problems by obviating the need for copying large volumes of data and maintaining multiple copies. When higher level processes require information, they follow the pointer. Combined with level-banding, higher level processes can work with data at an infrequently changing higher level, while other processes can utilize the freshest data without continuously interrupting and restarting higher level processes.

Further reducing stale data, embodied designs can sense instead of store data. SCAs may not need to sense the entire environment due to sensor and actuator understanding, minimizing generation of rapidly stale data. Instead, SCAs may know how to acquire requisite data and the latencies within the acquisition processes. If these costs are acceptable, then SCAs can acquire data on demand. Furthermore, by better understanding sensors and actuators, SCAs can better detect and filter erroneous data. Filtering may have a large impact if the emergent effects resulting from interactions between the environment and hardware are understood.

Along with sensing on demand, commonsense reasoning may improve attention targeting to important data sources. Generation of defaults and preconceptions may allow better predictions of what data is required and how to be cognizant of it. Thus,

14

stale data may be minimized and fresh data may be more current. Defaults and preconceptions also provide prototypical data comparing current data against, and potentially filtering out, erroneous information. Finally, commonsense may assist in mental model selection. Improved selection may increase processing speed, improving high level construct production and preventing data buffer clogging.

As SCAs grow in complexity, data processing requirements will increase. When SCAs are able to effectively utilize volumes of data while purging stale data, SCAs may be able to maintain greater SA and minimize falling behind the dynamism of the environment.

3 Conclusion

Future SCAs will require an understanding of the current domain state and effects of actions to function in complex, dynamic domains. Bestowing SCAs with SA may improve understanding of dynamism, complexity, and action selection and execution. This paper presented an initial exploration into required features for bestowing SCAs with SA. While difficult to implement, design mechanisms were also presented that may ease the task. If SCAs achieve SA, robotics and other fields utilizing SCAs may be one step closer to attaining human level intelligence and performance.

References

[Anderson et al., 2004] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S. A., Lebiere, C., and Qin, Y. (2004). An integrated theory of mind. *Psychological Review*, 111(4):1036–1060.

[Arkin and Balch, 1997] Arkin, R. C. and Balch, T. (1997). AuRA: Principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2/3):175–188.

[Connell, 1992] Connell, J. H. (1992). SSS: A hybrid architecture applied to robot navigation. In *Proceedings of the 1992 IEEE Conference on Robotics and Automation*, pages 2719–2724.

[Cox, 2005] Cox, M. T. (2005). Metacognition in computation: A selected research review. In *Artificial Intelligence*, volume 169, pages 104–141. Elsevier B. V.

[Endsley, 1988] Endsley, M. R. (1988). Situation awareness global assessment technique (SAGAT). In *Proceedings of the National Aerospace and Electronics Conference*, pages 789–795.

[Feil-Seifer and Matarić, 2008] Feil-Seifer, D. J. and Matarić, M. J. (2008). Human-robot interaction. In *Encyclopedia of Complexity and System Science*. Springer.

[Flach, 1995] Flach, J. M. (1995). Situation awareness: Proceed with caution. In *Human Factors*, volume 37, pages 149–157.

[Hawes et al., 2007] Hawes, N., Sloman, A., Wyatt, J., Zillich, M., Jacobsson, H., Kruijff, G.-J. M., Brenner, M., Berginc, G., and Skocaj, D. (2007). Towards an integrated robot with multiple cognitive functions. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 1548–53, Menlo Park, California. AAAI Press, AAAI Press.

[Lehman et al., 2006] Lehman, J. F., Laird, J., and Rosenbloom, P. (2006). A gentle introduction to soar, an architecture for human cognition: 2006 update. Technical report, University of Michigan.

[Liu and Singh, 2004] Liu, H. and Singh, P. (2004). ConceptNet - a practical commonsense reasoning tool-kit. *BT Technology Journal*, 22(4):211–226.

[Minsky, 1986] Minsky, M. (1986). *The Society of Mind*. Simon & Schuster.

[Minsky, 2006] Minsky, M. (2006). *The Emotion Machine*. Simon & Schuster.

[Minsky et al., 2004] Minsky, M., Singh, P., and Sloman, A. (2004). The St. Thomas common sense symposium: Designing architectures for human-level intelligence. In *AI Magazine*, volume 25, pages 113–124.

[Sloman, 1989] Sloman, A. (1989). On designing a visual system. *Journal of Experimental and Theoretical AI*, pages 289–337.