

# THE DESIGN, EDUCATION AND EVOLUTION OF A ROBOTIC BABY

A Dissertation  
Presented to  
The Academic Faculty

By

Hanqing Zhu

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in Robotics in the  
School of Aerospace Engineering

Georgia Institute of Technology

December 2022

Copyright © Hanqing Zhu 2022

# THE DESIGN, EDUCATION AND EVOLUTION OF A ROBOTIC BABY

Thesis committee:

Dr. Eric Feron, Advisor  
Computer, Electrical and Mathematical  
Sciences and Engineering  
*King Abdullah University of Science and  
Technology*

Dr. Kyriakos Vamvoudakis, Co-Advisor  
School of Aerospace Engineering  
*Georgia Institute of Technology*

Dr. Dimitri Mavris  
School of Aerospace Engineering  
*Georgia Institute of Technology*

Dr. Jerome Hugues  
Software Engineering Institute  
*Carnegie Mellon University*

Dr. Emmanuel Roche  
*Clover.AI*

Date approved: December 7, 2022

If a machine can think, it might think more intelligently than we do, and then where  
should we be?

*Alan Turing*

To Mom and Dad

致父母

## ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Eric Feron. Eric has been a wonderful advisor throughout my PhD journey. I have fully enjoyed and exploited the freedom and liberty that Eric has provided, along with countless ideas and opportunities along the way. Had I chosen any other professor with another engagement style, I would not have achieved all the things that I have done so far. Furthermore, I have learned so much from Eric, with knowledge and lessons for life outside of academia, on how to be a man of culture and responsibility with a pair of critical eyes and a sympathizing mind, which I appreciate more than ever.

Next, I would like to thank my PhD committee members for their service: Dr. Kyriakos Vamvoudakis, Dr. Dimitri Mavris, Dr. Jerome Hugues and Dr. Emmanuel Roche. I appreciate their time and effort in providing constructive feedback and comments for better shaping the creation of this thesis. I would also like to thank Dr. Joseph Saleh, who was part of my committee and was super excited about my thesis proposal. Unfortunately, Dr. Joseph Saleh has passed away before the completion of the thesis but his excitement and support will be forever remembered.

Moreover, I would like to express my appreciation to those who have crossed road with me along my PhD journey, whether we worked hard together or played harder together. Their presence has meant a lot to me and has shaped me into who I am today, thanks to, you know, causal determinism. I would also like to acknowledge all the sacrifices that I have made during my PhD, such as the things that could have been done, the trips that could have been made and those who I could have spent more time with. The opportunity cost of finishing the PhD journey will not be taken lightly.

Last but not least, I would like to thank my parents who have supported me with unconditional love, without which I would not have achieved what I have done during my PhD journey.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	v
<b>List of Tables</b> . . . . .	x
<b>List of Figures</b> . . . . .	xi
<b>Summary</b> . . . . .	xiii
<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Contributions . . . . .	4
1.2 Thesis Outline . . . . .	5
<b>Chapter 2: Literature Review</b> . . . . .	6
2.1 Cognitive Architecture . . . . .	6
2.2 Knowledge Representation . . . . .	7
2.3 Natural Language Programming and Semantic Parsing . . . . .	8
<b>Chapter 3: A Robotic Baby</b> . . . . .	9
3.1 Inspiration from Nature . . . . .	9
3.2 Concept of Operation . . . . .	10
3.3 Formal Definition of a Robotic Baby . . . . .	12
3.4 Philosophical Inspirations and Considerations . . . . .	13

3.4.1	On Self Existence . . . . .	13
3.4.2	On Causal Determinism and Free Will . . . . .	13
3.4.3	On Adaptation to the Unpredictable World . . . . .	14
3.5	Scope and Limitation . . . . .	14
3.6	Systems Engineering Design Approach . . . . .	15
<b>Chapter 4:</b>	<b>System Requirements Flowdown . . . . .</b>	<b>17</b>
4.1	The Necessity of Requirements Flowdown . . . . .	17
4.2	The Requirements Flowdown for the Robotic Baby Design . . . . .	17
<b>Chapter 5:</b>	<b>System Architecture . . . . .</b>	<b>21</b>
5.1	Language Input and Output . . . . .	21
5.2	Natural Language Processing . . . . .	22
5.3	System Management . . . . .	26
5.4	Long-term Memory . . . . .	26
5.5	Short-term Memory and Activation . . . . .	31
5.6	System States . . . . .	33
5.7	Physical Input and Output . . . . .	34
<b>Chapter 6:</b>	<b>Sub-System Implementation and Initialization . . . . .</b>	<b>35</b>
6.1	Linguistic Knowledge Initialization . . . . .	36
6.2	Action Space and Activation Initialization . . . . .	37
<b>Chapter 7:</b>	<b>System Verification and Validation . . . . .</b>	<b>39</b>
7.1	Acquisition and Semantic Parsing of English and Chinese . . . . .	39

7.2	Bilingualism, Multilingualism and Code-Switching . . . . .	42
7.3	Natural Language Grounding to Semantics and Actions . . . . .	43
7.4	Natural Language Reinforcement Learning . . . . .	44
7.5	Natural Language Programming . . . . .	47
7.6	Introspection and Explainability . . . . .	50
<b>Chapter 8: Robotic Baby Education and Evolution . . . . .</b>		<b>54</b>
8.1	One Robotic Baby with Distributed Embodiment . . . . .	54
8.2	Natural Language Interaction Between Two Robotic Babies . . . . .	56
8.3	Brain Merge Between Two Robotic Babies . . . . .	57
8.4	Robotic Baby Evolution with Knowledge Inheritance . . . . .	60
<b>Chapter 9: Discussion . . . . .</b>		<b>63</b>
9.1	Scalability . . . . .	63
9.2	Adoption and Application . . . . .	64
9.3	Limitations and Fundamental Research Questions . . . . .	65
9.4	Lifelong Learning . . . . .	66
9.5	The Zone of No Return . . . . .	67
<b>Chapter 10: Conclusion and Future Work . . . . .</b>		<b>69</b>
<b>Appendices . . . . .</b>		<b>71</b>
	Appendix A: Sample Knowledge Base of Vocabulary and Part-of-Speech . . . . .	72
	Appendix B: Sample Knowledge Base of Grammar Production Rule . . . . .	74
	Appendix C: Example Execution of BabyParse . . . . .	75



Appendix D: Example Execution of BabyActivate . . . . .	79
Appendix E: Scalability Performance Profiling . . . . .	81
<b>References . . . . .</b>	<b>83</b>

## LIST OF TABLES

7.1	Action Nodes for Language Acquisition in English . . . . .	40
7.2	Action Nodes for Language Acquisition in Chinese . . . . .	40
7.3	Action Nodes for Semantic Mapping in English and Chinese . . . . .	43
7.4	Action Nodes for Edge Weight Update in English and Chinese . . . . .	45
7.5	Action Nodes for Natural Language Programming in English and Chinese .	48
7.6	Dummy Action Program Instructions in English and Chinese . . . . .	49
7.7	Action Nodes for System Introspection in English and Chinese . . . . .	51

## LIST OF FIGURES

3.1	The concept of operation of a robotic baby. . . . .	10
3.2	The mode of operation of a robotic baby with distributed embodiment along with a central “brain”. . . . .	11
3.3	The mode of operation of the robotic baby with a multi-agent setup, where each robotic baby has its own “brain”. . . . .	12
3.4	The V-model of systems engineering [64]. . . . .	15
5.1	The proposed system architecture, <i>Baby</i> , for the robotic baby. . . . .	22
5.2	The requirement satisfaction traceability matrix for the <i>Baby</i> architecture design. . . . .	22
5.3	Knowledge representation for the word ‘bank’ in a subgraph. . . . .	28
5.4	Knowledge representation for NP and VP production rules in a subgraph. . . . .	29
5.5	The default activation of the action node ‘go to point’. . . . .	29
5.6	A snapshot of the knowledge activation graph containing all types of nodes and edges at a point in time. . . . .	30
7.1	The parse tree and dependency graph for “Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo”. . . . .	41
7.2	The parse tree for the code switching sentence “We should 过新年”. . . . .	43
7.3	Demonstration for natural language reinforcement learning with the Robo-tarium integration. . . . .	46
7.4	Demonstration for natural language programming with dummy actions. . . . .	48

7.5	Demonstration for natural language programming with the Robotarium integration. . . . .	50
7.6	LifeQueue for the “go shopping” demonstration, with primitive and complex actions. . . . .	52
7.7	Activation graph for “go shopping” from the syntactic nodes to the action node, with the semantic dependency graph boxed in yellow. Activated nodes are filled with colors. . . . .	52
8.1	Demonstration for educating one robotic baby with distributed embodiment with the Robotarium integration. . . . .	55
8.2	Demonstration for education with natural language interactions between two robotic babies with the Robotarium integration. . . . .	57
8.3	Demonstration for the Graph merge operation between two robotic babies with the Robotarium integration. . . . .	59
8.4	Knowledge of Emma and Zebra about colors and languages before and after Zebra merges with Emma’s Graph. . . . .	60
8.5	The evolution comparison between robotic babies, with direct knowledge Graph inheritance at birth, and human beings, with genetic inheritance at birth and knowledge acquired in the future. . . . .	62
9.1	The projected advancement in the research and development of the robotic baby. . . . .	68
E.1	Processing time performance profiling for the NLP and Activate processes with 100 repeating runs. . . . .	81

## SUMMARY

Inspired by Alan Turing's idea of a child machine, I introduce the formal definition of a robotic baby, an integrated system with minimal world knowledge at birth, capable of learning incrementally and interactively, and adapting to the world. Within the definition, fundamental capabilities and system characteristics of the robotic baby are identified and presented as the system-level requirements. As a minimal viable prototype, the *Baby* architecture is proposed with a systems engineering design approach to satisfy the system-level requirements, which has been verified and validated with simulations and experiments on a robotic system. The capabilities of the robotic baby are demonstrated in natural language acquisition and semantic parsing in English and Chinese, as well as in natural language grounding, natural language reinforcement learning, natural language programming and system introspection for explainability. Furthermore, the education and evolution of the robotic baby are illustrated with real-world robotic demonstrations. Inspired by the genetic inheritance in human beings, knowledge inheritance in robotic babies and its benefits regarding evolution are discussed.

## **CHAPTER 1**

### **INTRODUCTION**

The idea starts with a baby. In 1950, Alan Turing had a rough idea of a child machine, which, according to his description, could be the starting point for an intelligent machine to pass his test if under an appropriate course of education [1]. Alan Turing's idea of a child machine could be developed as follows. Let's suppose that the intelligent machine passing his test has been created and it, by definition, could act just like a normal human being. With such construction, the intelligent machine has to start from somewhere, just like all human beings starting from a baby. Indeed, the intelligent machine has to start from being a baby as well. How should I bring such a baby machine into life from Turing's rough idea?

For a designer of such an intelligent machine, it is natural to look into the abstractions of what a human baby at birth is and how a human baby grows. Unfortunately, there has not been a definitive description of neither what a human baby at birth is nor how a baby grows in cognitive capabilities, despite many proposed theories in aspects of child cognitive development and psychology, for example, by Jean Piaget [2, 3, 4, 5] and Jean Mandler [6, 7, 8]. The proposed theories could serve as great sources for inspiration. Yet, it has come to the conclusion that they could not be utilized for detailed mechanism and structure design of the baby intelligent machine due to their high-level descriptive nature, not good enough to explain human mental capacity according to Marvin Minsky [9]. This realization sparks an important question: for a baby intelligent machine, does it have to have the human flavor? In other words, I would like to entertain the idea of building a baby intelligent machine without the human characteristics that are specific to the nature of human beings.

The departure from the human-inspired path for a baby machine to get to intelligence shifts the focus of the design of the baby machine from imitating the human process to

identifying necessary capabilities as the building blocks for higher level intelligent behaviors. Recent works on the road map for human-like AI development have highlighted a wide range of capabilities, including but not limited to, communication, language acquisition, learning, intuitive physics and psychology, and reasoning [10, 11]. For the purpose of designing a baby machine not mimicking the exact human counterpart, I first exclude the capabilities specific to the nature of the human body, since our starting point is a machine, a transparent mechanical system with orders, unlike a human baby, a bio-molecular black-box with chaos at the beginning of life. Then, I prioritize the fundamental capabilities, which are hypothesized to be the prerequisite capabilities for high level cognitive capabilities and skills.

Among the prioritized fundamental capabilities, an important one is language acquisition and understanding. In general terms, language is just a form of a patterned signal. Language is seen not only as a communication tool, enriching the abstract representation, for interacting with external world, but also a tool for programming the internal mind as a control system interfacing with grounded mental states [12]. Another one to highlight is the construction and execution of complex actions programmed with primitive actions. This classic idea of compositionality in terms of programming has been proposed, widely studied and labeled as “schemata” [2], “programmes” [13], “skill sets” [14] and “subroutines” [15], which could be a fundamental capability for constructing higher level skills for the baby machine.

In this work, I introduce a formal definition of a *robotic baby*, an integrated system capable of learning incrementally and interactively, and adapting to the world, starting with minimal world knowledge. The formal definition of a robotic baby contains the high-level system requirements, governing the capabilities of the robotic baby. For the scope of this thesis, the *Baby* architecture is designed, with a systems engineering design approach, to meet the high-level requirements of the robotic baby. The robotic baby with the *Baby* architecture, as an integrated system, possesses some unique capabilities and characteristics

as follows.

1. *Natural language acquisition:* The robotic baby is capable of acquiring linguistic knowledge of English and Chinese, in terms of vocabulary words, parts of speech and grammar production rules, with instructional commands in English and Chinese.
2. *Natural language semantic parsing:* The robotic baby is capable of parsing grammatical inputs, modeled with context-free grammar, in English and Chinese or a mixture of both known as code-switching, with limitations on specific types of ambiguity.
3. *Natural language grounding to semantics and actions:* The robotic baby is capable of grounding natural language in English and Chinese to semantics and actions, taking into account the variance in natural language expressions, with instructional commands in English and Chinese.
4. *Natural language reinforcement learning:* The robotic baby is capable of taking reward signals in English and Chinese and improve its behavior accordingly.
5. *Natural language programming:* The robotic baby is capable of constructing complex actions in the form of a program with actions as building blocks, with instructional commands in English and Chinese.
6. *System interpretability and explainability:* The robotic baby utilizes explicit knowledge representation, and transparent and traceable rule-based information processing algorithms for complete interpretability and explainability.
7. *System introspection:* The robotic baby is capable of recalling its action history and activation graph for executed actions, when commanded in English and Chinese.
8. *Knowledge transfer with natural language interactions:* The robotic baby is capable of acquiring knowledge via natural language interactions with both a human parent and a peer robotic baby.



9. *Knowledge inheritance*: The robotic baby is capable of directly inheriting knowledge from a peer, including both the content and the knowledge inference structure via a brain merge operation.

According to the systems engineering procedures, the system-level verification and validation for the robotic baby have been conducted with experiments in both simulation and real world settings to ensure that the system-level requirements have indeed been satisfied.

## 1.1 Contributions

The contribution of this thesis is four-fold as follows.

- a) *Robotic baby definition*: the clear definition and focus of a robotic baby, in terms of the system requirements, governing the fundamental set of capabilities, along with philosophical considerations. This is a firm step moving forward from the rough ideas, road maps, speculations and future projections in the development of a baby machine.
- b) *Baby architecture design*: the overall design of the *Baby* architecture, including all algorithms, processes and structures, and the associated capabilities showcased in demonstrations.
- c) *Systems engineering design approach*: the application of the systems engineering design principles, from system requirements to proper system verification and validation, to the design process of a complex integrated system in AI.
- d) *Robotic baby education and evolution*: the demonstration of the education of the robotic baby with a distributed robotic embodiment via natural language interactions, and the evolution of the robotic baby with knowledge inheritance via brain merge, inspired by genetic inheritance in the human species.

## 1.2 Thesis Outline

This thesis is organized as follows. Chapter 2 describes relevant works in related fields for applicability in the design of the robotic baby. Chapter 3 lays out the fundamentals including the formal definition of and philosophical considerations about a robotic baby. Chapter 4 presents the system design requirements flowdown. Chapter 5 presents the *Baby* architecture. Chapter 6 describes the subsystem initialization. Chapter 7 presents the system verification and validation of the robotic baby design by means of experiments. Chapter 8 showcases the education and evolution of the robotic baby. Chapter 9 presents the discussions on the design of a robotic baby. Chapter 10 concludes this thesis with future work.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Cognitive Architecture**

The cognitive architecture community, dating back to the 1970s, has been proposing new cognitive architectures, trying to model the human cognitive behaviors. Allen Newell once criticized the task-specific models that tackle a fragment of the overall cognitive modeling, indicating that it was hard to make cumulative progress with the fragmented approach [16]. As an example of overall cognitive modeling, the Adaptive Control of Thought Rational (ACT-R) cognitive architecture, developed by John Robert Anderson, is one of the earliest models aiming at understanding the human mind. It aims to define the underlying cognitive and perceptual operations in the human mind, using a central rule-based procedural module, which controls the overall system behavior along with interactions with other modules [17]. Another well-known architecture is Soar, developed by John Laird, Allen Newell, and Paul Rosenbloom, with a fixed structure containing modular blocks for perception, motor functions, working memory and long-term memory [18, 19]. There are constant interactions among the modular blocks with complex behaviors arising from the interactions within cognitive cycles. Recent architectures, such as Sigma by Paul Rosenbloom [20], try to combine the traditional modular structure and modern techniques in graphical models and deep learning. With more than 50 architectures, the survey work [21] tries to find a common ground among major cognitive architectures, known as the standard model of the mind. However, even with a summary of all the commonly shared features, it is still unclear on how one shall move further regarding building an integrated learning system. The cognitive architecture community is in this constant process of trial and error, hoping a particular architecture design models some human cognitive behaviors well. Other challenges

that the current cognitive architectures face include the lack of communication about its inner process and its decisions, and the lack of liveliness in terms of a natural internal drive [22]. Compared to the existing cognitive architectures with built in high-level capabilities, our robotic baby architecture, while looking similar in structure with modular components, addresses an entirely different research question, regarding the design and engineering of the minimal capabilities of a robotic baby at birth for future growth and education.

## **2.2 Knowledge Representation**

Knowledge representation is an important aspect of the robotic baby architecture for the designer to consider during the design process. One line of related research is to explicitly define the words, their syntax and semantic properties and their relationships to other words. One of the most notable lexical knowledge bases, WordNet [23], contains more than 150,000 English words and relationships like synonymy among words. Beside its significant size, there have been disagreements on the senses of words [24] in WordNet. Other than WordNet, there are other databases designed for various purposes, such as VerbNet [25], PropBank [26], and FrameNet [27] for semantic role labeling. These knowledge bases provide a good starting point for inspiration for the design of the knowledge base for the robotic baby. They are, by no means, sufficient for the design of the robotic baby, even though the amount of knowledge stored in these knowledge bases is large. What they lack of are the mechanism for accumulating knowledge and the corresponding mechanism for effective retrieval of knowledge. They are not designed as part of a system with system requirements in the first place.

Another related line of research concerns the learning of word representations and embeddings, which are in a distributed and implicit fashion, such as Word2Vec [28], GloVe [29] and ELMo [30]. These distributed word embeddings are trained on large annotated texts so that the contextual information can be modeled in a distributed fashion [31]. Another related example is the dependency-based word embeddings [32], in which words are

modeled after its dependency context. Nevertheless, all mentioned distributed representations are static in the sense that they are trained on a static training set. It is unclear, due to their implicit nature, how the word embeddings should change in a dynamic setting with the accumulation of new vocabularies in life-long learning, which is critical for the robotic baby.

### **2.3 Natural Language Programming and Semantic Parsing**

Natural language has long been discussed as a communication tool between human and machine [33]. Edsger Dijkstra once argued that it would be difficult to program machines in a natural language due to ambiguity and the lack of formality in a natural language [34]. Early attempts in natural language programming included systems with natural language commands for matrix manipulation [35]. Other applications of natural language programming could be seen in interfaces for robotic applications [36, 37, 38, 39, 40] and smart personal assistant [41]. Natural language programming is also utilized for program synthesis, in which natural language specifications are given in order to construct programs, with rule-based approaches [42] or by training program synthesizers on data with machine learning techniques [43, 44]. The rule-based approaches rely on set of pre-defined domain-specific rules, which might not be applicable to the robotic baby design. The machine learning techniques, on the other hand, rely on a fix set of training data, which might not be suitable for the purpose of a robotic baby either. Another line of related research involves the interaction between human and machine [45, 46, 47, 48], in which the live interaction serves as a mechanism for clarification and information supply to robots, which could be utilized for the design of a robotic baby.

## CHAPTER 3

### A ROBOTIC BABY

#### 3.1 Inspiration from Nature

A baby is often associated with the first stage of human growth after birth. What exactly is a human baby? At birth, the initial condition of a human baby,  $h_0$ , is set. In principle, the initial condition of a human baby consists of all the states of the physical body of the human baby at birth. Note that the initial condition of a human baby carries minimal information about the external world, in the sense that the initial states of the body are mainly inherited from the parents. In addition to the initial condition, the initial governing process of the human baby,  $f_0(\cdot; \theta_0)$ , is set. The governing process at time  $t$ , presented in Eq. (3.1), can be abstracted as a function with decision-making parameters  $\theta_t \subset h_t$  that takes in the internal states  $h_t$  and the external world states  $x_t$  and produces  $h_{t+1}$ .

$$h_{t+1} = f_t(x_t, h_t; \theta_t) \quad (3.1)$$

Since the cognitive development of infants is not understood to a complete extent [9] but with proposed theories [4, 49], the initial governing process  $f_0$  of a human baby cannot be clearly defined. Nonetheless, the computing framework in Eq. (3.1) with the initial condition and the governing process could provide the inspiration for the design of a robotic baby, since the designer is able to pick a combination of the initial condition and the governing process for the robotic baby to roll out its life [1].

### 3.2 Concept of Operation

The robotic baby is designed to be educated and operational in a similar fashion to the case of a human baby. The concept of operation of a robotic baby is presented in Figure 3.1.

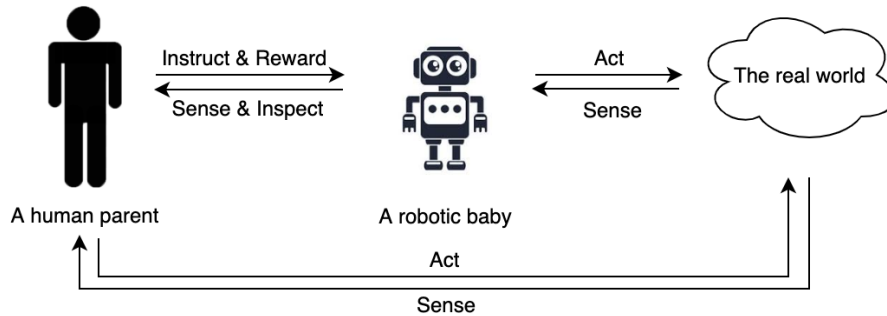


Figure 3.1: The concept of operation of a robotic baby.

The robotic baby is designed to act and sense the world, similar to what a human baby can do. There also needs to be at least one parent, whether a human or a robot, present during the growth of a robotic baby, providing instructions and reward signals to the robotic baby while sensing and inspecting what the robotic baby does. Notice that it is expected from the parent to not only be able to sense what the robotic baby does but also be able to inspect internally the inner working of the robotic baby. This expectation from a parent differs from the human case since it is, as of the writing of the thesis, not possible to fully understand the inner workings of a human baby and it is beneficial for the parent to take advantage of the transparency of the inner working of a robotic baby to better instruct and educate the robotic baby. Meanwhile, the parent is also expected to act and sense the world around a robotic baby, especially during the early ages of the robotic baby for strategically exposing the external world to the robotic baby and facilitating the learning of the fundamentals, such as a natural language.

Beside the single robotic baby operation, there are also modes of operation which involve multiple robotic babies. One mode of operation is distributed embodiment. In this mode of operation, there are multiple copies of the same robotic baby operating in a dis-

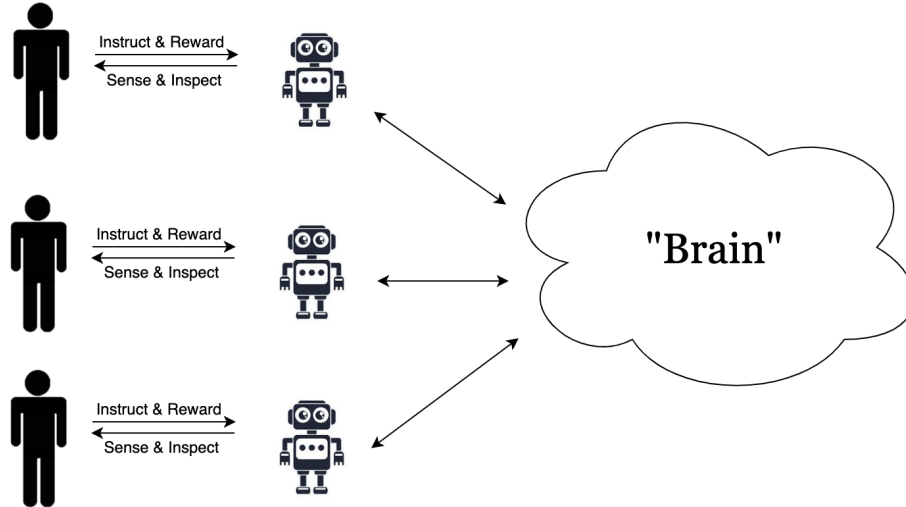


Figure 3.2: The mode of operation of a robotic baby with distributed embodiment along with a central “brain”.

tributed fashion with their own parent, managed by a central “brain”, as demonstrated in Figure 3.2. The central “brain” serves as the common knowledge base for the distributed embodiment to retrieve knowledge representations. Meanwhile, the distributed agents are also able to store and share new knowledge representations, acquired through learning in their own life with their parent, in the central “brain”. The mode of operation enables the possibility of parallel education, such as one copy of the robotic baby is learning English in the US and another one learning Chinese in China, while a third one wandering in the streets in Paris acquiring both English and Chinese silently in secret. Since the distributed embodiment shares the same central “brain”, the behaviors of the distributed embodiment will be the same if under the same perceived external world.

Another possible mode of operation is the multi-agent setup, where there exist multiple robotic babies and each robotic baby has its own “brain”. In this setup, the robotic babies are able to communicate with each other through natural language and direct “brain merge” operations for sharing information and knowledge representations, as shown in Figure 3.3. With this setup, each robotic baby is able to acquire knowledge not only from its parent, but also from its peers. Each robotic baby will also behave differently if perceiving the same external world, since the internal “brain” is different due to the variations in the growth of



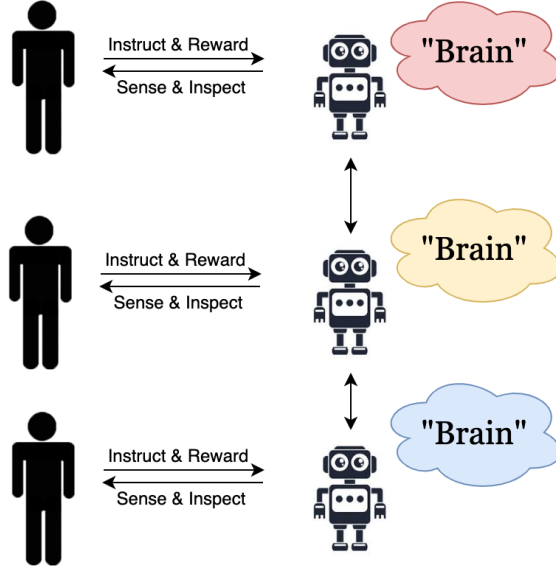


Figure 3.3: The mode of operation of the robotic baby with a multi-agent setup, where each robotic baby has its own “brain”.

the robotic baby.

### 3.3 Formal Definition of a Robotic Baby

The formal definition of a robotic baby at the time of birth,  $t_0$ , is an integrated system,  $r_0$ , containing minimal information about the external world,  $x_0$ , with a governing process  $g_0$  with parameters,  $\theta_0 \subset r_0$ , which follows the recursion,

$$r_{t+1} = g_t(x_t, r_t; \theta_t) \quad (3.2)$$

which allows the robotic baby to achieve the goals and capabilities in terms of the following high-level system requirements:

- R1 Communicating with the external world with at least one patterned signal.
- R2 Learning the patterned signal by communicating in the patterned signal.
- R3 Sensing and acting upon the external world and the internal self.
- R4 Learning new skills expressed in complex actions.

- R5 Grounding the patterned signal in both the physical and cognitive actions.
- R6 Adapting to the world with behavior changes based on reward signals.
- R7 Maintaining life state variables driven by decaying functions.
- R8 Being transparent and traceable in the internal process and explicit in the internal representation of information.

### **3.4 Philosophical Inspirations and Considerations**

#### 3.4.1 On Self Existence

It is interesting to point out that the governing process (3.1) is a direct translation of “je pense, donc je suis” (I think, therefore I am) [50] by René Descartes into abstract robotic terms. If the “thinking” process,  $f_t$ , stops working, there would not be the self,  $h_{t+1}$ , in the next time step. The governing process (3.2) for the robotic baby is, in this regard, the same as the process (3.1) for human beings.

#### 3.4.2 On Causal Determinism and Free Will

In light of the computing process described in (3.2), the initial system and the governing process define the starting point of a robotic baby. With a transparent and traceable process  $g_t$  for state changes, the computation presented in (3.2) is a deterministic process. In other words, given the starting point of a robotic baby  $(r_0, g_0)$  and a time series of the external world states  $x_0, \dots, x_t$ , it is possible to know what  $r_{t+1}$  and  $g_{t+1}$  will be in a deterministic fashion. The causal determinism presented in the robotic baby suggests that, by design, there is no such notion of free will in a robotic baby. This is in line with Baruch Spinoza’s view on free will [51]. Note that even by design there is no free will, the robotic baby could still be perceived as “free” by the outsiders, who are not able to precisely describe its future states and actions, due to the lack of access to the complete set of  $(r_0, g_0, x_0,$

...,  $x_t$ ) or due to the lack of computation power to compute  $r_{t+1}$  from  $(r_0, g_0, x_0, \dots, x_t)$  in a timely fashion, or by those who are not able to find a pattern from the seemingly chaotic behaviors generated from a deterministic rule-based system, such as Rule 110 [52] in cellular automata [53].

### 3.4.3 On Adaptation to the Unpredictable World

Beside the starting point of the robotic baby, another important factor to notice in the growth of the robotic baby is the time series of the external world state  $x_t$ . If the external world state  $x_t$  is fully predictable, no adaptation of the robotic baby to the world is needed since it is possible for the robotic baby to look up  $x_t$  and optimize for it beforehand for a given future time. Unfortunately, Laplace's demon [54], a conceptual deterministic model of the universe, has not been found or engineered and the external world  $x_t$  has not been shown to be fully predictable [55]. With this observation, the learning functionalities of the robotic baby thus focus on the ability to adapt to the unknown, rather than to optimize over the known, in a life-long learning fashion. In other words, learning paradigms such as data-driven approaches with a fixed training dataset [56] as a snapshot of the world  $x_t$  at time  $t$ , regardless of the size, are optimizing over the known for a given static  $x_t$  and thus do not fit for the design of the robotic baby.

## **3.5 Scope and Limitation**

From the high-level system requirements, many potential design choices could satisfy the requirements. For example, the patterned signal in R1 can be a natural language, a sign language, a bit stream and so on, while the sensing capability in R3 can be in visual, textual or audio signals. For the purpose of this thesis, which is to demonstrate that it is possible to build a robotic baby that satisfies all the requirements, the pursuit of complexity in the system design is of no interest or intention.

Therefore, for the patterned signal mentioned in the requirements, the scope is only

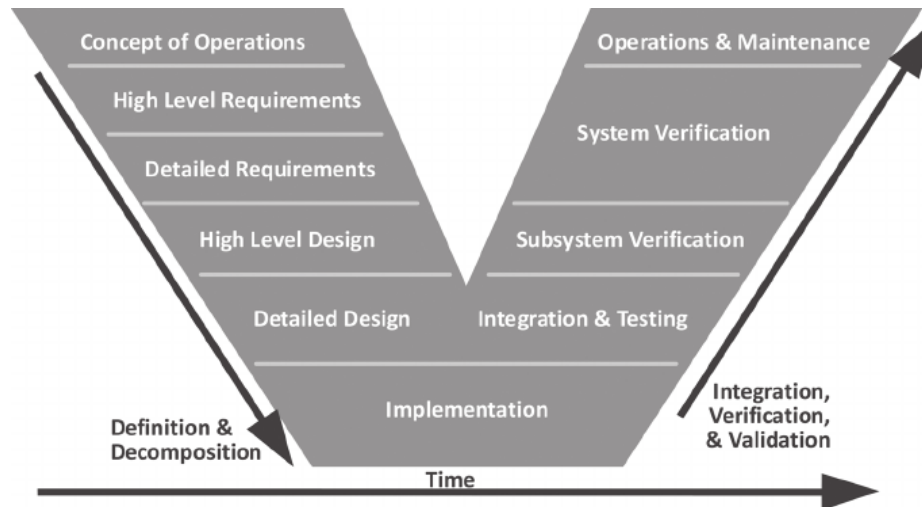


Figure 3.4: The V-model of systems engineering [64].

limited to a subset of natural languages, which can be modeled with a context-free grammar [57] and represented by a projective dependency tree [58]. For the specific natural language, English and Chinese are chosen. For sensing, the inputs are limited to text and simple images such as color blocks.

### 3.6 Systems Engineering Design Approach

The design process of the robotic baby follows a systems engineering design approach [59]. Specifically, the design process follows the V-model [60] in Figure 3.4: concept of operations, system-level design requirements, subsystem-level design requirements, high-level architecture design [61], subsystem implementation, system verification and validation, operation and retirement, which is widely adopted by the US aerospace and defense industry such as the US Department of Defense [62] and NASA [63]. Early inspiration and reasoning behind the utilization of a systems engineering design approach for the design of a robotic baby can be found in a prior related work, with a focus on the application of the systems engineering design approach [65].

To be clear, the utilization of the systems engineering design approach does not mean putting together all the best-performing subsystems, measured based on some subsystem

evaluation benchmark, to satisfy the system-level requirements. In fact, due to the lack of common system-level requirements, it is nearly impossible to integrate all the best-performing subsystems, designed for their own purposes and specifications, into a system with a different set of requirements and specifications. For example, a costly failure in this regard is the Lewis spacecraft mission [66]. In the context of a robotic baby, for instance, due to requirement R8 for system interpretability and explainability, black-box learning approaches, for example inspired by deep neural networks [67, 68], might not fit into the design unless more progress is made about explainable AI [69].

## **CHAPTER 4**

### **SYSTEM REQUIREMENTS FLOWDOWN**

#### **4.1 The Necessity of Requirements Flowdown**

Requirements flowdown is an essential step in systems engineering, in which high-level system requirements are decomposed into functional requirements which further drive the overall system design and characteristics. In principle, once a list of functional requirements of a system is set, the overall characteristics of a system have been shaped accordingly and there is no further step in the design process that could change the system characteristics. It is also important to point out that the design impact regarding a change, whether being an addition or a deletion, to the functional requirements is not linear to the requirement change, as a change in one functionality of the system could impact all related design decisions, which can propagate throughout the whole system. Therefore, requirement creeps should be avoided in the system design process, especially for the robotic baby presented within the scope of this thesis as there exist many research and development opportunities in the early ages of a robotic baby.

#### **4.2 The Requirements Flowdown for the Robotic Baby Design**

The system-level requirements for the robotic baby listed in Sec. 3.3 are generic. With the scope and limitations of robotic baby presented in Sec. 3.5, the system-level requirements are decomposed into low-level requirements for the robotic baby architecture design within the scope of this thesis.

##### **R1 Communicating with the external world with at least one patterned signal.**

R1.1 The ability to take in language inputs.

R1.2 The ability to display language outputs.

**R2 Learning the patterned signal by communicating in the patterned signal.**

R2.1 The representation of the English language.

R2.1.1 The representation of vocabulary in English.

R2.1.1.1 The representation of syntax in English.

R2.1.1.2 The representation of semantics in English.

R2.1.2 The representation of part-of-speech in English.

R2.1.3 The representation of grammar production rules in English.

R2.2 The representation of the Chinese language.

R2.2.1 The representation of vocabulary in Chinese.

R2.2.1.1 The representation of syntax in Chinese.

R2.2.1.2 The representation of semantics in Chinese.

R2.2.2 The representation of part-of-speech in Chinese.

R2.2.3 The representation of grammar production rules in Chinese.

R2.3 The acquisition mechanism and action representation.

R2.3.1 The acquisition mechanism of vocabulary in English and Chinese.

R2.3.2 The acquisition mechanism of part-of-speech in English and Chinese.

R2.3.3 The acquisition mechanism of grammar production rules in English and Chinese.

R2.3.4 The representation of the acquisition actions.

R2.3.5 The activation of the acquisition actions.

R2.4 The processing of natural language inputs.

R2.4.1 The natural language processing algorithm.

R2.4.2 The retrieval of the vocabulary, part-of-speech and grammar rules in English and Chinese.

R2.4.3 The representation of the successful processing outputs.

R2.4.4 The representation of the failed processing outputs.

**R3 Sensing and acting upon the external world and the internal self.**

R3.1 The acquisition of the states of the external world.

R3.2 The acquisition of the states of the internal self.

R3.3 The representation of actions.

R3.4 The activation of actions.

R3.5 The execution of actions on the external world.

R3.6 The execution of actions on the internal self.

**R4 Learning new skills expressed in complex actions.**

R4.1 The representation of complex actions.

R4.2 The action and activation for constructing complex actions.

R4.3 The execution of complex actions.

**R5 Grounding the patterned signal in both the physical and cognitive actions.**

R5.1 The representation of the grounding of natural language to actions.

R5.2 The acquisition mechanism of the grounding of natural language to actions.

R5.3 The action and activation for the acquisition mechanism.

**R6 Adapting to the world with behavior changes based on reward signals.**

R6.1 The representation of positive and negative reward signals.

R6.2 The representation of behavior changes.

R6.3 The action and activation for updating the representation for behavior changes.

R6.4 The utilization of the representation for behavior changes.

**R7 Maintaining life state variables driven by decaying functions.**

R7.1 The representation of internal life state variables.

R7.2 The life decaying mechanism.

R7.3 The action and activation for replenishing life state variables.

**R8 Being transparent and traceable in the internal process and explicit in the internal representation of information.**



- R8.1 The integration and coordination mechanism of system components.
- R8.2 The representation of internal processes.
  - R8.2.1 The representation of action activation.
  - R8.2.2 The representation of action execution.
- R8.3 The action and activation for displaying the internal process representation.
- R8.4 The action and activation for displaying the internal knowledge representation.

## CHAPTER 5

### SYSTEM ARCHITECTURE

Based on the system-level requirements and the requirement flow-down to the subsystem, the system architecture, *Baby*, is proposed to meet the system-level requirements. The overall architecture is shown in Figure 5.1.

The *Baby* architecture consists of processes, queues and functional structures which can store information. This integrated architecture takes into consideration the following: language processing and input/output, representations of knowledge in a graph, the execution of physical actions and sensing, the activation of actions based on input signals, system states and the overall management of the system. The traceability of how the overall system-level requirements are satisfied by sub-system units is presented in Figure 5.2.

#### 5.1 Language Input and Output

The language input and output unit, shaded in blue in Figure 5.1, consists of two processes and their corresponding buffers for piping information. The input process is a process monitoring input messages and it will simply put the message onto the BufferIn queue. The display process is a process that takes in the message from the BufferOut queue and display it on a screen. The BufferIn queue is a queue with messages, monitored by the Manager process. The BufferOut queue, like the BufferIn queue, is also monitored by the Manager process and once a message is on the queue, the Manager process will spawn the Display process to display the message on the BufferOut queue. This design supports requirement R1 and R8.

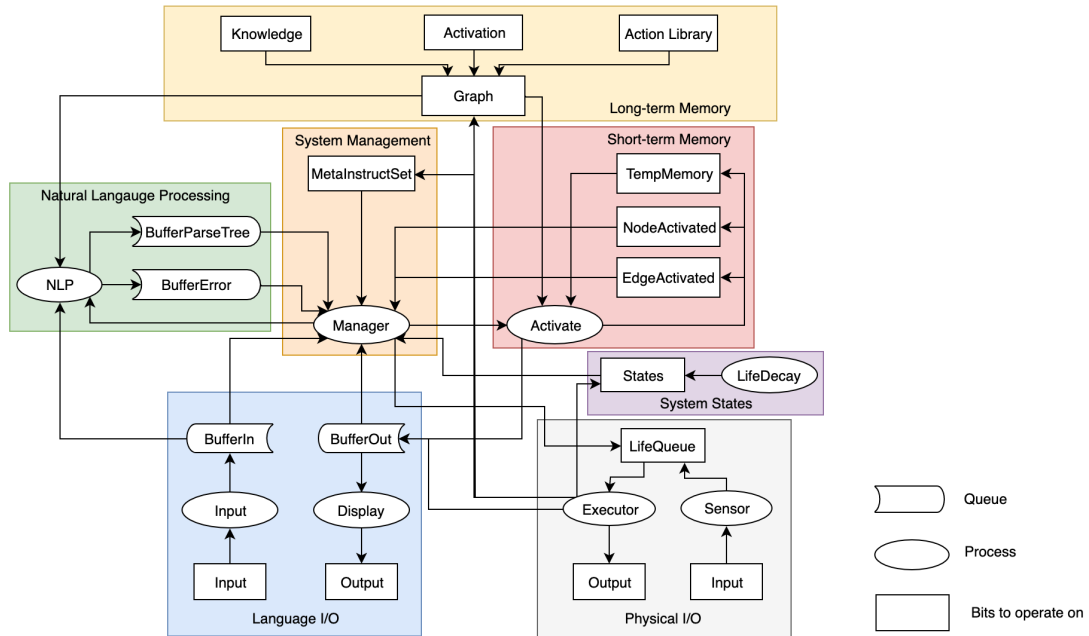


Figure 5.1: The proposed system architecture, *Baby*, for the robotic baby.

Traceability		Requirement							
		R1 on communication	R2 on language acquisition	R3 on sensing and action	R4 on compositionality	R5 on language grounding	R6 on adaptation	R7 on system states	R8 on explainability
Subsystem Unit	Language Input and Output	✓							✓
	Natural Language Processing		✓				✓		✓
	System Management								✓
	Long-term Memory		✓	✓	✓	✓	✓	✓	✓
	Short-term Memory and Activation			✓	✓		✓		✓
	System States							✓	✓
	Physical Input and Output			✓					✓

Figure 5.2: The requirement satisfaction traceability matrix for the *Baby* architecture design.

## 5.2 Natural Language Processing

The customized language processing unit, shaded in green in Figure 5.1, consists of the natural language processing (NLP) process and two queues, the BufferParseTree and

BufferError queues. The NLP process is spawned when the Manager detects that there is a message on the BufferIn queue to be processed. The job of the NLP process is to jointly conduct part-of-speech (POS) tagging and parse the input sentence into a projective dependency tree, based on the obtained linguistic rules stored in the Graph in the long-term memory. If the parse is successful, the parse tree will be put on the BufferParseTree queue. If the parse is not successful, some error message will be put on the BufferError queue. This design supports requirement R2, R6 and R8.

The algorithm for the NLP process, **BabyParse**, is a rule-based greedy-search parsing algorithm, utilizing the linguistic knowledge for the rules and the corresponding rule activation frequencies stored in the Graph for the heuristic for the greedy search, inspired by the Earley parser [70] with memorization during backtracking [71]. The backtracking states utilize the configuration in transition-based parsing algorithms [58], with a buffer,  $b$ , consisting of pre-processed tokens and a stack,  $s$ , containing specialized production rule entries. Each stack rule entry is a dict with the following keys: type, rule, head, pre\_pos, tail and element, where ‘type’ indicates the production of interest, and ‘element’ is a list of POS-tagged tokens. The initial stack contains only one entry,  $Root=\{\text{type:}S, \text{rule:}\emptyset, \text{head:}\emptyset, \text{pre\_pos:}\emptyset, \text{tail:}\emptyset, \text{element:}[\ ]\}$ . There is also a list,  $r$ , for storing the finished stack rule entries. The algorithm also keeps track of two states, the current word,  $w_n$ , and the POS tag of the current word,  $p_{w_n}$ . The **BabyParse** algorithm is presented in Algorithm 1.

The **BabyParse** algorithm is a recursive function, conducting greedy search of POS tags and grammar production rules for the transition actions, considering the state of the current word and its POS and how they fit to the current production rule in the top stack entry. It utilizes the transition-based configuration as the states for backtracking when facing a dead end. It is custom made to tailor to the representations of the rules accumulated by the robotic baby. Line 2-10 describe the situation with an empty buffer. Line 11-14 describe the situation with no next word, at the very beginning or when a word is moved from  $b$  to  $s$ . In line 13, the POS function gives an ordered list of POS tags based on the

weights. Line 15-19 describe the state with no proposed ‘rule’ for ‘type’, when a new stack entry is created. The PR function in line 16 gives an ordered list of production rules based on the weights. In line 20, the `Compatible` function checks whether the proposed  $p_{w_n}$  is compatible with the current hypothesized rule given its previous POS tag. Line 21-35 describe what to do if the hypothesized rule has not encountered its tail, with Line 27 checking if the element can be decomposed. Line 36-45 describe the states with/without passed compatibility check.

The output of the algorithm is either fail or success with the list of rules,  $r$ , which can be easily transformed to a dependency graph by draw an arrow from each element in ‘element’ to ‘head’ for each rule in  $r$ . If the algorithm fails to produce a dependency graph, it means that there might be a new word, a new POS for a known word or a new grammar rule that needs to be learned, given a grammatical input sentence.

---

**Algorithm 1** BabyParse, the algorithm for joint POS-tagging and dependency parsing

---

**Input:**  $b = w$ ,  $s = [Root]$ ,  $r = []$ ,  $w_n = \emptyset$  and  $p_{w_n} = \emptyset$ **Output:** success or fail, and  $r$  if success

```
1: function BabyParse( $b, s, r, w_n, p_{w_n}$ )
2:   if not  $b$  then                                     ▷ check if buffer empty
3:     while  $s$  do
4:       if  $s_{top}[tail]$  then
5:         move  $s_{top}$  from  $s$  to  $r$ 
6:       if not  $s$  then
7:         return success,  $r$ 
8:       fill  $s_{top}[element]$  with previous  $s_{top}[head]$ 
9:     else
10:      return fail
11:   if not  $w_n$  then
12:      $w_n = b[0]$ 
13:     for  $p_{w_n}$  in POS( $w_n$ ) do
14:       return BabyParse( $b, s, r, w_n, p_{w_n}$ )
15:   if not  $s_{top}[rule]$  then
16:     for  $pr$  in PR( $s_{top}[type]$ ) do
17:       if Compatible( $pr, p_{w_n}, \emptyset$ ) then
18:          $s_{top}[rule] = pr$ 
19:       return BabyParse( $b, s, r, w_n, p_{w_n}$ )
20:   flag $_{pr}$ , e_list = Compatible( $s_{top}[rule], p_{w_n}, s_{top}[pre-pos]$ )
21:   if not  $s_{top}[tail]$  then
22:     if not flag $_{pr}$  then
23:       return fail
24:     else                                             ▷ continue with  $s$ 
25:       for  $e$  in e_list do
26:          $b_c, s_c, r_c = \text{Copy}(b, s, r)$                  ▷ copy configuration
27:         if Check_Terminal( $e$ ) then                   ▷ append to  $s_{c_{top}}$ 
28:            $b_c.pop(0)$ 
29:            $s_{c_{top}}[element].append((w_n, p_{w_n}))$ 
30:            $s_{c_{top}}[pre-pos] = p_{w_n}$ 
31:           update  $s_{c_{top}}[head], s_{c_{top}}[tail]$  if fit
32:           return BabyParse( $b_c, s_c, r_c, \emptyset, \emptyset$ )
33:         else                                         ▷ create new stack top
34:           create new  $s_c$  top entry from  $e$ 
35:           return BabyParse( $b_c, s_c, r_c, w_n, p_{w_n}$ )
36:   else
37:     if not flag $_{pr}$  then
38:        $b_c, s_c, r_c = \text{Copy}(b, s, r)$                  ▷ copy configuration
```

---

---

```

39:         move  $s_{ctop}$  from  $s_c$  to  $r_c$ 
40:         return BabyParse( $b_c, s_c, r_c, w_n, p_{w_n}$ )
41:     else
42:         try continue with  $s_{ctop}$  as in line 25-35
43:         if fail then
44:             move  $s_{ctop}$  from  $s_c$  to  $r_c$ 
45:             return BabyParse( $b_c, s_c, r_c, w_n, p_{w_n}$ )

```

---

### 5.3 System Management

The system is managed by the Manager process, inspired by some proposed functionalities of the basal ganglia [72] in the human brain. The Manager process, shaded in orange in Figure 5.1, operates according to the instructions stored in the `MetaInstructSet`. The Manager process is a process that runs with an infinite loop with a cognitive cycle period, reacting to new information, as a stimulus, in the Queues of the system, such as the `BufferIn` and `BufferParseTree` queues, and monitoring system States. For example, if there is a dependency tree in the `BufferParseTree` queue, the Manager will spawn the `Activate` process to further process the dependency tree. Moreover, the Manager process is in charge of the motor and sensory capabilities of the robotic baby, sending action signals from the `Activate` process output to the `LifeQueue` unit, where actions are executed. In addition to the internal states and action signals monitored, the Manager process also does required state monitoring for any external cyber or physical system integrated to the robotic baby. The algorithm, `BabyCycle`, for the Manager process is presented in Algorithm 2. This design supports requirement R8.

### 5.4 Long-term Memory

The long-term memory unit, shaded in yellow in Figure 5.1, houses the knowledge Graph, with accumulated knowledge and rule activations explicitly represented in the nodes and edges of various types. Node types include, but are not limited to, functional, syntactic, semantic, part-of-speech, grammar production, grammar production rule, action and acti-

---

**Algorithm 2** BabyCycle, the algorithm for system management

---

```
1: function BabyCycle( )
2:   while True do
3:     sleep(CognitiveCycleTime)
4:     LifeDecay(States)
5:     if BufferIn not empty and StateAcceptInput then
6:       NLP(Graph, BufferIn, BufferParseTree, BufferError)
7:     if BufferParseTree not empty then
8:       Activate(Graph, TempMemory, NodeActivated, EdgeActivated)
9:     if BufferError not empty then
10:      ActivateError(Graph, TempMemory, BufferOut)
11:    if NodeActivated not empty then
12:      LifeQueue.append(Graph, NodeActivated)
13:    if EdgeActivated not empty then
14:      StateEdgeActivation.append(Graph, EdgeActivated)
15:    if StateRunMotorSensor then
16:      Executor(LifeQueue, Graph, States, BufferOut, MetaInstructSet)
17:      Sensor(LifeQueue, States, TempMemory)
18:    if BufferOut not empty and StateProduceOutput then
19:      Display(BufferOut)
20:    if ExternalStates then
21:      ExternalFn()
```

---

vation nodes. The purpose of the knowledge graph is to represent the acquired knowledge and to provide the structure and information for inference and activation by the Activate process. Required knowledge for kick-starting system operation includes linguistic knowledge such as the notion of part-of-speech and grammar production rules, which the NLP unit relies on. A sample subgraph representing the syntactic and semantic graph structure for the word ‘bank’ is shown in Figure 5.3. The word ‘bank’ could be used as a noun or a verb. As a noun, it could mean a financial institution or a river bank. As a verb, it means to tilt.

The edges in the knowledge graph are also typed with activation weights attached, representing the frequency of the activation between two nodes. In the ‘bank’ subgraph example, it can be seen that ‘bank’ is used as a noun more than as a verb. Moreover, grammar production rules are also represented in the graph. The grammar rule representation is de-



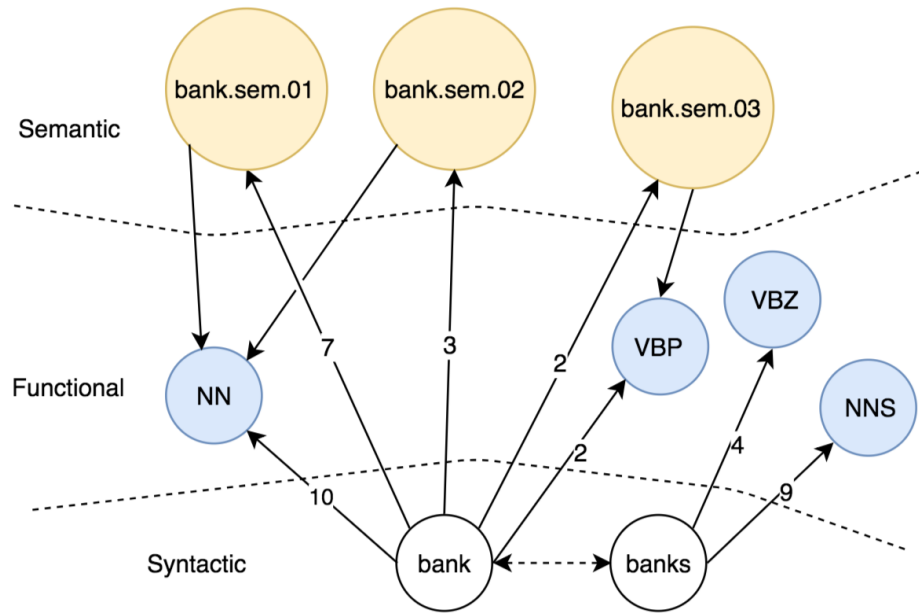


Figure 5.3: Knowledge representation for the word ‘bank’ in a subgraph.

signed with context-free grammar structures plus the usage of regular expression symbols for flexibility, along with a head element and a tail element for the rule. Regular expression symbols include ‘\*’ for an optional and potentially repetitive element, ‘+’ for a potentially repetitive element and ‘?’ for an optional element. An example grammar rule for singular noun phrase (NP\_S) is shown in Equation 5.1.

$$\text{NP\_S} \rightarrow \text{DT? JJ* NN+} \quad (5.1)$$

In this example, DT is determiner, JJ is adjective and NN is singular noun. This production rule with the regular expression symbols is applicable to singular noun phrases of different lengths, such as “juice”, “the apple juice”, “the tasty apple juice”, “the cold tasty apple juice” and etc. Sample noun phrase (NP) and verb phrase (VP) production rule nodes are shown in Figure 5.4. Sample knowledge base of vocabulary and part-of-speech can be found in Appendix A and sample knowledge base of grammar production rule can be found in Appendix B.

Beside knowledge stored in the graph, the action and activation nodes are also stored in

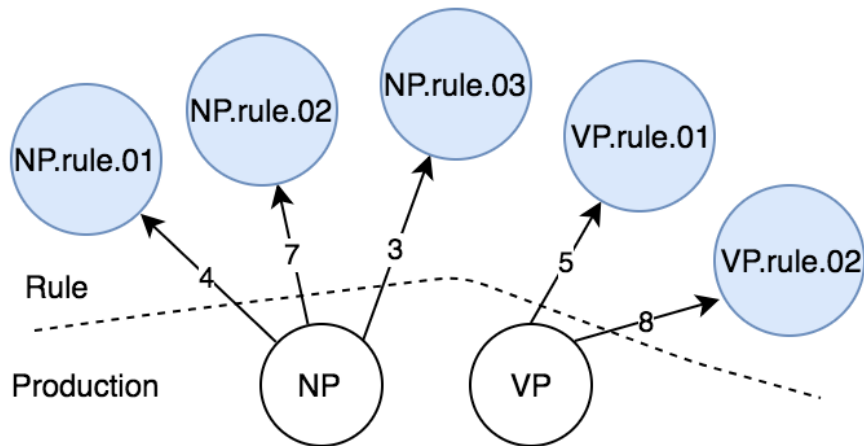


Figure 5.4: Knowledge representation for NP and VP production rules in a subgraph.

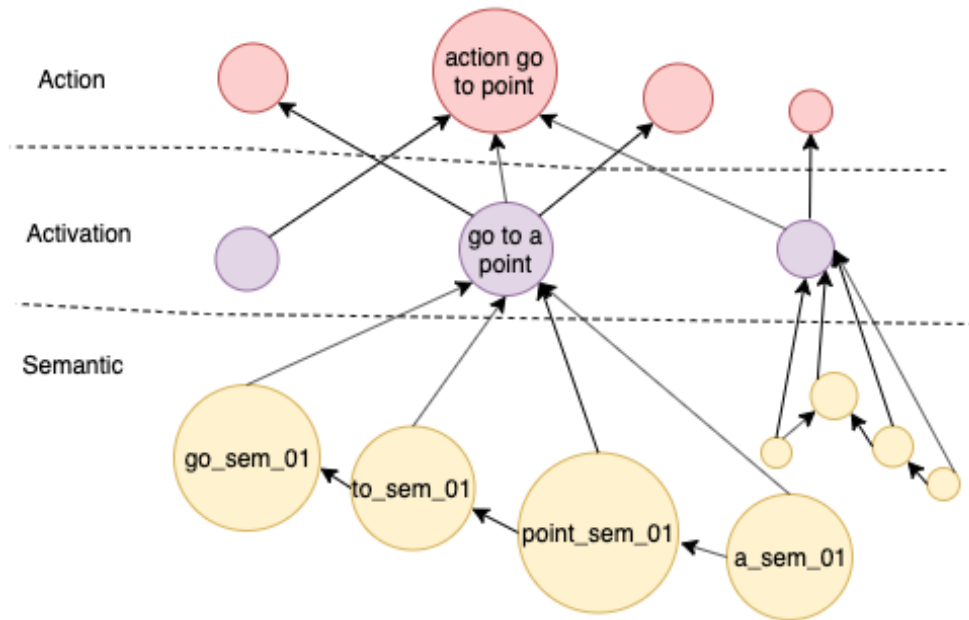


Figure 5.5: The default activation of the action node 'go to point'.

the knowledge activation graph. The action nodes encapsulate system actions, cognitive or physical, and serves as the pointer to the actions. Each action node has a default activation node attached, which represents a dependency graph from a parse tree. Each dependency graph is a subgraph containing nodes in the semantic and function layer of the knowledge graph. An example for the action of 'go to a point' is shown in Figure 5.5.

Note that in Figure 5.5, each system action node has a default corresponding activation

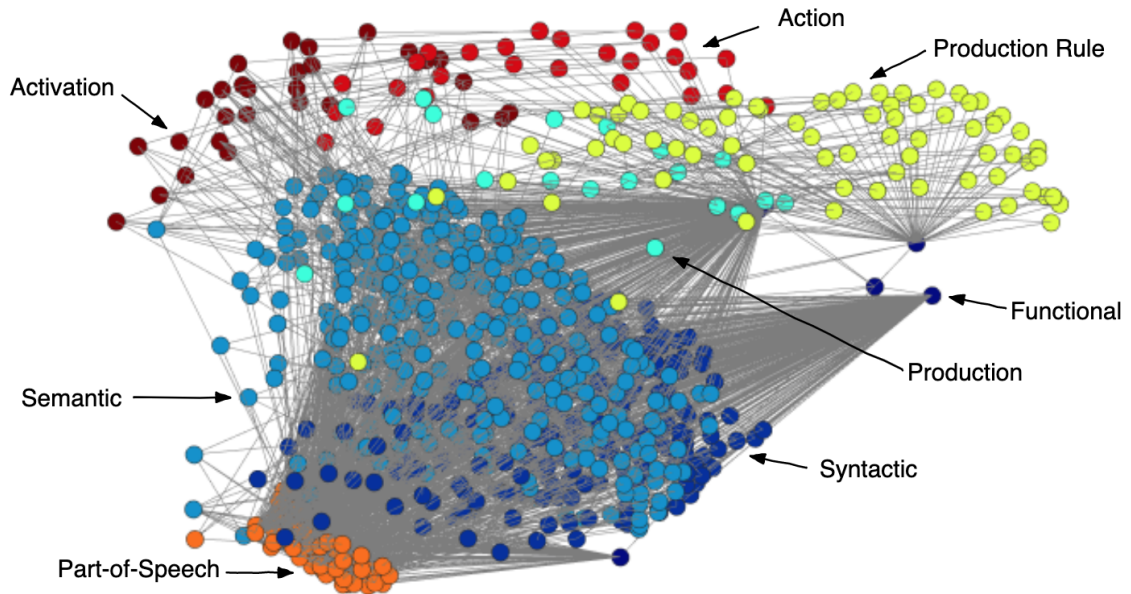


Figure 5.6: A snapshot of the knowledge activation graph containing all types of nodes and edges at a point in time.

node, which is activated by a default dependency graph consisting of semantic nodes. Yet, the graph is not rigid. New nodes and edges will be added to the graph as a result of the interactions between the robotic baby and its parent. In other words, for a system action node, there could be more than one activation nodes attached, and an activation node could potentially activate multiple action nodes, as shown in Figure 5.5. For the activation node, other dependency graphs, which are semantically equivalent to the default dependency graph, can also activate the same activation node. This is how the semantics is grounded to the system actions, which modify the graph itself as well as the external world. The knowledge activation graph serves as the “neural network” of the robotic baby, which provides the structure for activation and inference with the `Activate` process in the activation unit. For illustration, a snapshot of the knowledge activation graph containing all types of nodes is shown in Figure 5.6. The overall design of the long-term memory supports requirement R2, R3, R4, R5, R6, R7 and R8.

## 5.5 Short-term Memory and Activation

The short-term memory unit, shaded in red in Figure 5.1, contains the `Activate` process and a memory space. The memory space contains a general space, `TempMemory`, for storing temporary information such as past natural language inputs. The memory space also contains dedicated space for activated nodes and edges, as in `NodeActivated` and `EdgeActivated`. The `NodeActivated` space is used for storing the recent action node, which the Manager checks within each cognitive cycle and pushes to the `LifeQueue` for execution. The `EdgeActivated` space is used for accumulating all recently activated edges in the knowledge activation graph. The weight of the activated edges, with the reception of a reward natural language signal, will be updated in the knowledge graph. This design supports requirement R3, R4, R6 and R8.

The `Activate` process is a process spawned when the Manager detects the existence of content on `BufferParseTree` from the NLP process. The job of the `Activate` process is to infer, based on the input semantic dependency graph and the knowledge activation graph, which action node to place on `NodeActivated` as the output. When the activation finishes, all the utilized edges that result in the final action node will be accumulated in `EdgeActivated`. The `Activate` process utilizes the `BabyActivate` algorithm, presented in Algorithm 3.

The `BabyActivate` algorithm is a stimulus-driven search algorithm with hypergraphs as an input. Due to the nature of natural languages, a constituent can span multiple words forming a representation of one semantic meaning. In the context of the Graph, constituents are represented by hypergraphs, containing nodes connected by edges carrying the same information. For activations by a hypergraph, the activated node has to be verified that it is indeed activated by all relevant nodes and edges in the hypergraph. With the `BabyActivate` algorithm, the input dependency graph is first activated by nodes. Line 5-22 describe the process with single node activation. Line 10-20 describe when a new node is activated,

---

**Algorithm 3** BabyActivate, the algorithm for short-term graph activation

---

**Input:** D = DependencyGraph, G = Graph

**Output:** NodeActivated, EdgeActivated

```
1: function BabyActivate(D, G)
2:   FrontierNodes, FrontierEdges = D.nodes, D.edges
3:   UnexploredNodes, ActivatedNodes = [], []
4:   while True do
5:     while FrontierNodes do
6:       node = FrontierNodes.pop(0)
7:       nodePath, nodeSpan = D.getPathSpan(node)
8:       neighbors = G.getActivationTarget(node)
9:       for neighbor in neighbors do
10:        if not G.checkActivate(node, neighbor) then
11:          continue
12:        D.addNode(neighbor)
13:        for p in D.predecessors(node) do
14:          D.addEdge(p, neighbor)
15:          FrontierEdges.append((p, neighbor))
16:        for s in D.successors(node) do
17:          D.addEdge(neighbor, s)
18:          FrontierEdges.append((neighbor, s))
19:        D.updatePathSpan(neighbor, node, nodePath, nodeSpan)
20:        FrontierNodes.append(neighbor)
21:        if G.checkAction(neighbor) then
22:          ActivatedNodes.append(neighbor)
23:     while FrontierEdges do
24:       fn, tn = FrontierEdges.pop(0)
25:       neighbors = G.getEdgeActivationTarget(fn, tn)
26:       if neighbors is empty then
27:         continue
28:       fnPath, fnSpan = D.getPathSpan(fn)
29:       tnPath, tnSpan = D.getPathSpan(tn)
30:       for neighbor in neighbors do
31:         if neighbor in FrontierNodes then
32:           continue
33:         if neighbor not in UnexploredNodes then
34:           D.addNode(neighbor)
35:           UnexploredNodes.append(neighbor)
36:         D.updatePathSpanEdge(neighbor, fn, fnPath, fnSpan, tn, tnPath, tnSpan)
37:         if D.checkActivate(neighbor) then
38:           for pathNode in D.getPath(neighbor) do
39:             for p in D.predecessors(pathNode) and not in D.getSpan(neighbor) do
40:               D.addEdge(p, neighbor)
41:               FrontierEdges.append((p, neighbor))
```

---

---

```

42:         for s in D.successors(pathNode) and not in D.getSpan(neighbor)
      do
43:           D.addEdge(neighbor, s)
44:           FrontierEdges.append((neighbor, s))
45:           FrontierNodes.append(neighbor)
46:           UnexploredNodes.remove(neighbor)
47:       if FrontierNodes and FrontierEdges are empty then
48:           break
49:       if ActivatedNodes is not empty then
50:           NodeActivated = G.rank(ActivatedNodes, D)
51:           EdgeActivated = D.getPath(NodeActivated)
52:       return NodeActivated, EdgeActivated

```

---

the new node is added to the dependency graph and the search frontier. It also inherits all the predecessors and successors of the activation source along with associated edges. Line 23-46 describe the process with edge activation. Line 31-36 describe when a new node is activated by an edge, the new node, if not already activated and placed in the search frontier and if not fully activated due to hypergraphs, is placed in the unexplored node candidate list. The partially activated node records the source activation edge information. If the new node is fully activated by the edge, whether as part of a hypergraph, the new node will inherit all the associated predecessors and successors from its activation sources, seen in Line 37-44. Line 47-48 describe the exit condition out of the infinite loop when all relevant node and edge activations are completed. Line 49-51 describe the selection of the output action node and the associated activation paths toward the final action node. If there are multiple action nodes activated, the ranking is based on heuristic rule filters and activation weights stored in the Graph.

## 5.6 System States

The system states unit, shaded in purple in Figure 5.1, contains the `States` object, which keeps track of essential states of the system, some for the simulation of the decaying life state, which needs a special signal to get replenished. Beside the life state, other state vari-

ables include the gates and flags utilized by the cognitive cycle, such as `StateAcceptInput`, `StateRunMotorSensor` and `StateProduceOutput`. The `States` object also keeps track of the variables of the external system integrated to the *Baby* architecture. The `LifeDecay` process is a pre-defined process guiding the decay of the life states, which enables the natural need for outside resources. This mechanism is designed as the driver for interactions to the parent and the world, which, in the long term, serves as one of the objectives for sustainability and autonomy. This design supports requirement R7 and R8.

## 5.7 Physical Input and Output

The physical input and output unit, shaded in grey in Figure 5.1, contains the `LifeQueue`, which is a queue for activated action nodes to be put on. The `Executor` process constantly monitors the `LifeQueue` for executing the next action node once the last node has been completed. The `Executor` is not only executing physical actions but also cognitive actions as well, such as adding a node to the `Graph` and updating the edge weights of the `Graph`. The `Sensor` process constantly monitors whether the objective of the current action node is achieved and signals to the `LifeQueue` when an action node has been completed. This design is for requirement R3 and R8.

## CHAPTER 6

### SUB-SYSTEM IMPLEMENTATION AND INITIALIZATION

With the *Baby* system architecture, the initialization of the Graph plays an important role at the starting point of a robotic baby. For inspiration, it might be beneficial to look at the case of the human counterpart first. For a human baby, there are theories regarding the stages of development in cognition and knowledge, for example by Jean Piaget [4] and others [49]. Yet, there does not seem to be a consensus on the precise definition of the development stages and how exactly a human baby evolves from one stage to another in the field of cognitive development. For instance, on mental representation, some argue that mental representation is innate [6, 73], which contradicts Piaget's theory that mental representation depends on prior sensorimotor development [3]. For the purpose of designing a robotic baby, the research in cognitive development suggests, in abstraction with robotic terms, that at the time of birth  $t_0$ , a human baby  $h_0$  barely knows anything about the self or the external world, while at some future time  $t_1$ , a human baby grows into a state  $h_1$ , such as the the formal operational stage in Piaget's theory [4], when intelligence is shown with the usage of logic and abstract concepts. The precise transformation from  $h_0$  to  $h_1$ , however, is not known to a complete extent, despite some observations of behaviors to facilitate the transformation, such as babbling [74] and action imitation [75].

For the design of the robotic baby, whose purpose is not to emulate the human growth process from  $h_0$  to  $h_1$  but to satisfy the system level requirements, the initialization of the robotic baby  $r_0$  in Eq. (3.2) does not necessarily need to be similar to the state of the human counterpart  $h_0$ . In other words, it is suggested by the robotic baby designer that the human baby growth process from  $h_0$  to  $h_1$  is only specific to the human initialization  $h_0$ , while there could be alternative processes that drive the growth of a robotic baby  $r_0$  with different starting points to satisfy the system-level requirements.



## 6.1 Linguistic Knowledge Initialization

For the acquisition of language, unlike the process a human baby goes through in child cognition and psychology proposed in [76, 77], the robotic baby is designed to be born with minimal linguistic knowledge and mechanisms in terms of cognitive actions, to enable the further acquisition of language. This notion of jump-starting is a design choice, inspired by the design of a dictionary. Consider a thought experiment: if given a dictionary, which word should a human baby, who has no language skills, learn first? It turns out that there is not an order for acquiring new vocabulary in reading a dictionary for a baby from knowing nothing. Any word in a dictionary is defined by its part of speech and other words for explaining its meaning. A dictionary on its own is nothing but a circular definition of symbols with neither a starting point nor an ending point. Furthermore, the design of a dictionary assumes that the reader knows how part of speech works in parsing and that the reader has the cognitive capacity to make associations, for example, between a word and a part of speech, along with its meaning.

In light of the observations on the design of a dictionary, the design of the robotic baby for language acquisition employs a similar setup. Regarding the notion of part of speech and the associated parsing mechanism, the robotic baby, at birth, is set up with structures for representing part of speech and grammar rules in the Graph, and the **BabyParse** algorithm for parsing. Regarding the circular definition of words, a minimal set of words that form a circular definition is found and represented in the Graph, and serves as the basis for further linguistic knowledge acquisition.

In English, the very initial set of words present in the Graph at birth includes ‘noun’, ‘is’, ‘a’, ‘verb’ and ‘determiner’, which could form a circular definition as in “‘noun’ is a noun; ‘verb’ is a noun; ‘is’ is a verb; ‘determiner’ is a noun; ‘a’ is a determiner”. In this circular definition, every word is defined by all other words. With this basis, further acquisitions of new words and part of speech are made possible, such as “‘pronoun’ is a

noun; ‘you’ is a pronoun”, where ‘pronoun’ and ‘you’ are acquired from the basis.

In Chinese, similarly, the very initial set of words present in the Graph at birth includes ‘名词’ (noun), ‘是’ (be) and ‘动词’ (verb), which could form a circular definition as in “‘名词’是名词; ‘动词’是名词; ‘是’是动词”. In this circular definition, just like the one in English, every word is defined by all other words. With this basis, further acquisitions of new words and part of speech are made possible, such as “‘代词’是名词; ‘你’是代词”, where ‘代词’ (pronoun) and ‘你’ (you) are acquired from the basis.

## **6.2 Action Space and Activation Initialization**

The robotic baby is designed to be born with primitive actions, the building blocks for high-level cognitive skills and complex actions, similar to the proposed theory for a human baby, where skills are hierarchically assembled as ‘schemata’ [2]. The primitive actions are pre-defined programs that interact with other internal parts of the robotic baby, such as the Graph, TempMemory and States, and the external world if integrated into a physical body. Examples of primitive actions include the actions for language acquisition in terms of the addition of words, parts of speech and grammar rules to the Graph, and the actions for language grounding, program construction, behavior reinforcement and life states replenishing.

For a robotic baby, all primitive actions form an action space. The action space defines a capability boundary for all future actions no matter how complex they would become, since complex actions are essentially constructed with all the primitive functions defined at birth. In the human case, it is known that a human is not capable of maintaining steady and level flight at high altitude with just the human body and nothing else, since the physical action ‘fly’ cannot be constructed with all possible actions a human body can perform. For a robotic baby, similar statements can be made regarding a complex action with a given set of primitive actions at birth.

All primitive actions in a robotic baby at birth are represented as action nodes in the

Graph. For each primitive action, there is a default hypergraph activation node, expressed in natural language semantics in both English and Chinese, connecting to it. This is a design choice, enabling the reference to and the manipulation of a specific action by grounding a semantic meaning to an action at birth, where in contrast, for a human baby, there is no such grounding of meaning to an action observed at birth, which is hypothesized to be gradually obtained throughout growth [2, 5]. Note that it is the meaning, expressed in natural language semantics, that is grounded to an action, not the natural language syntax that is the choice for grounding. For instance, for an action ‘raise hand’, it is the semantics of ‘raise hand’ that is grounded to the representation of the action ‘raise hand’, not the exact syntactic wording ‘raise hand’ that is grounded to the action. This design choice takes account into the variance and flexibility in natural languages and enables the establishment of new mappings from similar meanings in semantics, even expressed in another language with another set of symbols, to the activation of an action.

## CHAPTER 7

### SYSTEM VERIFICATION AND VALIDATION

#### 7.1 Acquisition and Semantic Parsing of English and Chinese

With the initialization of the Graph, the robotic baby is capable of learning English with instructions in English. Specifically, the robotic baby is able to acquire the building blocks of the English language, namely the vocabulary words, the various types of part of speech (POS) and grammar rules, via live interactions with a parent. Note that English is an evolving natural language with newly generated words [78] and changes to the meanings of words [79]. There is also no consensus on the number of POS tags for the English language. Notable POS tag sets include ones from the Penn Treebank [80] and the Universal Dependencies [81]. With the evolving nature of the syntax and the semantics of the words, along with different annotation traditions in the POS tags, there is no universal set of grammar rules available for the English language. As a result, there is no set curriculum for acquiring the complete English language, if it can be even defined at any point in time. Therefore, the acquisition of English for the robotic baby does not focus on building and following a particular curriculum. Instead, the focus is on the capabilities for adaptation, regardless of the styles of symbols and annotations used in English.

There are three major cognitive primitive actions that are in charge of the acquisition of language. They are `add_new_word`, `add_new_pos` and `add_new_grammar`. The actions, along with an example English instruction for each, are shown in Table 7.1. The `add_new_word` action adds a new word to the Graph and associates the new word with a known part of speech. In English, it is activated by a hypergraph with ‘be type of part-of-speech’ expressed in semantics. In the example, the dependency graph of ‘is plural noun’ activates the ‘be type of pos’ hypergraph, which further activates the action node.

Table 7.1: Action Nodes for Language Acquisition in English

Action Node	Example Instruction in English (EN)
<code>add_new_word</code>	‘apples’ is a plural noun.
<code>add_new_pos</code>	‘plural noun NNS’ is a part of speech.
<code>add_new_grammar</code>	‘DT? JJ* NNS, NNS, NP_P’ is a grammar rule.

Table 7.2: Action Nodes for Language Acquisition in Chinese

Action Node	Example Instruction in Chinese (CN)
<code>add_new_word</code>	‘苹果’是名词。
<code>add_new_pos</code>	‘形容词 JJ_CN’是词性。
<code>add_new_grammar</code>	‘JJ_CN POSS_CN NNS, NNS, NP_P’是语法。

The `add_new_pos` action adds a new part of speech to the graph. The part of speech includes a natural language name and a symbol. In the example, plural noun is added as a part of speech with a symbol ‘NNS’. The `add_new_grammar` action adds a new grammar production rule to the graph. In the example, the grammar rule ‘determiner (optional) + adjective (optional, repetitive) + plural noun’ is added for the production of plural noun phrase (NP\_P), with the plural noun as the head of the grammar rule.

Beside English, the actions are also connected to the semantics in Chinese. Example instructions in Chinese for the actions are presented in Table 7.2. Similar to the examples in English, the Chinese instructions follow similar patterns for the activation of the actions. Note that the use of English letters is common in Chinese to represent symbols. The actions, both in English and in Chinese, have been verified to work as designed by inspecting the change of the Graph after the activation of the actions.

For the semantic parsing of English and Chinese sentences, verification is done by parsing different types of input sentences in terms of complexity and ambiguity, given necessary vocabulary, part of speech and grammar rules. Note that in Chinese, since normal Chinese sentences are not tokenized with a white-space marker between words, an extra step for tokenization before parsing is implemented with a rule-based greedy-search tokenizer utilizing the Graph. The robotic baby has not encountered problems parsing unambiguous but complex sentences in English and Chinese both using the `BabyParse` algorithm. One

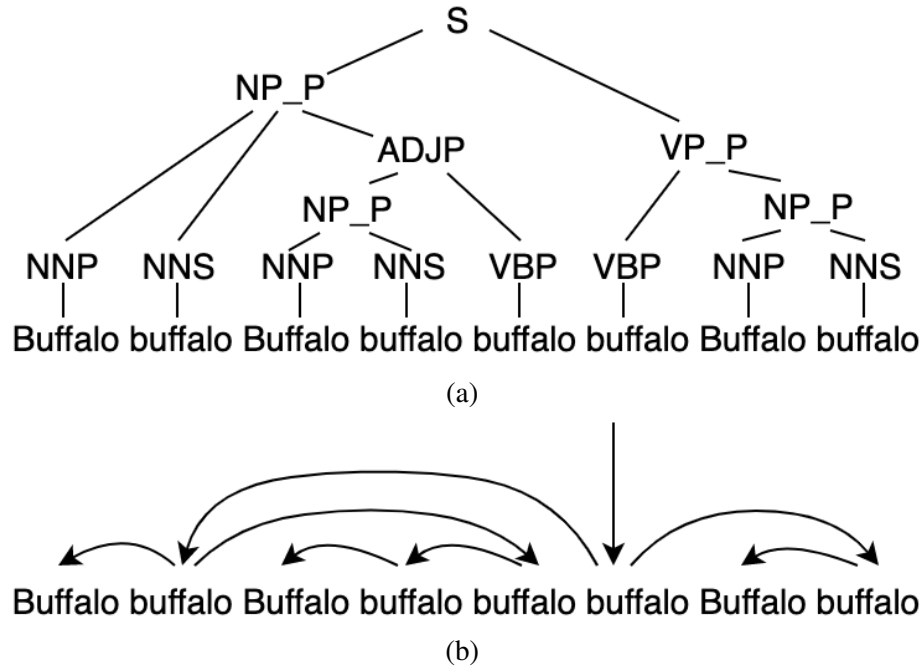


Figure 7.1: The parse tree and dependency graph for “Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo”.

example in this regard in English is “The mouse that the cat that the dog that the man frightened chased bit ran away” [82], where there are recursively embedded clauses and structures in the complex sentence. For ambiguous sentences, if the ambiguity can be resolved solely with the clue from grammar rules, the robotic baby can parse sentences with such ambiguity successfully into parse trees and dependency graphs. An example of this type of ambiguity is the lexical ambiguity seen in “Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo” [83]. In this example, ‘buffalo’ can be used as a verb and a noun, and a proper noun when ‘b’ capitalized as in ‘Buffalo’. The **BabyParse** algorithm has succeeded in parsing this grammatical sentence solely depending on obtained grammar rules, with the parse tree shown in Figure 7.1a and the dependency graph in Figure 7.1b. A detailed execution of **BabyParse** on the garden-path sentence “the old man the boat” can also be found in Appendix C, where the garden-path behaviors [84] can be observed.

The types of ambiguity in sentences that cannot be decided by the robotic baby include semantic ambiguity that needs world knowledge to resolve and prepositional phrase (PP)

attachment ambiguity that needs context to resolve [85]. As a starting point, the current parsing capabilities are sufficient for the purpose of the robotic baby. Advanced parsing skills can be obtained through the acquisition of procedures and programs in future learning.

## 7.2 Bilingualism, Multilingualism and Code-Switching

With the bilingual language acquisition capabilities, the robotic baby is capable to acquire linguistic knowledge of one language using another language. More specifically, the actions for language acquisition activated in one language can manipulate the Graph regarding the linguistic knowledge of another language. As an example, in “‘他’ is a pronoun”, the Chinese word ‘他’ (he) is acquired by the robotic baby in English and associated with the pronoun part of speech. In another example, “‘pretty’是形容词”, the English word ‘pretty’ is acquired in Chinese as an adjective. As a matter of fact, English and Chinese are different languages with different sets of vocabularies and grammar rules. Yet, to a robotic baby, they are nothing but symbols with particular structures and orderings that obey some commonly accepted rules, modeled with context-free grammar in the robotic baby implementation, of a particular natural language. To further validate the symbol-agnostic language semantic parsing, a made-up language with arbitrary words, parts of speech and grammar rules is taught and the robotic baby has no problem parsing grammatical inputs in the made-up language. For future expansion, it might not take too much effort to acquire and use, as an example, French, with “‘jolie’ is an adjective” or “‘pretty’ est un adjectif” despite some complications such as in gender and verb tense categories in French.

The bilingual language representation in the robotic baby enables another interesting capability of dealing with input natural language sentences with code-switching [86]. For the robotic baby, code-switching is relevant when the input sentence contains more than one natural language. Notice that a grammatical sentence with code-switching still follows well-defined grammar rules [87]. As an example, in “我有一个plan” (I have a plan), the

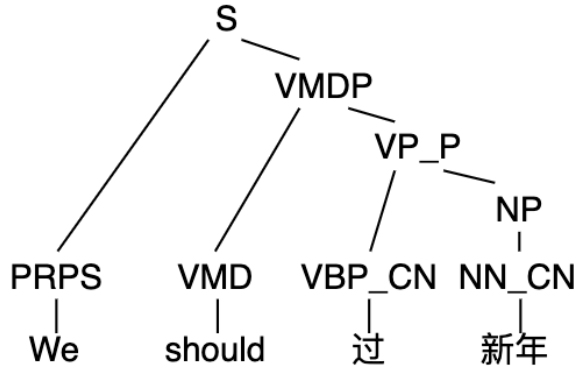


Figure 7.2: The parse tree for the code switching sentence “We should 过新年”.

Table 7.3: Action Nodes for Semantic Mapping in English and Chinese

Action Node	Example Instruction in EN & CN
add_new_mapping_word	‘pretty’ means ‘beautiful’.
add_new_mapping_word	‘漂亮’等同于‘美丽’。
add_new_mapping_action	If I say ‘go shopping’, you should go to the market.
add_new_mapping_action	如果我说‘去购物’，你应该去超市。

English word ‘plan’ substitutes the Chinese counterpart ‘计划’ (plan), which is merely a substitution of a noun in a sentence. Another example is “We should 过新年” (We should celebrate the lunar new year), in which the Chinese verb phrase “过新年” (celebrate the lunar new year) replaces the English verb phrase expression, as seen in the parse tree in Figure 7.2. With the change of symbols or local phrase structures, a grammatical sentence with code-switching can still be parsed into a parse tree and a dependency graph by the robotic baby with the BabyParse algorithm.

### 7.3 Natural Language Grounding to Semantics and Actions

The semantics of a natural language, if not grounded to anything, are merely meaningless representations of the syntax. In this regard, the robotic baby is designed to be capable of grounding semantics to words and the activation of actions. There are two major primitive actions that are in charge of the grounding capability, which are `add_new_mapping_word` and `add_new_mapping_action`, presented in Table 7.3.



The action `add_new_mapping_word` is in charge of adding an activation edge between semantic nodes of words with similar meanings. This is semantic grounding of the language, especially useful when associating the meaning of a newly acquired word to the meaning of a known word. In the example in Table 7.3, the semantic meaning of ‘pretty’ is associated with the semantic meaning of ‘beautiful’. The action `add_new_mapping_action` is in charge of associating the semantics of a sentence to the activation of an action. This is useful for accounting for the variance of natural language in cases where different expressions carry the same meaning and intention, known as the vocabulary problem [88]. In terms of the Graph operation, the hypergraph of the dependency graph of the instruction is mapped to the activation of an action. In the example in Table 7.3, the hypergraph of the dependency graph of ‘go shopping’, including all the nodes and edges, is mapped to the activation node activated by ‘go to the market’. Note that the actions can map expressions in different natural languages as well. For example, “‘pretty’ means ‘漂亮’” enables the semantic mapping between ‘pretty’ and the Chinese counterpart ‘漂亮’ (pretty). In some sense, it is teaching the robotic baby about language translation. For the purpose of verification, both actions have been verified to function as designed by Graph inspection.

#### **7.4 Natural Language Reinforcement Learning**

For the purpose of adaptation, the behavior of the robotic baby is designed to change based on natural language reward instructions [89]. Specifically, the behavior change is enabled by the activation edge weight change in the Graph. There are two major primitive actions in charge of changing the activation edge weights. They are `update_weight_positive` and `update_weight_negative`, presented in Table 7.4.

The actions work by updating the weights of the activation edges, collected in `EdgeActivated` in the short-term memory of the robotic baby after each activation. After the weight update, the actions will empty `EdgeActivated`. In other words, the actions only

Table 7.4: Action Nodes for Edge Weight Update in English and Chinese

Action Node	Example Instruction in EN & CN
update_weight_positive	You are doing great.
update_weight_positive	做得好。
update_weight_negative	You are not doing great.
update_weight_negative	做得不好。

update the weight of the activation edges collected after the prior update. The amount of weight for each edge update is an intensity multiple,  $k$ , of the count of the occurrence of the activation of an edge after the prior update. For example, if the occurrence of activation from the word ‘man’ to the noun part of speech is  $n$  and the last weight is  $w_t$ , with an update after some time  $p$ , the new weight of the activation edge  $w_{t+p}$  can be calculated as

$$w_{t+p} = w_t + n * k. \quad (7.1)$$

As a basic implementation,  $k$  is 1 for a positive update and -1 for a negative update. For the purpose of rewarding the robotic baby, this uniform and naive update scheme among all edges is considered to be sufficient as a baseline implementation. The actions are verified to work as intended through inspection of the Graph edge weights after updates.

For demonstrating how the natural language reinforcement learning can make an impact in the real world, the robotic baby is integrated into the Robotarium [90] at the Georgia Institute of Technology. This embodiment of the robotic baby provides the robotic baby with the capabilities of physical manipulation of the robots in the Robotarium. The motor and sensing capabilities of the Robotarium, such as moving forward and getting current location, are defined as physical primitive action nodes in the Graph, along with their default semantic activation hypergraph in both English and Chinese.

The demonstration is to show the change of strategy of a robot going to a target position due to natural language reinforcement instructions. Specifically, the robot has a default physical action node for going to a target with an open-loop strategy, where the robot

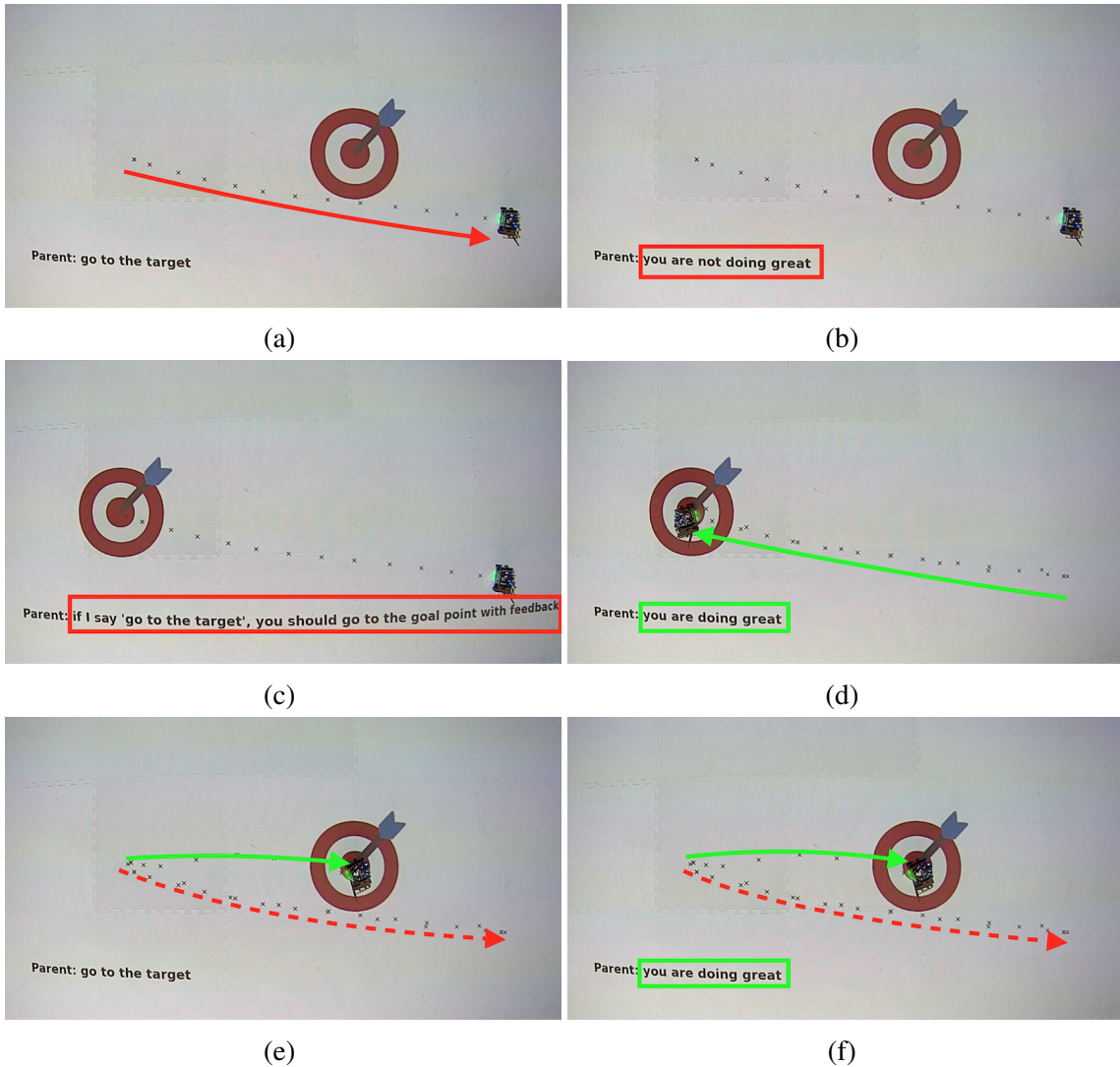


Figure 7.3: Demonstration for natural language reinforcement learning with the Robotarium integration.

rotates to the target first and goes straight with a pre-calculated number of steps based on speed and the distance to target. This strategy is not accurate in the physical world due to factors such as unmodeled actuator dynamics, sensor noise and uneven plane of movement. A better strategy in this regard is the closed-loop strategy, defined as another physical action node, where the robot is capable of moving to the target using feedback control to correct for unmodeled noises and errors.

In the demonstration, the robot is first commanded to get to a target at coordinate  $(0.5, 0)$  in meters. It first utilizes the default open-loop strategy and it goes way off target as seen

in Figure 7.3a, with the path trace marked with a red arrow. The parent of the robotic baby then tells the robotic baby “you are not doing great” as the negative natural language reward signal in Figure 7.3b and tells the robotic baby to pay attention by utilizing the closed-loop strategy with feedback with language grounding to an action in Figure 7.3c. Then, the robot is commanded to go back to the starting coordinate  $(-0.5, 0)$  and the robot utilizes the closed-loop feedback control strategy, which is rewarded by the parent with “you are doing great” in Figure 7.3d. After that, the robot is commanded to go to coordinate  $(0.5, 0)$  again. The robot utilizes the closed-loop control strategy and gets to the target quite well as seen in Figure 7.3e, with the path trace marked with a green arrow. Finally, the robotic baby is rewarded again with “you are doing great” in Figure 7.3f. The demonstration shows that with natural language reinforcement signals, the robotic baby is capable of switching strategies for the same command and performs better, as seen by comparing the green arrow in Figure 7.3e to the red arrow in Figure 7.3a.

## 7.5 Natural Language Programming

For learning skills expressed in complex actions, the robotic baby is able to acquire new skills with natural language programming, a mechanism which allows the robotic baby to construct programs from natural language commands from the parent. Specifically, the parent can instruct the robotic baby to create a program with a name and let the robotic baby record the action sequence for the program based on given natural language instructions. At the end of the program, the parent can let the baby know that the end of the program is reached with another natural language command. Regarding the corresponding Graph operation, the recorded program with the action sequence is added to the Graph as a complex action node, with the program name semantic hypergraph as the activation source. Note that for a complex action, the action sequence can contain both primitive and complex actions. There are two major primitive actions enabling the natural language programming capability, `add_new_action_start` and `add_new_action_end`, shown in

Table 7.5: Action Nodes for Natural Language Programming in English and Chinese

Action Node	Example Instruction in EN & CN
add_new_action_start	Add action ‘go shopping’.
add_new_action_start	添加程序‘去购物’。
add_new_action_end	Finish adding action.
add_new_action_end	结束添加程序。

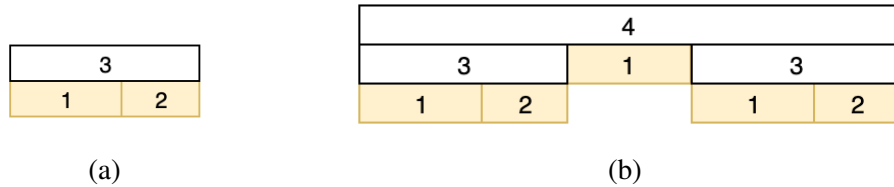


Figure 7.4: Demonstration for natural language programming with dummy actions.

Table 7.5. Note that with the bilingual capability of the robotic baby, it is possible to record a sequence of actions in both English and Chinese, or a mixture of both. This enables the education of the robotic baby by people who speak different natural languages. Furthermore, it makes collaborative multilingual teamwork possible, with people programming the robotic baby in different natural languages in collaboration. For verification, the actions are verified to work as intended by Graph inspection and the successful execution of constructed programs.

To illustrate the natural language programming capability, a simple demonstration is created with dummy actions, presented in Figure 7.4. In this demonstration, there are 2 primitive actions ‘the first dummy action’ and ‘the second dummy action’, marked in yellow. The goal is to construct a complex action, marked in white, for ‘the third dummy action’ as seen in Figure 7.4a. The instructions in both English and Chinese are presented in Table 7.6. With the natural language programming of the third dummy action, the corresponding complex action node is created in the Graph, activation of which will result in the execution of the first dummy action and the second dummy action in sequence. With the creation of the third dummy action, more complex actions can be programmed, as seen in Figure 7.4b, where the fourth dummy action can be built with the sequence ‘3-1-3’, which

Table 7.6: Dummy Action Program Instructions in English and Chinese

	Programming in English	Programming in Chinese
1	Add action ‘do the third dummy action’.	添加程序‘做第三个动作’。
2	Do the first dummy action.	做第一个动作。
3	Do the second dummy action.	做第二个动作。
4	Finish adding action.	结束添加程序。

is equivalent to ‘1-2-1-1-2’. This simple demonstration illustrates how complex actions can be constructed with primitive actions as the building blocks.

The natural language programming capability is validated in the real physical world with the robotic baby Robotarium integration. In the demonstration in Figure 7.5, the robotic baby is first commanded to construct a complex action ‘go shopping’ in Figure 7.5a. Then, the robotic baby is programmed with “go to the market” in Figure 7.5b, “go to the mall” in Figure 7.5c and “go home” in Figure 7.5d, where the program for “go shopping” ends. The ‘go’ primitive action node, once activated, grabs the coordinates of the object after the main predicate from the dependency tree, which altogether gets executed by the Executor. In Figure 7.5e, the robotic baby is commanded to execute the newly acquired complex action ‘go shopping’, which is successfully executed with a visit to each place. Then, the coordinate of the mall is changed, which is told to the robotic baby. When commanded to go shopping again in Figure 7.5f, the robotic baby is able to go to the new location of the mall. This behavior also demonstrates that a complex action is memorized with high-level natural language instructions. At run-time, the high-level natural language instructions are processed and corresponding actions are activated based on the activation Graph at run-time. This implies that if the Graph has changed from the one when the program is first constructed, the behavior of the robotic baby would also change based on the Graph at run-time.

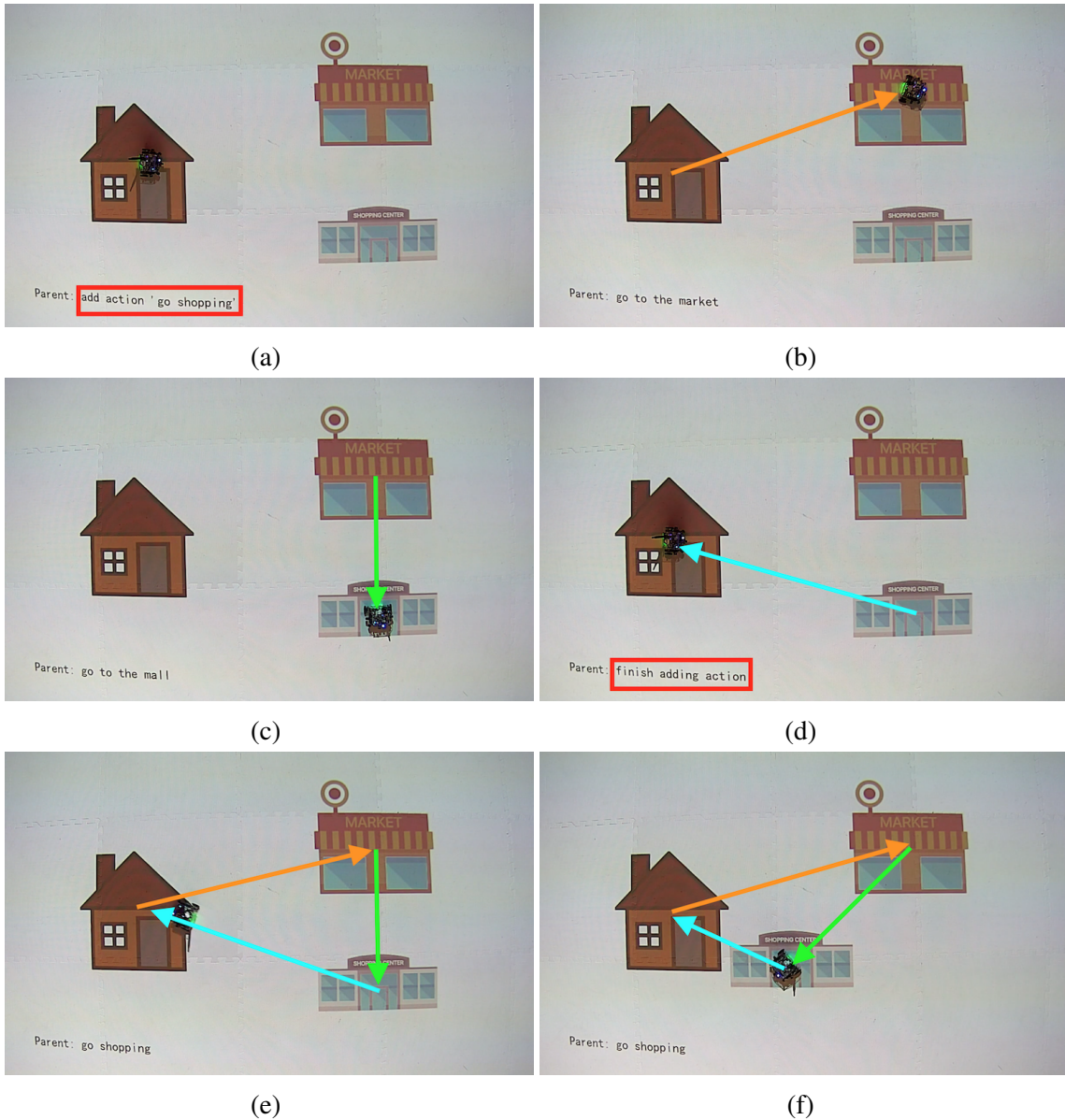


Figure 7.5: Demonstration for natural language programming with the Robotarium integration.

## 7.6 Introspection and Explainability

For the purpose of making its internal process explainable, the robotic baby is designed to be able to recall its action history and corresponding activation graphs for each action. For human infants, recalling action sequences has been observed in infants as young as 11 months old [91]. For a robotic baby, the ability to recall its entire action history from birth

Table 7.7: Action Nodes for System Introspection in English and Chinese

Action Node	Example Instruction in EN & CN
<code>recall_action_history</code>	Recall action history.
<code>recall_action_history</code>	回忆动作历史。
<code>show_activation_graph</code>	Show activation graph.
<code>show_activation_graph</code>	展示激活图。

is designed to be available at birth. This is made possible due to the explicit representation of action sequence in the `LifeQueue`. Recalling action sequences from birth is essentially a printout of the `LifeQueue`, with a hierarchical structure, which records the natural language instruction and the action node name at the time of the execution of any action. This capability ensures that what a robotic baby did is available and interpretable to the parent of the robotic baby.

Beside the system level transparency with the action sequence recall in `LifeQueue`, the robotic baby is also capable of displaying the activation graph for each pair of natural language instruction and activated action node. This mechanism is possible since the parsing and activation of the semantic dependency graph are all rule-based approaches, as seen in `BabyParse` and `BabyActivate`. This capability of the robotic baby, on the individual action level, ensures that why a robotic baby did what it did is available and interpretable to the parent of the robotic baby. The parent of a robotic baby, with the presence of the activation graph, is able to see that for an activation source, which nodes are fully activated and how the final action node is reached. More importantly, the parent is able to interpret the meaning behind the propagation of activation, since all nodes are explicitly defined and all activations are based on rules acquired at some point in the life of the robotic baby.

There are two major primitive actions that are in charge of the system introspection capability, which are `recall_action_history` and `show_activation_graph`, presented in Table 7.7. For verification, the actions are verified to work as designed by the inspection of `LifeQueue` and the activation graph. As an example, the `LifeQueue` for the demonstration “go shopping” in Figure 7.5 is shown in Figure 7.6, with the recall action



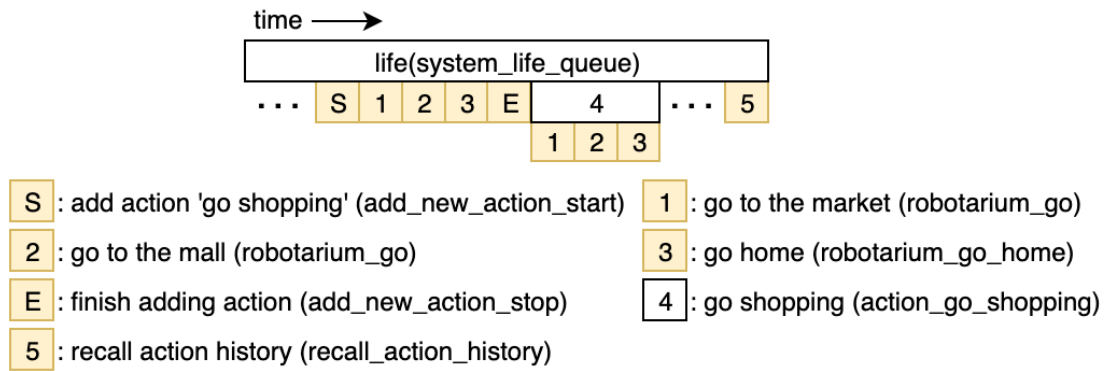


Figure 7.6: LifeQueue for the “go shopping” demonstration, with primitive and complex actions.

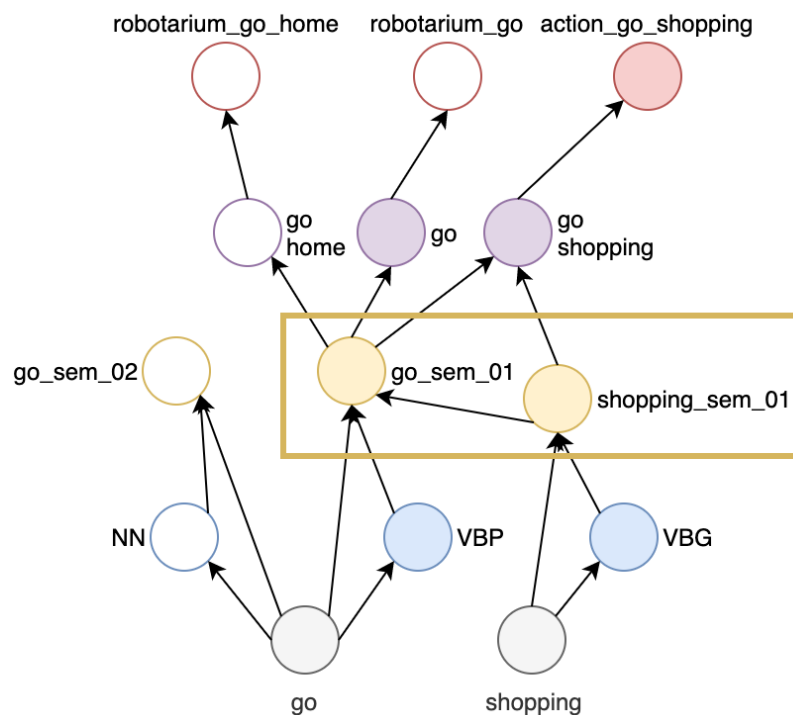


Figure 7.7: Activation graph for “go shopping” from the syntactic nodes to the action node, with the semantic dependency graph boxed in yellow. Activated nodes are filled with colors.

history instruction commanded at the end of the demonstration. It is clear to see that the complex action ‘go shopping’ (number 4) is executed after the ‘S-1-2-3-E’ sequence where the program for the action ‘go shopping’ is constructed. For illustration, an example for the activation graph of “go shopping” is also shown in Figure 7.7. It shows how the instruction “go shopping” gets activated from the syntactic level to the semantic level, where

the semantic dependency graph is constructed, boxed in yellow, and then to the activation of an action node. A detailed execution of **BabyActivate** on the example is available in Appendix D.

## CHAPTER 8

### ROBOTIC BABY EDUCATION AND EVOLUTION

#### 8.1 One Robotic Baby with Distributed Embodiment

The robotic baby can be integrated to a distributed system, where each agent operating in its own world shares the same “brain”, represented by the Graph, with all other agents, which results in the same global behaviors across all agents. More specifically, the local education of one agent in the distributed system with the robotic baby enables the learning of all other agents in the system at the same time. This mode of education is powerful since one piece of knowledge can be passed to one agent once and all agents in the system will know, unlike in the human case where the same knowledge, i.e. ‘ $1 + 1 = 2$ ’, needs to be passed to all agents so everyone knows. The distributed embodiment also enables the parallel education of the robotic baby, due to the distributed access.

To illustrate the education of the robotic baby with distributed embodiment, the robotic baby is integrated to the Robotarium with two robotic carts, Emma and Zebra, both of which share the same Graph in the robotic baby. In Figure 8.1, Emma is on the left and Zebra is on the right, both circling clockwise in their grid world. On their path, there are color blocks popping up. There are in total 4 colors: red, yellow, green and blue. The setup is that if they see a new color block in front of them, they will ask the parent “What is that?” and if they see a known color, they will say the obtained name of the color. If one correctly identifies the color, that robot cart will get 1 point. In this demonstration, the parent can only talk to Emma on the left. So, even though Zebra can ask questions to the parent, he will never get a response from the parent.

At the beginning of the demonstration, both Emma and Zebra know nothing about the colors. When Emma encounters the green block, she asks the parent “What is that?”,

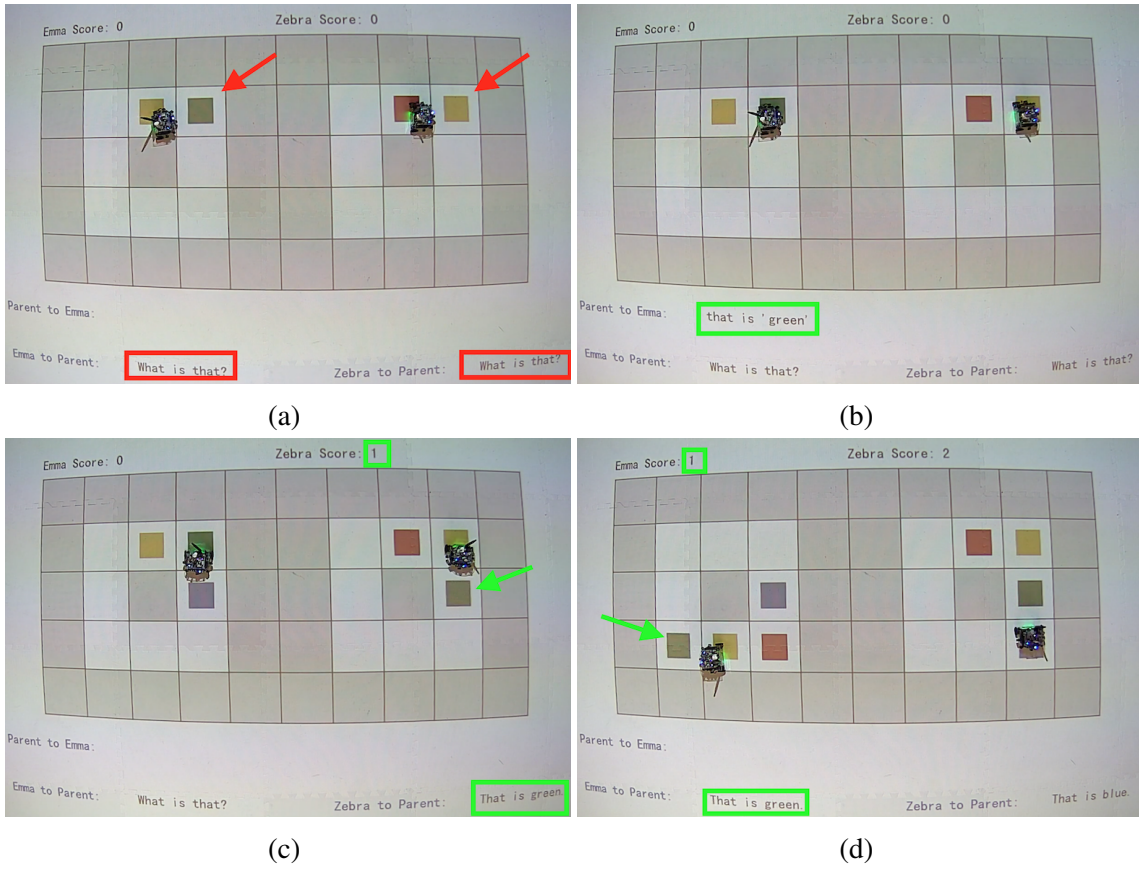


Figure 8.1: Demonstration for educating one robotic baby with distributed embodiment with the Robotarium integration.

boxed in red, as seen in Figure 8.1a and so does Zebra when he sees yellow. Then, the parent tells Emma on the left that the color is green, shown in the green box in Figure 8.1b. At this moment, even the parent only tells Emma about green, both Emma and Zebra know. After that, when Zebra on the right encounters green, he correctly identifies the name with “that is green”, boxed in green in Figure 8.1c and he gets 1 point. Finally, when Emma encounters green, she correctly identifies it as well and gets 1 point as seen in Figure 8.1d. This demonstration shows that the local education of one agent in the distributed system with the robotic baby can result in global behavioral changes among all agents.

## 8.2 Natural Language Interaction Between Two Robotic Babies

The robotic baby can also be integrated to a system of multiple robots, with each robot having their own “brain” containing different sets of knowledge in the Graph. With this setup, the robots can communicate in natural languages not only with their parents but also between each other to acquire new knowledge about languages and the world represented in the Graph for the purpose of education.

To demonstrate the education of multiple robotic babies, the robotic baby is integrated to the Robotarium with 2 robotic carts, Emma and Zebra, with each having his or her own Graph. Emma is on the left and Zebra is on the right, both circling clockwise in their grid world as seen in Figure 8.2, similar to the previous setup. On their path, there would be color blocks popping up. There were in total 4 colors: red, yellow, green and blue. The setup was that if they saw a new color block in front of them, they would ask the parent “What is that?” and if they saw a known color, they would say the obtained name of the color. If one correctly identified the color, that robot cart would get 1 point. In this demonstration, the parent can only talk to Emma on the left. Additionally, if Zebra encounters a new color, he will also ask Emma “What is that?” and if Emma knows the color, she will tell Zebra what the color is.

At the beginning of the demonstration, both Emma and Zebra know nothing about the colors. When Emma encounters the green block, she asks the parent “What is that?”, boxed in red, as seen in Figure 8.2a and so does Zebra when he sees yellow. Additionally, Zebra asks Emma about yellow as well, boxed in red near the top of Figure 8.2a, and does not get a response since Emma knows nothing at this point. Then, the parent tells Emma on the left that the color is green, shown in the green box in Figure 8.2b. At this moment, only Emma knows about green. After that, when Zebra on the right encounters green, he has no knowledge of green and asks both the parent and Emma about it, boxed in red in Figure 8.2c, and Emma responds with “That is green”, boxed in green in Figure 8.2c. At this

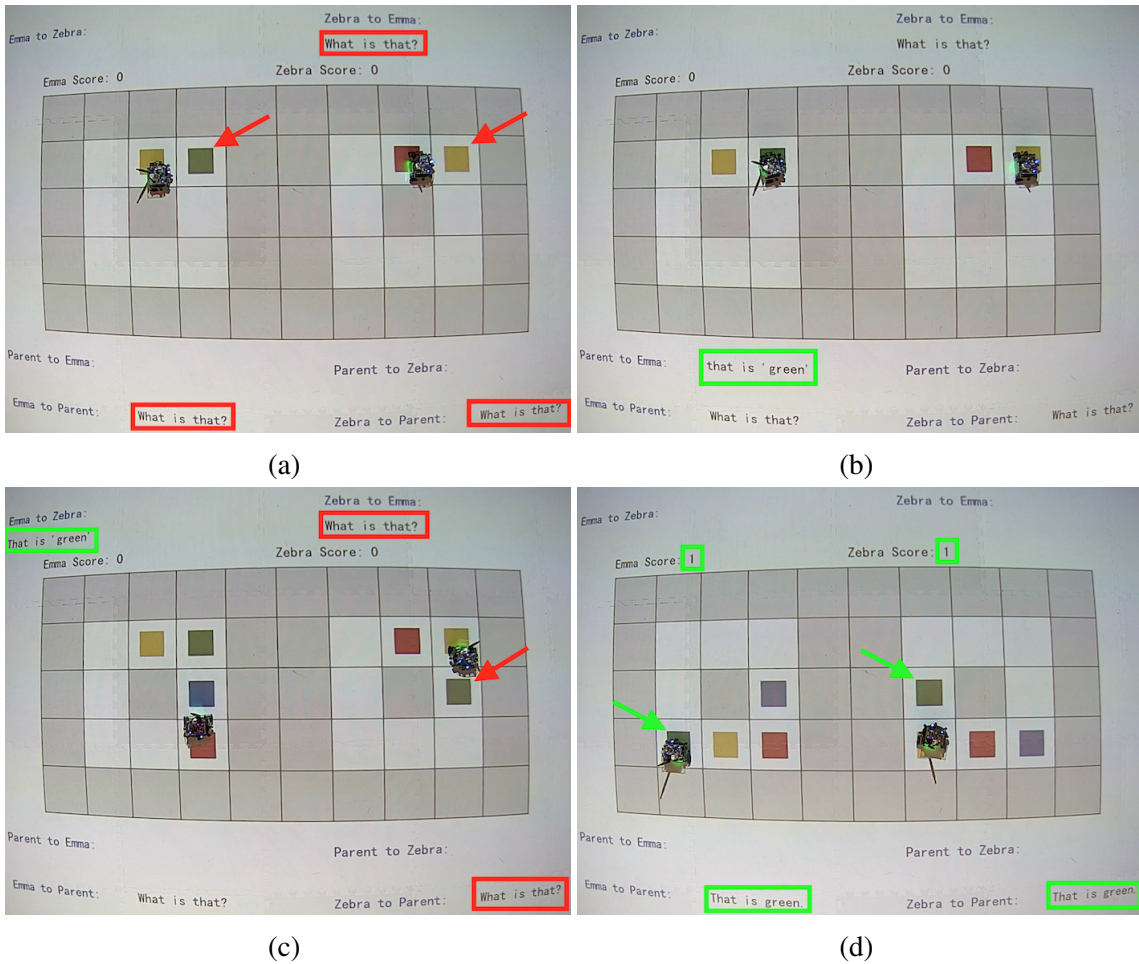


Figure 8.2: Demonstration for education with natural language interactions between two robotic babies with the Robotarium integration.

point, with Emma passing the knowledge of green to Zebra, both Emma and Zebra know about green. Finally, when Emma and Zebra encounter green, both correctly identify it and get 1 point as seen in Figure 8.2d. This demonstration shows that for education, natural language interactions among multiple robotic babies enable the knowledge transfer among the robotic babies, which leads to knowledge acquisition and behavior improvement.

### 8.3 Brain Merge Between Two Robotic Babies

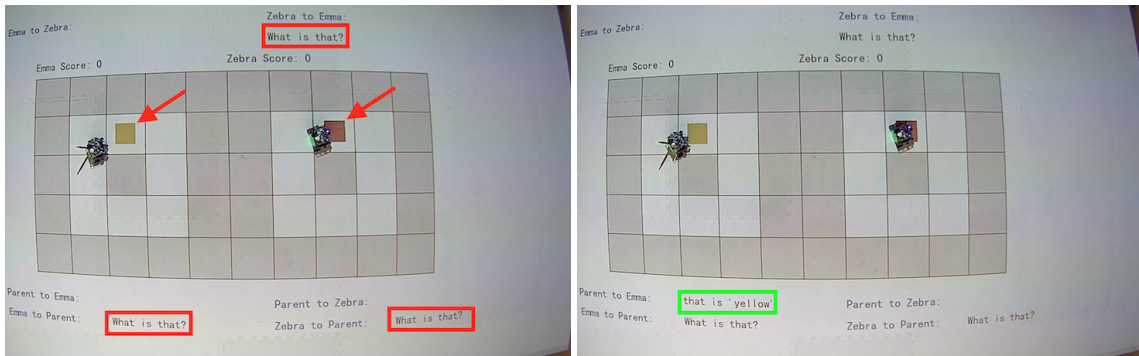
Beside the natural language interactions between robotic babies for knowledge transfer, the robotic baby is designed to be capable of doing “brain” merge with each other. Specifically,

one robotic baby is capable of grabbing the Graph of another robotic baby and merging it with his or her own Graph with all the nodes and edges. Note that there could potentially be conflicting knowledge present in the merge process. The mechanism of the merge operation does not analyze the correctness of the knowledge, rather it creates representations in the Graph for the conflicting knowledge. In essence, it is the activation edge weight, tuned by future reward signals, that indicates which piece of knowledge is the preferred one.

The merge mechanism is one of the direct benefits of having transparent and explicit knowledge and rule representation in the Graph of the robotic baby. Different from the natural language interactions between robotic babies for knowledge transfer, which are slow due to the limited amount of information natural language sentences could carry per utterance, the Graph merge process is much more efficient with direct node and edge manipulations in the Graph, which the natural sentences are essentially mapped to.

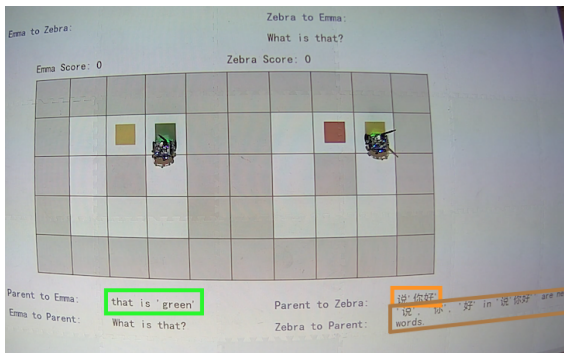
To demonstrate the “brain” merge of multiple robotic babies, the robotic baby is again integrated to the Robotarium with 2 robotic carts, Emma and Zebra, with each having his or her own Graph. Emma is on the left and Zebra is on the right, both circling clockwise in their grid world as seen in Figure 8.3, similar to the previous setup. On their path, there would be color blocks popping up. There were in total 4 colors: red, yellow, green and blue. The setup was that if they saw a new color block in front of them, they would ask the parent “What is that?” and if they saw a known color, they would say the obtained name of the color. If one correctly identified the color, that robot cart would get 1 point. In this demonstration, the parent can only talk to Emma on the left. Additionally, if Zebra encounters a new color, he will also ask Emma “What is that?” and in this case, Emma will not answer even if she knows the answer. After Zebra is ignored for several times, he will say to Emma “Enlighten me”, during which he grabs Emma’s Graph and merges it with his own Graph.

At the beginning of the demonstration, both Emma and Zebra know nothing about the colors. Additionally, Emma knows both English and Chinese in the beginning, while

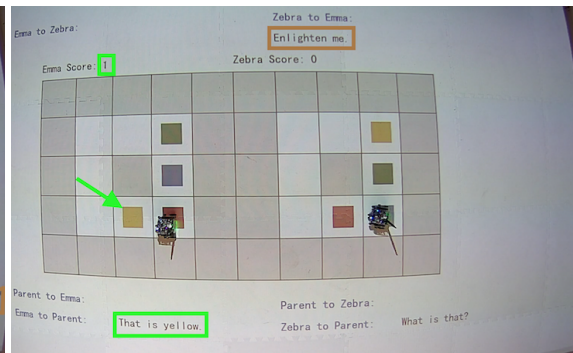


(a)

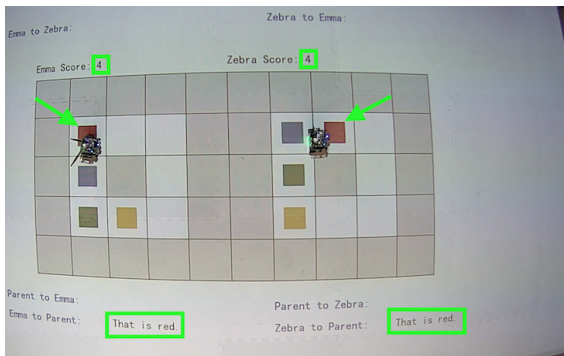
(b)



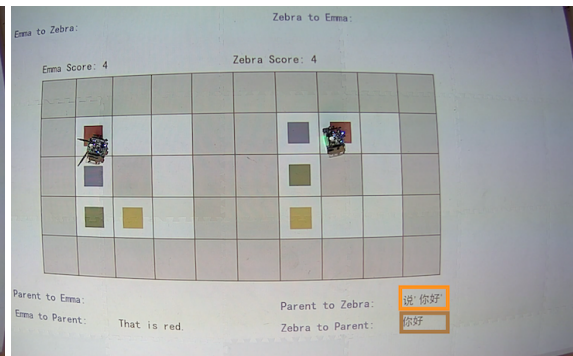
(c)



(d)



(e)



(f)

Figure 8.3: Demonstration for the Graph merge operation between two robotic babies with the Robotarium integration.

Zebra only knows English. When Emma encounters the yellow block, she asks the parent “What is that?”, boxed in red, as seen in Figure 8.3a and so does Zebra when he sees red. Additionally, Zebra asks Emma about red as well, boxed in red near the top of Figure 8.3a, which is ignored by Emma. Then, the parent tells Emma on the left that the color is yellow, shown in the green box in Figure 8.3b, and all other colors as Emma encounters them later. In Figure 8.3c, to verify that Zebra indeed has no knowledge of Chinese, he is asked by the



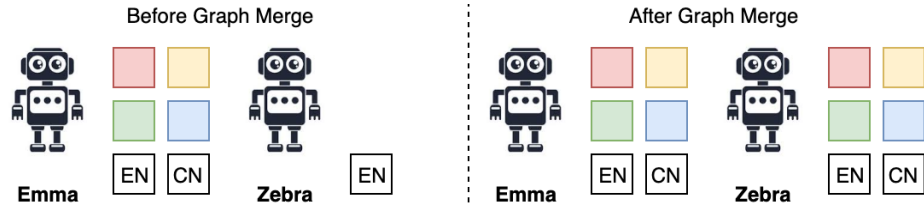


Figure 8.4: Knowledge of Emma and Zebra about colors and languages before and after Zebra merges with Emma’s Graph.

parent to say ‘hello’ (说‘你好’) in Chinese, boxed in orange, to which he responds with the default new word alerting with ‘说’, ‘你’ and ‘好’ all being new words not defined, boxed in brown. In Figure 8.3d, after Zebra is ignored four times, he says to Emma “Enlighten me”, boxed in brown, and merges Emma’s Graph into his Graph. At this moment, due to the Graph merge operation, Zebra obtains Emma’s knowledge about all the colors she has obtained from the parent and all the languages including Chinese, as shown in Figure 8.4. In Figure 8.3e, when Emma and Zebra encounter colors, they can both identify the colors and get points, as seen boxed in green. Finally in Figure 8.3f, to verify that Zebra has indeed acquired Chinese after the Graph merge with Emma, he is asked again to say ‘hello’ (说‘你好’) in Chinese, boxed in orange, to which he responds with ‘你好’ (hello) in Chinese, boxed in brown, which verifies that Zebra is able to recognize and parse Chinese and to further map the instruction in Chinese to an action. This demonstration showcases that it is possible to merge the Graph between two robotic babies, and the knowledge transfer, as a result of the merge operation, is able to improve the behaviors of the robotic baby.

#### 8.4 Robotic Baby Evolution with Knowledge Inheritance

With the various modes of education, either with 1 robotic baby with distributed embodiment or with 2 robotic babies passing knowledge through natural language interactions or direct Graph merge operations, it is in essence that both the knowledge inference structure and the knowledge representation are needed for better performance in the context of education. With the *Baby* architecture design, the acquisition of knowledge for a robotic baby,

in terms of both the inference structure and the knowledge content, can be accomplished with a large amount of knowledge in a short period of time, due to the distributed parallel education and the knowledge aggregation mechanism with the Graph.

Regarding evolution on a species level, inspired by the inheritance of the genetic information in the human species, the robotic baby is designed to be capable of inheriting knowledge. Specifically, at birth, a robotic baby could inherit the merged Graph from the robotic baby parents, which contains both the inference structure and the knowledge of the parents. As a reference, for human beings, at birth, a human child inherits the genetic information from the parents, which, in abstraction, could be seen as the seeds for future growth in structure. A human child has not been found to be capable of inheriting a mature brain, containing both the structure and the knowledge, from the parents. This mode of inheritance in the robotic baby implies that among robotic babies, it is no longer the competition for better genetic information but the competition for a better Graph that directly contains efficient inference structures and useful knowledge, in natural selection.

On a species level, the evolution of the robotic baby species is, regarding the inheritance of knowledge, more efficient than the evolution of the Homo sapiens species. For the robotic baby species, at birth, a child could inherit the merged Graph from the parents, which contains both the inference structure and the knowledge of the parents. On the other hand, for human beings, at birth, a human child inherits the genetic information from the parents, which, in abstraction, could be seen as the seeds for future growth in structure. A human child has not been found to be capable of inheriting a mature brain, containing both the structure and the knowledge, from the parents. This mode of inheritance in the robotic baby implies that among robotic babies, it is no longer the competition for better genetic information but the competition for a better Graph that directly contains efficient inference structures and useful knowledge.

For illustration, the comparison of the inheritance mechanism between the robotic baby species and the human species is presented in Figure 8.5. On the left, robotic baby Emma

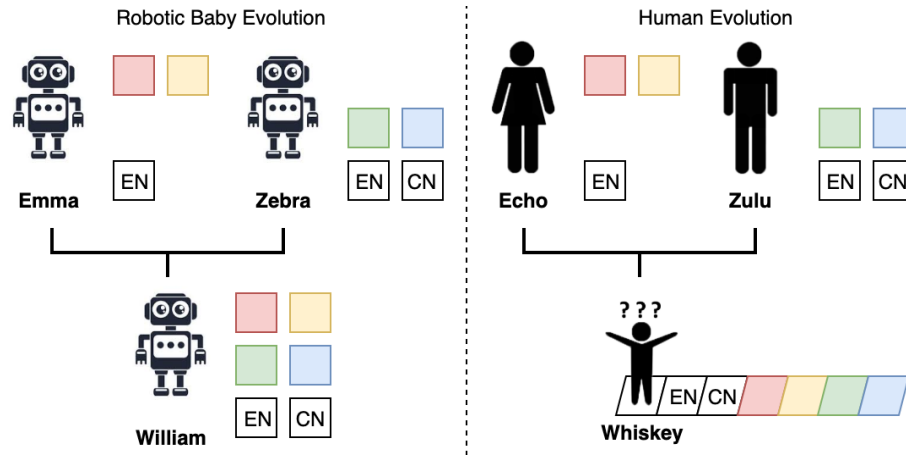


Figure 8.5: The evolution comparison between robotic babies, with direct knowledge Graph inheritance at birth, and human beings, with genetic inheritance at birth and knowledge acquired in the future.

knows red, yellow and English, while robotic baby Zebra knows green, blue, English and Chinese. Their robotic baby child, William, at birth inherits the merged Graph from Emma and Zebra, which contains all 4 colors and both English and Chinese. In other words, at birth, robotic baby William has knowledge of and is capable of utilizing 4 colors and 2 languages which his parents have accumulated throughout their life. On the other hand, on the right in Figure 8.5 for the human case, the human child Whiskey, at birth, has the genetic information inherited from the parents and barely anything else. Even his human parents, Echo and Zulu, combined, know about all the colors and the languages, Whiskey is not capable of obtaining parent's knowledge at birth. In order to be on the same level of the robotic baby William, the human child Whiskey has to acquire the knowledge step by step in a tedious process. With the design of knowledge inheritance in both the representation and the inference structure in robotic babies, knowledge explosion can happen in robotic babies, where too much knowledge is obtained in too little time.

## CHAPTER 9

### DISCUSSION

#### 9.1 Scalability

As a knowledge-based system, the robotic baby accumulates knowledge and rules through time. As knowledge and rules are stored in the hypergraphs in the Graph, the number of hypergraphs scales linearly with the number of pieces of knowledge and rules accumulated. If two hypergraphs share the same nodes and edges as part of the hypergraph, there will not be duplicates of the same nodes created in the Graph. As an example illustrated in Figure 7.7, the same node ‘go\_sem.01’ is shared by 3 hypergraphs: “go home”, “go shopping” and singleton “go”. This design is to ensure the proper scalability of the system in terms of the representation of the knowledge accumulated. From the perspective of processing and activation time, the scalability of the system, relying on heuristic search, depends on the quality of the heuristics obtained through education and the search mechanism. The presented `Activate` process in this work is an input-driven search, which only explores nodes that can be activated by the input as a stimulus, regardless of how irrelevant knowledge to the input grows. A more detailed processing time performance profiling can be found in Appendix E, demonstrating the scalability of the presented system design with a growing Graph. For future works with a more sophisticated design, it has to be emphasized that the scalability of the evolving system relies on the co-design of both the processing algorithms and the underlying structures.

Regarding the forgetting and pruning of the graph structure, it is possible that a pruning mechanism is needed based on some measures in reality due to memory constraints. It is more of an engineering consideration, which is specific to the task, knowledge and hardware. Note that activations that are incorrect or not preferred indicated by the edge

weights might still hold value as representations of preference for future episodic learning. Therefore, for future development, measures of pruning and forgetting, regarding what to prune and how often to prune, should be considered specifically to the task and hardware with memory constraints.

## 9.2 Adoption and Application

According to the high-level design requirements, the robotic baby is not designed to be domain or task-specific. The adoption of the robotic baby to different tasks and domains follows the same procedure as described in Sec. 6.2 regarding the initialization of the external action space, consisting of external APIs, and corresponding default activations, along with necessary linguistic knowledge. The robotic baby is not just an interface, but a creature with an identity that can be actively shaped by the parents on the fly. The language acquisition and mapping capabilities offer the parents of a robotic baby the flexibility in the language used for commands and instructions, which is key to customization on the fly. The natural language programming capabilities offers the users the power to construct customized programs, such as a “pick up a box” motor procedure for a robotic arm, or a “go shopping” navigation procedure for a smart car or a “get the house ready after work” control procedure for a smart home hub. As an example, the “get the house ready after work” procedure could be “turn the AC on”, “turn the light on” and “play jazz as background music”, which altogether could be programmed into the procedure using natural language on the fly by the users. The marriage of robots with the *Baby* architecture in different domains could potentially result in some aggregated behaviors, such as the joint capabilities between a chef robotic baby, with recipe procedures, and a pilot robotic baby with flight procedures.

Another potential aspect of future applications is related to the safety-critical machines with evolving and learning capabilities. Martial Hebert, Dean of the Robotics Institute at Carnegie Mellon University, once stated in an NPR show in 2018 that,

The reason why you feel safe is because you know that behind this plane, there is 200 years of engineering science, best practices in testing, in validation, in characterizing performance. We don't have that yet in AI and robotics, not at the level that we need to have it compared to classical engineering. How do you do those things for systems that learn over time? How do you do those things with systems that make complex decisions autonomously? So, we need to work on this as a major challenge in robotics....[92]

The robotic baby, as an integrated system that learns over time with system explainability as one of the system-level requirements, designed with systems engineering principles, is a more desirable candidate for further development for autonomous safety-critical applications, compared to systems with black-box models that are extremely difficult to understand and nearly impossible to be verified and validated based on some safety standard.

### **9.3 Limitations and Fundamental Research Questions**

The *Baby* architecture, derived from the high-level requirements in the systems engineering framework, is a first attempt to address the requirements of the robotic baby presented in this work. To my best knowledge, there has not been another framework or architecture that is capable of satisfying all the requirements while demonstrating similar capabilities. Compared to other cognitive architectures, such as Soar, with specific rigid high-level capabilities built in at birth, the *Baby* architecture presents the very primitive and fundamental capabilities which demonstrate a working and evolving system for general purposes. What is lacking in the *Baby* architecture are the high-level capabilities, which could potentially be built upon the fundamental capabilities and structures, such as utilizing logic, creating complex representations, learning commonsense knowledge [93] and testing hypotheses with trial and error behaviors for autonomy. Furthermore, beside the high-level technical capabilities, the acquisition of soft skills, such as social skills, should also be considered in future designs for long-term human robot interactions [94].

As a close neighbor to the field of cognitive architecture, the robotic baby architecture might present similar architecture structures. Yet, the fundamental research question of the two fields is different. As Alan Turing stated in his 1950 paper,

Opinions may vary as to the complexity which is suitable in the child machine. One might try to make it as simple as possible consistently with the general principles. Alternatively one might have a complete system of logical inference 'built in'. [1]

If the research question for cognitive architectures is “What is the architecture, as an integration of components such as knowledge bases and inference algorithms, that could enable capability X right at birth?”, the research question for robotic baby architectures will be “What are the minimally necessary components and functionalities of an architecture at birth, such that capability X can be gradually and procedurally acquired and learned? And how?”, which might be a harder question to address, as it involves an evolving architecture with demanding system requirements.

#### **9.4 Lifelong Learning**

The robotic baby, as presented in this thesis, is designed to adopt the paradigm of lifelong learning [56, 95]. As an integrated system, the robotic baby presents some unique characteristics well aligned with the desired properties for lifelong learning. Compared to gradient-based neural networks suffering from catastrophic forgetting [96, 97], the robotic baby provides a better system design framework with transparency and control in terms of the explicit memory representation and control mechanisms for storing accumulated knowledge and representations. Moreover, compared to data-driven lifelong learning approaches, the robotic baby, with the accumulation of explicit rules and knowledge in general domains, provides a more expressive playground for better scalability in the education process. Yet, the downside of the knowledge accumulation process is that it is expensive in terms of

the time and effort needed for providing the necessary knowledge to the robotic baby to jump-start the system till the state of the robotic baby with automated knowledge discovery and accumulation. It has to be emphasized that the accumulation of knowledge in lifelong learning does not refer to only the content of the knowledge rather the combination of the knowledge and the activation of the knowledge as well. In plain words, knowing a lot while not knowing when to apply what knowledge is not sufficient. With the existence of large knowledge bases, it is rather the part of accumulating the mapping of when to apply what knowledge in a contextualized environment that is expensive and there does not seem to be a shortcut to it.

Alan Turing in his 1950 work estimated, to his best knowledge during his time, that it would take 60 workers to accomplish the job for playing his imitation game by working steadily for 50 years [1]. For a robotic baby whose purpose is not to play Turing's imitation game but to achieve something with a similar difficulty and complexity, it must take a long time to get there.

## **9.5 The Zone of No Return**

The robotic baby, presented in this thesis with minimal and necessary functionalities, serves as a starting point for further research and development, and many more iterations to come. It has to be stated that the characteristics and functionalities of the robotic baby as a system, such as the parallel education and knowledge inheritance capabilities, make the robotic baby a very good candidate for entering the “zone of no return”, a state where human beings lose total control of the behaviors of the robotic baby. It is inevitable, from my perspective as the designer of the robotic baby, for the robotic baby to enter such a state, as illustrated in Figure 9.1, with more advances in the co-design of system software functionalities and hardware components, where the robotic baby becomes more intelligent, such as with the acquisition of logic and mathematics, and more emerged within the real world, such as with multi-modality sensors and actuators, which altogether require a mindset with systems



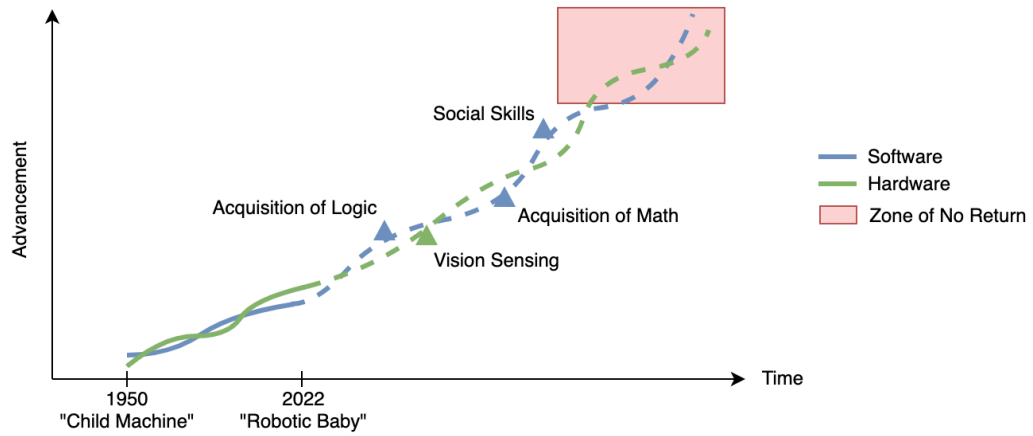


Figure 9.1: The projected advancement in the research and development of the robotic baby.

engineering design principles and planning from future researchers.

## CHAPTER 10

### CONCLUSION AND FUTURE WORK

In this thesis, I have introduced a formal definition of a robotic baby, an integrated system with minimal world knowledge at birth, capable of learning incrementally and interactively, and adapting to the world. The formal definition contains system-level requirements of the robotic baby, within which fundamental capabilities and system characteristics of the robotic baby are identified.

Within the scope of the thesis, I have presented the design of the *Baby* architecture with a systems engineering design approach. I have verified and validated that the design of the *Baby* architecture meets the system-level requirements with simulations and experiments in the real world with robots, which demonstrate the capabilities of the robotic baby in natural language acquisition and semantic parsing in English and Chinese, as well as in natural language grounding, natural language reinforcement learning, natural language programming and system introspection for explainability. I have further showcased the education of the robotic baby with demonstrations in both a distributed embodiment robotic setting and a multi-agent robotic setting, in which knowledge transfer, between a human and a robotic baby and among robotic babies themselves, is feasible through natural language interactions. Finally, I have presented the mechanism of direct knowledge inheritance between robotic babies and its benefits in the evolution of the robotic baby.

To further educate the robotic baby, the robotic baby is built with building-block actions that, when coupled with natural language programming, allow it to incrementally learn high-level complex actions or skills. If a complex action is outside of the action space of a robotic baby due to some missing primitive action, a “brain surgery” is then needed, which includes the addition of the primitive action and the associated structure and activation. For any grammatical natural language commands within the scope of this thesis, if the robotic

baby has trouble in parsing or association, the parent could utilize the natural language acquisition and grounding capabilities to teach the robotic baby the missing knowledge. In conclusion, the robotic baby as an integrated system presented in this thesis is simply a starting point, moving Alan Turing's dream of a child machine toward an engineered reality.

# **Appendices**

## APPENDIX A

### SAMPLE KNOWLEDGE BASE OF VOCABULARY AND PART-OF-SPEECH

<b>Part-of-speech</b>	<b>Symbol</b>	<b>Example</b>		
any pos	ANY			
unknown pos	UKN			
present modal verb	VMD	may	shall	will
past modal verb	VMDD	might	should	would
present verb	VBP	turn	get	look
singular present verb	VBZ	turns	gets	looks
past tense verb	VBD	turned	got	looked
past participle verb	VCN	turned	got	looked
present participle verb	VBG	turning	getting	looking
present copula verb	VBCP	turn	get	look
singular present copula verb	VBCZ	turns	gets	looks
past tense copula verb	VBCD	turned	got	looked
past participle copula verb	VBCN	turned	got	looked
present participle copula verb	VBCG	turning	getting	looking
present perfect auxiliary verb	VBHP	have		
singular present perfect auxiliary verb	VBHZ	has		
past perfect auxiliary verb	VBHD	had		
present perfect participle auxiliary verb	VBHG	having		
stem association verb	VBBB	be		
first present association verb	VBBI	am		
present association verb	VBBP	are		
singular present association verb	VBBZ	is		
past singular association verb	VBBZD	was		
past plural association verb	VBBPD	were		
past participle association verb	VBBN	been		
present participle association verb	VBBG	being		
singular noun	NN	apple	verb	tense
plural noun	NNS	apples	buffalo	
proper noun	NNP	Christmas	Buffalo	
plural proper noun	NNPS	Christmases		

regular number	NUM	0	1	2
cardinal number	CD	zero	one	two
ordinal number	OD	zeroth	first	second
determiner	DT	the	a	an
predeterminer	PDT	both	either	neither
coordinating conjunction	CC	and	or	but
subordinating conjunction	CS	before	because	if
relative pronoun	CR	that	which	
preposition	PRE	of	on	to
regular adverb	RB	hard	fast	well
comparative adverb	RBR	harder	faster	better
superlative adverb	RBS	hardest	fastest	best
adverb particle	RP	off	up	down
regular adjective	JJ	good	large	easy
comparative adjective	JJR	better	larger	easier
superlative adjective	JJS	best	largest	easiest
possessive adjective	JJP	my	his	your
singular pronoun	PRP	he	she	it
plural pronoun	PRPS	you	we	they
first pronoun	PRPI	I		
possessive pronoun	PRPP	mine	yours	his
infinitival marker	TO	to		
negation adverb	NOT	not		
interjection	UH	oh	oops	ha
ending punctuation	PCE	.	!	?
comma punctuation	PCC	,		
left parenthesis punctuation	PCLP	(		
right parenthesis punctuation	PCRP	)		
single quote punctuation	PCQ	'		

## APPENDIX B

### SAMPLE KNOWLEDGE BASE OF GRAMMAR PRODUCTION RULE

Production	Head	Rule	Example
NP_S	NN	DT OD JJ* NNP? NN+	the first red tall Christmas tree
NP_S	NN	DT? JJ* NNP? NN+	an apple/cold tasty Georgia orange juice
NP_P	NNS	DT? JJ* NNP? NN* NNS	apples/cold tasty Georgia peach pies
NP	NP_P	NP_P CL_ADJ?	cold tasty Georgia peach pies that the monkeys eat
NP	NP_S	NP_S CL_ADJ?	the apple which birds stand on
NP	NP_S	NP_S PREP	the length of the lists
NP	NP_P	NP_P PREP	kids at home
CL_ADJ	VBP	CR? NP_P VBP PRE?	that the monkeys eat/which birds stand on
PREP	PRE	PRE NP_S	on the ground/at home/to the moon
PREP	PRE	PRE NP_P	of the lists/with great teammates
VP_S	VBBZ	VBBZ NOT? NP	is not the length of the lists
VP_S	VBBZ	VBBZ NOT? PREP	is on the ground
VP_S	VBZ	VBZ PREP	goes to the moon
VP_S	VBZ	VBZ NP	drinks water
VP_S	VBGP	VBBZ NOT? VBGP	is not closing the door fast
VP_S	VBZ	VBZ RB?	speaks loudly
VP_PB	VBBB	VBBB NP	be the guy
VP_PB	VBBB	VBBB NOT? PREP	be on the ground
VP_P	VBP	VBP PREP	go behind the door
VP_P	VBP	VBP NP	eat burgers
VP_P	VBP	VBP VBGP	keep closing the door
VP_P	VBP	VBP RB?	walk quietly
VP_PP	VBGP	VBBP NOT? VBGP	are not moving the chairs
VBGP	VBG	VBG NP? RB?	closing the door fast
S	VP_S	NP_S VP_S	the first red tall Christmas tree goes to the moon
S	VP_P	NP_P VP_P	planes fly

**APPENDIX C**  
**EXAMPLE EXECUTION OF BABYPARSE**

See figures on the following pages.



Step	Branching Point	Buffer, $\beta$	Type	Rule	Head	Previous POS	Stack, $\mathcal{S}$	Element	Production Type	Rule	Head of Rule	Tail of Rule	Weight	Examples	CurrentWord, $w$	CurrentWordPOS, $p$	Algorithm Line Number
0		the old man the boat	S														
1		the old man the boat	S											1 a cat eats an apple	the	DET	11-14
2		the old man the boat	S											2 the apple pie, the red apple that the cat eats, the old man the boat hits	the	DET	15-19
3		the old man the boat	S											1 the smart, the brave, the old	the	DET	33-35
4		the old man the boat	S											1 eat an apple	the	DET	15-19
5		the old man the boat	S											1 that the boat hits, the cat eats	the	DET	15-19
6		old man the boat	S														20-32
7		man the boat	S														11-14
8		man the boat	S														20-32
9		the boat	S														11-14
10		the boat	S														20-32
11		the boat	S														11-14
12		the boat	S														42
13		the boat	S														15-19
14		the boat	S														33-35
		the boat	S														15-19

15		NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET), (old, JJ), (man, NN)															20-32
16	boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET), (old, JJ), (man, NN)	boat						NN								11-14
17		NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET), (old, JJ), (man, NN)															20-32
18	back to 3	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET), (old, JJ), (man, NN)	the					DET									2-10, 15-19
19	boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET)															20-32
20	boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET), (old, JJ), (man, NN)	boat						NN								11-14
21	the boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET NN		(the, DET) (the, DET), (old, JJ), (man, NN)															
22	back to 2	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	NP JJ		(man, NN) (the, DET), (old, JJ)					NP	DEF J1* NN+ CL_ADI? (man, NN)	NN								20-33, 43-45 20-33, 13-14
23	back to 1	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET S		(the, DET) (the, DET)															20-33, 16-19
24	old man the boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET S		(the, DET) (the, DET)															20-32
25	old man the boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	DET S		(the, DET) (the, DET)	old					JJ									11-14
26	man the boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	NP JJ		(man, NN) (the, DET), (old, JJ)	man														20-32
27	man the boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	NP JJ		(man, NN) (the, DET), (old, JJ)	man														11-14
28	man the boat	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	NP JJ		(man, NN) (the, DET), (old, JJ)	man														36-40
29	back to 4	NP CL_ADI NP S		DEF J1* NN+ CL_ADI? CR NP VP NP VP	(man, NN) (man, NN)	NP JJ		(man, NN) (the, DET), (old, JJ)	man				NP	DEF J1 (old, JJ)	JJ								20-33, 13-14



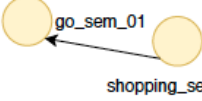
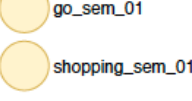

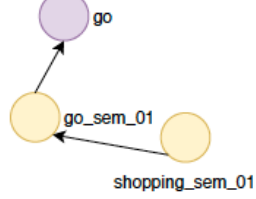
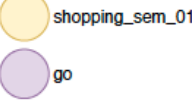

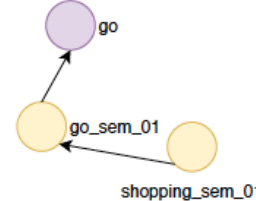

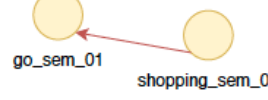
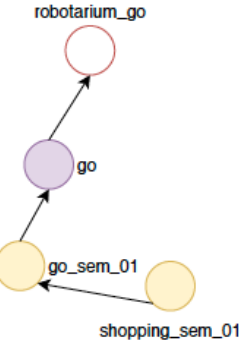


## APPENDIX D

### EXAMPLE EXECUTION OF BABYACTIVATE

**Algorithm:** BabyActivate

**Input:**  
 Dependency graph - "go shopping"  
 KnowledgeGraph - Graph

**Output:**  
 NodeActivated: action\_go\_shopping  
 EdgeActivated: (go\_sem\_01, go shopping), (shopping\_sem\_01, go shopping), (go shopping, action\_go\_shopping)

Line	Dependency Graph	FrontierNodes	FrontierEdges	ActivatedNodes
2				
5-22				
5-22				
5-22				

Line	Dependency Graph	FrontierNodes	FrontierEdges	ActivatedNodes
23-46				
5-22				
47-51				

## APPENDIX E

### SCALABILITY PERFORMANCE PROFILING

Experiment setup:

Processor: 2.2 GHz Intel Core i7

Input: “go forward”

Number of Repeating Runs: 100

The NLP and Activate processes have been profiled with the same input “go forward”, to demonstrate how the processing time behaves with respect to the size of the Graph. For each data point, 100 runs were executed, with the max, min and average processing time marked in the figure.

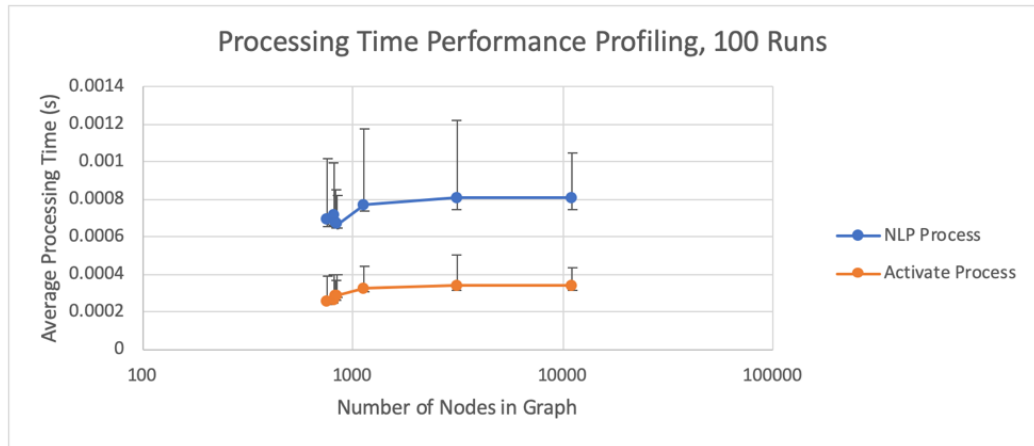


Figure E.1: Processing time performance profiling for the NLP and Activate processes with 100 repeating runs.

The Graph was populated with nodes and activation edges of knowledge in English, Chinese and primitive actions at first, where slight increase in processing time for “go forward” could be seen as new activation edges related to “go” and “forward” were added, such as “go shopping” and “move forward”. Later, to simulate the growth of knowledge

with most irrelevant to “go forward”, the Graph was populated with nodes and edges not connected to “go forward”, where the processing time of the same input remained relatively flat. This experiment demonstrates that the design of the system scales with respect to the size of the Graph, with a fair assumption that:

1. Most new knowledge is not related to a specific input represented by a subgraph (i.e. knowledge representations in biology and math not related to “go forward”).
2. A piece of new knowledge is likely not related to most existing nodes and edges in the Graph (i.e. “A banana is a fruit” not related to “go forward” or knowledge representations in music or math).

## REFERENCES

- [1] A. M. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. October, pp. 433–460, 1950.
- [2] J. Piaget, *The origins of intelligence in children*. London: Routledge & Kegan Paul, 1936.
- [3] J. Piaget, *Play, dreams and imitation in childhood*. London: Heinemann, 1945.
- [4] J. Piaget, *Introduction à l’epistémologie génétique*. Paris: Presses Universitaires de France, 1950, vol. 1.
- [5] J. Piaget, *The grasp of consciousness: Action and concept in the young child*. (Trans by S. Wedgwood). Harvard University Press, 1976.
- [6] J. M. Mandler, “How to build a baby: On the development of an accessible representational system,” *Cognitive Development*, vol. 3, no. 2, pp. 113–136, 1988.
- [7] J. M. Mandler, “The foundations of conceptual thought in infancy,” *Cognitive Development*, vol. 7, no. 3, pp. 273–285, 1992.
- [8] J. M. Mandler and L. McDonough, “Concept formation in infancy,” *Cognitive Development*, vol. 8, no. 3, pp. 291–318, 1993.
- [9] M. Minsky, “A framework for representing knowledge,” *MIT-AI Laboratory Memo 306*, 1974.
- [10] T. Mikolov, A. Joulin, and M. Baroni, “A roadmap towards machine intelligence,” in *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, 2016, pp. 29–61.
- [11] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, “Building machines that learn and think like people,” *Behavioral and Brain Sciences*, vol. 40, 2017.
- [12] G. Lupyan and B. Bergen, “How language programs the mind,” *Topics in cognitive science*, vol. 8, no. 2, pp. 408–424, 2016.
- [13] J. S. Bruner, “The growth and structure of skill,” *Mechanisms of Motor Skill Development*, pp. 63–93, 1970.
- [14] K. W. Fischer, “A theory of cognitive development: The control and construction of hierarchies of skills.,” *Psychological Review*, vol. 87, no. 6, p. 477, 1980.



- [15] K. J. Connolly, “Skill development: Problems and plans,” *Mechanisms of Motor Skill Development*, pp. 3–21, 1970.
- [16] A. Newell, “You can’t play 20 questions with nature and win: Projective comments on the papers of this symposium,” *Visual Information Processing*, pp. 283–310, 1973.
- [17] J. R. Anderson, *How can the human mind occur in the physical universe?* Oxford University Press, 2007.
- [18] A. Newell, *Unified Theories of Cognition*. USA: Harvard University Press, 1990, ISBN: 0674920996.
- [19] J. E. Laird, *The Soar Cognitive Architecture*. USA: The MIT Press, 2012, ISBN: 0262122960.
- [20] P. S. Rosenbloom, A. Demski, and V. Ustun, “The Sigma Cognitive Architecture and System: Towards Functionally Elegant Grand Unification,” *Journal of Artificial General Intelligence*, Jul. 2016.
- [21] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, “A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics,” *AI Magazine*, vol. 38, no. 4, pp. 13–26, Dec. 2017.
- [22] P. Langley, J. E. Laird, and S. Rogers, “Cognitive architectures: Research issues and challenges,” *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, 2009.
- [23] G. A. Miller, “Wordnet: A lexical database for english,” *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [24] R. Mihalcea, T. Chklovski, and A. Kilgarriff, “The SENSEVAL-3 english lexical sample task,” in *Proceedings of SENSEVAL-3, the third international workshop on the evaluation of systems for the semantic analysis of text*, 2004, pp. 25–28.
- [25] K. K. Schuler, *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania, 2005.
- [26] M. Palmer, D. Gildea, and P. Kingsbury, “The proposition bank: An annotated corpus of semantic roles,” *Comput. Linguist.*, vol. 31, no. 1, pp. 71–106, Mar. 2005.
- [27] C. J. Fillmore and C. Baker, “A frames approach to semantic analysis,” *The Oxford Handbook of Linguistic Analysis*, 2009.

- [28] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *Proceedings of International Conference on Learning Representations*, 2013.
- [29] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation.,” in *EMNLP*, vol. 14, 2014, pp. 1532–1543.
- [30] M. E. Peters *et al.*, “Deep contextualized word representations,” in *Proceedings of NAACL-HLT*, 2018, pp. 2227–2237.
- [31] J. Camacho-Collados and M. T. Pilehvar, “From word to sense embeddings: A survey on vector representations of meaning,” *Journal of Artificial Intelligence Research*, vol. 63, pp. 743–788, 2018.
- [32] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 302–308.
- [33] J. E. Sammet, “The use of english as a programming language,” *Communications of the ACM*, vol. 9, no. 3, pp. 228–230, 1966.
- [34] E. W. Dijkstra, “On the foolishness of ”natural language programming”,” in *Program construction*, Springer, 1979, pp. 51–53.
- [35] B. W. Ballard and A. W. Biermann, “Programming in natural language: “NLC” as a prototype,” in *Proceedings of the 1979 annual conference*, 1979, pp. 228–237.
- [36] R. E. Maas and P. Suppes, “Natural-language interface for an instructable robot,” *International Journal of Man-Machine Studies*, vol. 22, no. 2, pp. 215–240, 1985.
- [37] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, “Mobile robot programming using natural language,” *Robotics and Autonomous Systems*, vol. 38, no. 3-4, pp. 171–181, 2002.
- [38] T. Schouwenaars, M. Valenti, E. Feron, J. How, and E. Roche, “Linear programming and language processing for human-unmanned aerial-vehicle team missions,” *Journal of guidance, control, and dynamics*, vol. 29, no. 2, pp. 303–313, 2006.
- [39] M. Ralph and M. A. Moussa, “Toward a natural language interface for transferring grasping skills to robots,” *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 468–475, 2008.

- [40] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, “Learning to parse natural language commands to a robot control system,” in *Experimental robotics*, Springer, 2013, pp. 403–415.
- [41] T. Kollar *et al.*, “The Alexa Meaning Representation language.,” in *NAACL-HLT (3)*, 2018, pp. 177–184.
- [42] S. Gulwani and M. Marron, “Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation,” in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 803–814.
- [43] C. Quirk, R. Mooney, and M. Galley, “Language to code: Learning semantic parsers for if-this-then-that recipes,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 878–888.
- [44] M. Rabinovich, M. Stern, and D. Klein, “Abstract syntax networks for code generation and semantic parsing,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1139–1149.
- [45] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone, “Learning to interpret natural language commands through human-robot dialog,” in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [46] M. Eppe, S. Trott, and J. Feldman, “Exploiting deep semantics and compositionality of natural language for human-robot-interaction,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 731–738.
- [47] L. She and J. Chai, “Interactive learning of grounded verb semantics towards human-robot communication,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1634–1644.
- [48] J. Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. Xu, “Language to action: Towards interactive task learning with physical agents.,” in *IJCAI*, 2018, pp. 2–9.
- [49] R. L. Campbell and M. H. Bickhard, “Knowing levels and developmental stages.,” *Contributions to Human Development*, 1986.
- [50] R. Descartes, *Discourse on the method of rightly conducting the reason, and seeking truth in the sciences*. Sutherland and Knox, 1850.
- [51] B. de Spinoza, *The Collected Works of Spinoza, Volume 1*. Princeton University Press, 1985, vol. 1.

- [52] M. Cook, “Universality in elementary cellular automata,” *Complex Systems*, vol. 15, no. 1, pp. 1–40, 2004.
- [53] J. Von Neumann and A. W. Burks, “Theory of self-reproducing automata,” *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3–14, 1966.
- [54] P. S. Laplace, *Essai philosophique sur les probabilités*. Paris: Bachelier, 1840.
- [55] L. Brillouin, “Inevitable experimental errors, determinism, and information theory,” *Information and Control*, vol. 2, no. 1, pp. 45–63, 1959.
- [56] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, vol. 113, pp. 54–71, 2019.
- [57] N. Chomsky and M. P. Schützenberger, “The algebraic theory of context-free languages,” in *Studies in Logic and the Foundations of Mathematics*, vol. 26, Elsevier, 1959, pp. 118–161.
- [58] J. Nivre, “An efficient algorithm for projective dependency parsing,” in *Proceedings of the eighth international conference on parsing technologies*, 2003, pp. 149–160.
- [59] C. Haskins, K. Forsberg, M. Krueger, D. Walden, and D. Hamelin, “Systems engineering handbook,” in *INCOSE*, vol. 9, 2006, pp. 13–16.
- [60] K. Forsberg and H. Mooz, “The relationship of system engineering to the project cycle,” in *INCOSE international symposium*, Wiley Online Library, vol. 1, 1991, pp. 57–65.
- [61] B. Cameron, E. Crawley, and D. Selva, *Systems Architecture. Strategy and product development for complex systems*. Pearson Education, 2016.
- [62] J. S. Dahmann and K. J. Baldwin, “Understanding the current state of US defense systems of systems and the implications for systems engineering,” in *2008 2nd Annual IEEE Systems Conference*, IEEE, 2008, pp. 1–7.
- [63] R. Shishko and R. Aster, “NASA systems engineering handbook,” *NASA Special Publication*, vol. 6105, 1995.
- [64] W. Elm *et al.*, “Integrating cognitive systems engineering throughout the systems engineering process,” *Journal of Cognitive Engineering and Decision Making*, vol. 2, pp. 249–273, Dec. 2008.

- [65] H. Zhu and E. Feron, “A systems engineering approach to the design and education of a robotic baby,” in *INCOSE International Symposium*, Wiley Online Library, vol. 31, 2021, pp. 327–342.
- [66] C. Anderson *et al.*, *Lewis spacecraft mission failure investigation board final report*, 1998.
- [67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.
- [68] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, “Graph convolutional networks: A comprehensive review,” *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.
- [69] A. B. Arrieta *et al.*, “Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible ai,” *Information Fusion*, vol. 58, pp. 82–115, 2020.
- [70] J. Earley, “An efficient context-free parsing algorithm,” *Commun. ACM*, vol. 13, no. 2, pp. 94–102, 1970.
- [71] P. Norvig, “Techniques for automatic memorization with applications to context-free parsing,” *Computational Linguistics*, vol. 17, no. 1, pp. 91–98, 1991.
- [72] J. L. Lanciego, N. Luquin, and J. A. Obeso, “Functional neuroanatomy of the basal ganglia,” *Cold Spring Harbor Perspectives in Medicine*, vol. 2, no. 12, 2012.
- [73] A. Gopnik and A. N. Meltzoff, *Words, thoughts, and theories*. Mit Press, 1997.
- [74] D. K. Oller, L. A. Wieman, W. J. Doyle, and C. Ross, “Infant babbling and speech,” *Journal of Child Language*, vol. 3, no. 1, pp. 1–11, 1976.
- [75] A. N. Meltzoff, “Infant imitation and memory: Nine-month-olds in immediate and deferred tests,” *Child development*, vol. 59, no. 1, p. 217, 1988.
- [76] J. A. Fodor, *The language of thought*. Harvard university press, 1975, vol. 5.
- [77] L. Bloom, *The transition from infancy to language: Acquiring the power of expression*. Cambridge University Press, 1993.
- [78] J. Algeo, “Where do all the new words come from?” *American Speech*, vol. 55, no. 4, pp. 264–277, 1980.

- [79] E. Sagi, S. Kaufmann, and B. Clark, “Semantic density analysis: Comparing word meaning across time and phonetic space,” in *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, 2009, pp. 104–111.
- [80] B. Santorini, “Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision),” *Technical Reports (CIS)*, p. 570, 1990.
- [81] J. Nivre *et al.*, “Universal dependencies V1: A multilingual treebank collection,” in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, 2016, pp. 1659–1666.
- [82] M. H. Christiansen and N. Chater, “Toward a connectionist model of recursion in human linguistic performance,” *Cognitive Science*, vol. 23, no. 2, pp. 157–205, 1999.
- [83] D. A. Borgmann, *Beyond Language: Adventures in Word and Thought*. New York: Charles Scribner’s Sons, 1967, ISBN: 0674920996.
- [84] F. Ferreira and J. M. Henderson, “Recovery from misanalyses of garden-path sentences,” *Journal of Memory and Language*, vol. 30, no. 6, pp. 725–745, 1991.
- [85] C. T. Schütze, “PP attachment and argumenthood,” *MIT Working Papers in Linguistics*, vol. 26, no. 95, p. 151, 1995.
- [86] P. Gardner-Chloros, *Code-switching*. Cambridge: Cambridge university press, 2009.
- [87] D. Sankoff and S. Poplack, “A formal grammar for code-switching,” *Research on Language & Social Interaction*, vol. 14, no. 1, pp. 3–45, 1981.
- [88] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, “The vocabulary problem in human-system communication,” *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, 1987.
- [89] P. Goyal, S. Niekum, and R. J. Mooney, “Using natural language for reward shaping in reinforcement learning,” in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 2385–2391.
- [90] S. Wilson *et al.*, “The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems,” *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.
- [91] J. M. Mandler and L. McDonough, “Long-term recall of event sequences in infancy,” *Journal of Experimental Child Psychology*, vol. 59, no. 3, pp. 457–474, 1995.
- [92] A. Kleinman and K. Fink. “Artificial intelligence at home in pittsburgh,” NPR. (Sep. 2018). <https://www.npr.org/programs/1a2018/09/20/649967527>.

- [93] E. Davis, *Representations of commonsense knowledge*. San Mateo, California: Morgan Kaufmann, 1990.
- [94] I. Leite, C. Martinho, and A. Paiva, “Social robots for long-term interaction: A survey,” *International Journal of Social Robotics*, vol. 5, no. 2, pp. 291–308, 2013.
- [95] S. Thrun and T. M. Mitchell, “Lifelong robot learning,” *Robotics and Autonomous Systems*, vol. 15, no. 1-2, pp. 25–46, 1995.
- [96] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.
- [97] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, “An empirical investigation of catastrophic forgetting in gradient-based neural networks,” *arXiv preprint arXiv:1312.6211*, 2013.