Architecture cognitive constructiviste : un modèle pour concevoir un agent automotivé capable de faire du sens et de construire des connaissances de l'environnement

Jianyong Xue

N°d'ordre NNT : 2020LYSE1242

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Université Claude Bernard Lyon 1

Ecole Doctorale N° ED 512
Informatique et Mathématiques de Lyon (InfoMaths)

Spécialité de doctorat : Architecture Cognitive Constructiviste
Discipline : Informatique

Soutenue publiquement le 23/11/2020, par :
Jianyong XUE

Constructivist Cognitive Architecture: A model for Designing Self-motivated Agent Capable of Sense-making and Knowledge Construction of the Environment

---

Devant le jury composé de :

M. ALEXANDRE Frédéric, Professeur, INRIA Bordeaux — Rapporteur
M. DE LOOR Pierre, Professeur, Ecole Nationale D'ingénieurs de Brest — Rapporteur
Mme GHODOUS-SHARIAT TORBAGHAN Parisa, Professeure, Université Lyon 1 — Examinatrice
M. MARSHALL James, Professeur, Sarah Lawrence College — Examinateur
Mme MEEDEN Lisa, Professeure, Swarthmore College — Examinatrice

Mme HASSAS Salima, Professeure, Université Lyon 1 — Directrice de thèse
M. GEORGEON Olivier, Professeur Associé, Université Catholique de Lyon — Co-directeur de thèse

Université Claude Bernard – LYON 1

Administrateur provisoire de l'Université	M. Frédéric FLEURY
Président du Conseil Académique	M. Hamda BEN HADID
Vice-Président du Conseil d'Administration	M. Didier REVEL
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. Philippe CHEVALLIER
Vice-Président de la Commission de Recherche	M. Jean-François MORNEX
Directeur Général des Services	M. Pierre ROLLAND

COMPOSANTES SANTE

Département de Formation et Centre de Recherche en Biologie Humaine	Directrice : Mme Anne-Marie SCHOTT
Faculté d'Odontologie	Doyenne : Mme Dominique SEUX
Faculté de Médecine et Maïeutique Lyon Sud - Charles Mérieux	Doyenne : Mme Carole BURILLON
Faculté de Médecine Lyon-Est	Doyen : M. Gilles RODE
Institut des Sciences et Techniques de la Réadaptation (ISTR)	Directeur : M. Xavier PERROT
Institut des Sciences Pharmaceutiques et Biologiques (ISBP)	Directrice : Mme Christine VINCIGUERRA

COMPOSANTES & DEPARTEMENTS DE SCIENCES & TECHNOLOGIE

Département Génie Electrique et des Procédés (GEP)	Directrice : Mme Rosaria FERRIGNO
Département Informatique	Directeur : M. Behzad SHARIAT
Département Mécanique	Directeur M. Marc BUFFAT
Ecole Supérieure de Chimie, Physique, Electronique (CPE Lyon)	Directeur : Gérard PIGNAULT
Institut de Science Financière et d'Assurances (ISFA)	Directeur : M. Nicolas LEBOISNE
Institut National du Professorat et de l'Education	Administrateur Provisoire : M. Pierre CHAREYRON
Institut Universitaire de Technologie de Lyon 1	Directeur : M. Christophe VITON
Observatoire de Lyon	Directrice : Mme Isabelle DANIEL
Polytechnique Lyon	Directeur : Emmanuel PERRIN
UFR Biosciences	Administratrice provisoire : Mme Kathrin GIESELER
UFR des Sciences et Techniques des Activités Physiques et Sportives (STAPS)	Directeur : M. Yannick VANPOULLE
UFR Faculté des Sciences	Directeur : M. Bruno ANDRIOLETTI

Abstract

Infants are excellent at interacting with the environment. Especially in the initial phase of cognitive development, they exhibit amazing abilities to generate novel behaviors in unfamiliar situations, and explore actively to learn the best while lacking extrinsic rewards from the environment. These abilities of sense-making and knowledge construction of the environment set them apart from even the most advanced autonomous robots. However, for most artificial agents (and robots), acquiring such abilities is overwhelming.

While for most traditional Artificial Intelligence (AI) approaches, learning is usually insufficient, such as with various biases, and lacks of flexibility. Seeking ways to explain the learning mechanism behind infants' early cognitive development and try to replicate some of these abilities for an autonomous agent has been an active point in recent efforts of robotics and AI research.

Over the last decades, a multitude of theories and methods have been devoted to the study of learning mechanisms in infants' early-stage cognitive development and proposed various algorithms targeted at designing and implementing a self-motivated agent, like learning with constructivist paradigm, reinforcement learning paradigm, active learning, developmental learning, intrinsic motivation, curiosity-driven learning, enactive paradigm and attention mechanism.

Specifically, among these theories and approaches, the *constructivism* as a knowledge acquisition theory that describes the information processing mechanisms behind infants' cognitive development. As infants interact with the world around them, they continually absorb new knowledge build upon existing knowledge, and simultaneously adapt previous ideas to accommodate with new information. In this dissertation, I present a computational model of Constructivist Cognitive Architecture (CCA) as a way towards simulating the early learning mechanism of infants' cognitive development based on theories of enactive cognition, intrinsic motivation, and constructivist epistemology. Meanwhile, the CCA allows a self-motivated agent to autonomously construct the perception of the environment and acquire capabilities of self-adaption and flexibility to generate proper behaviors to tackle with diverse situations in interacting with the environment.

Different with traditional cognitive architectures, the introduced model neither initially endows the agent with prior knowledge of its environment, nor supplies it with knowledge during its learning process. Accordingly, I am not proposing an algorithm that optimizes exploration of a predefined problem-space to reach predefined goal states. Instead, I propose a way for the agent to autonomously encode the interaction experiences and reuse behavioral patterns based on the agent's self-motivation implemented as inborn proclivities that drive the agent in a proactive way. In addition, I introduce two forms of self-motivation: successfully enacting sequences of interactions (or called *autotelic motivation*), and preferably enacting interactions that have predefined positive values (or called *interactional motivation*). Following these drives, the agent autonomously learns regularities afforded by the environment, and constructs hierarchical sequences to perform higher-level behaviors.

Furthermore, I proposed a Bottom-up hiErarchical sequential Learning model based on the CCA, which is also called BEL-CA, as a solution for an autonomous agent learning hierarchical sequences of behaviors and acquiring capabilities of self-adaptation and flexibility. The agent represents its current situation in terms of perceived affordances that develop through the agent's experience. This situational representation works as an emerging situation awareness that is grounded in the agent's interaction with its environment and that in turn generates expectations and activates adapted behaviors. Through its activity and these aspects of behavior (behavioral proclivity, situation awareness, and hierarchical sequential learning), the agent starts to exhibit emergent sensibility, intrinsic motivation, and autonomous learning.

Moreover, I introduced an implementation of a toolkit to analyze the learning process at run time, which is called GAIT (Generating and Analyzing Interaction Traces Toolkit). I use GAIT to report and explain the detailed learning process and the structured behaviors that the agent has learned on each decision making step. The experiment demonstrated that the agent learned to successfully interact with its environment and to avoid unfavorable interactions using regularities discovered through interaction.

Following with dissertation, this initial autonomous mechanism provides a basis for implementing autonomously developing cognitive systems. Therefore, the agent gets the perception of this world and generates proper behaviors in different and complicated situations. Thus, the agent could moving around freely and learn regularities of the environment. Meanwhile, it spurs the agent to discover a long sequence of "correct" actions to find an accurate configuration of the environment and reuse it appropriately.

Résumé

Les nourrissons sont excellents pour interagir avec l'environnement. Surtout dans la phase initiale du développement cognitif, ils présentent des capacités étonnantes à générer de nouveaux comportements dans des situations inconnues et à explorer activement pour apprendre le meilleur tout en manquant de récompenses extrinsèques de l'environnement. Ces capacités de création de sens et de construction de connaissances de l'environnement les distinguent même des robots autonomes les plus avancés.

Pour la plupart des agents artificiels (et des robots), l'acquisition de telles capacités est écrasante. Dans la plupart des approches traditionnelles d'intelligence artificielle (IA), l'apprentissage est généralement insuffisant, avec divers biais et manque de flexibilité. Chercher des moyens d'expliquer le mécanisme d'apprentissage derrière le développement cognitif précoce des nourrissons et essayer de reproduire certaines de ces capacités que les bébés ont pour un agent autonome sont devenus un point focal des efforts récents en robotique et en recherche sur l'IA.

Au cours des dernières décennies, une multitude de théories et de méthodes ont été consacrées à l'étude des mécanismes d'apprentissage dans le développement cognitif précoce des nourrissons et au développement de divers algorithmes visant à concevoir et à mettre en œuvre un agent auto-motivé, sous diverses approches telles que l'apprentissage utilisant paradigme constructiviste, paradigme d'apprentissage par renforcement, apprentissage actif, apprentissage développemental, motivation intrinsèque, apprentissage axé sur la curiosité, paradigme énactif et mécanisme d'attention.

Parmi ces théories et approches, le *constructivisme* en tant que théorie d'acquisition des connaissances qui décrit les mécanismes de traitement de l'information derrière le développement cognitif des nourrissons. Au fur et à mesure que les nourrissons interagissent avec le monde qui les entoure, ils absorbent continuellement de nouvelles connaissances, s'appuient sur les connaissances existantes et adaptent des idées antérieures pour accueillir de nouvelles informations. Le paradigme d'apprentissage constructiviste suggère qu'un agent autonome construit itérativement la représentation de son environnement à travers ses expériences d'interactions et sans a priori.

Dans cette thèse, je propose un modèle informatique de l'architecture cognitive constructiviste (CCA) comme moyen de simuler le mécanisme d'apprentissage précoce du développement cognitif des nourrissons basé sur les théories de la cognition énactive, de la motivation intrinsèque et de l'épistémologie constructiviste. Pendant ce temps, le CCA permet à un agent motivé de construire de manière autonome la perception de l'environnement et d'acquérir des capacités d'auto-adaptation et de flexibilité pour générer des comportements appropriés pour faire face à diverses situations en interagissant avec l'environnement.

Contrairement aux architectures cognitives traditionnelles, le modèle introduit ne confère pas initialement à l'agent une connaissance préalable de son environnement, ni ne lui fournit des connaissances au cours de son processus d'apprentissage. En conséquence, je ne propose pas d'algorithme qui optimise l'exploration d'un espace-problème prédéfini

pour atteindre des états d'objectifs prédéfinis. Au lieu de cela, je propose un moyen pour l'agent d'encoder de manière autonome les expériences d'interaction et de réutiliser des modèles de comportement basés sur l'auto-motivation de l'agent implémentée comme des penchants innés qui conduisent l'agent de manière proactive. De plus, je présente deux formes d'auto-motivation: la mise en œuvre réussie de séquences d'interactions (ou appelées *motivation autotélique*), et de préférence des interactions qui ont des valeurs positives prédéfinies (ou appelées *motivation interactionnelle*). pulsions, l'agent apprend de manière autonome les régularités offertes par l'environnement, et construit la perception causale de phénomènes dont la présence hypothétique dans l'environnement explique ces régularités.

De plus, je propose un modèle d'apprentissage séquentiel ascendant hiErarchique basé sur le CCA, également appelé BEL-CA, comme solution pour un agent autonome apprenant des séquences hiérarchiques de comportements et acquérant des capacités d'auto-adaptation et de flexibilité. L'agent représente sa situation actuelle en termes d'affordances perçues qui se développent à travers l'expérience de l'agent. Cette représentation situationnelle fonctionne comme une prise de conscience de situation émergente qui est ancrée dans l'interaction de l'agent avec son environnement et qui à son tour génère des attentes et active des comportements adaptés. Par son activité et ces aspects du comportement (propension comportementale, conscience de situation et apprentissage séquentiel hiérarchique), l'agent commence à faire preuve d'une sensibilité émergente, d'une motivation intrinsèque et d'un apprentissage autonome.

De plus, j'introduis une implémentation d'une boîte à outils pour analyser le processus d'apprentissage au moment de l'exécution, qui s'appelle GAIT (Generating and Analyzing Interaction Traces Toolkit). J'utilise GAIT pour rendre compte et expliquer le processus d'apprentissage détaillé et les comportements structurés que l'agent a appris à chaque étape de prise de décision. Je rapporte une expérience dans laquelle l'agent a appris à interagir avec succès avec son environnement et à éviter les interactions défavorables en utilisant des régularités découvertes par interaction.

Suivant les théories du développement cognitif, je soutiens que ce mécanisme autonome initial fournit une base pour la mise en œuvre de systèmes cognitifs en développement autonome. Par conséquent, l'agent obtient la perception de ce monde et génère des comportements appropriés dans des situations différentes et compliquées. Ainsi, l'agent pourrait se déplacer librement et apprendre les régularités de l'environnement. Pendant ce temps, cela incite l'agent à découvrir une longue séquence d'actions "correctes" pour trouver une configuration précise de l'environnement et la réutiliser de manière appropriée.

Acknowledgements

The past three years of studying in Lyon are the most memorable and the most rewarding time for me. The unique study experience broadened my horizons, enriched my knowledge, and experienced a different lifestyle, which gave me new discoveries and thoughts. In the past three years, I have met knowledgeable scholars, made interesting friends, participated in various academic reports and lectures, and attended a variety of activities. At the same time, I have witnessed how fast the new ideas and theories in Artificial Intelligence are proposed and applications have been developed, and also I'm so excited to be a part in this. I would not have been able to make this journey without the help and encourage from so many nice people and I think I have not done enough.

First and foremost, I would like to express my sincere gratitude to my supervisors Salima Hassas and Olivier Georgeon. They both have a very insightful and high-level view about the field while also have uncommonly detail oriented and understands the nature of the problems very well. Each time I take part in a discussion with them, always gives me a lot of inspirations. More importantly, Salima and Olivier are extremely kind, caring and supportive advisors that I could not have asked for more. During three years of my thesis, they gave me enough patience and trust to allow me to keep exploring and trying. At the same time, they also gave me more encouragement and let me persevere all the way.

I would like to thank Alain Mille, a mentor with profound knowledge and rigorous scholarship, whom I admire the most and want to learn from. As a well-known scholar in the field of research, he has given me continuous attention and encouragement.

I want to thank warmly all members of the Systèmes Multi-Agents (SMA) team in the LIRIS laboratory. Véronique Deslandres, Laetitia Matignon, Frédéric Armetta, Mathieu Lefort for their precious advises and support during my thesis. I also thank all other members for welcoming me into this research team: Arthur Aubret, Rémy Chaput, Simon Forest, Alexandre Galdeano, Huan Vu, Benoit Vuillemin for their valuable feedback on my works. In particular, Alexandre as one of my best friends always gives me so many good suggestions and invited me to participate in various interesting activities. Every time I talk with him on the cutting-edge theories in the academic world and novel technologies in the industrial fiedlds, as well as hobbies like mechanism of sport cars, I always feel relaxed and happy, and benefit a lot. Simon and I share a same office in the lab and he helped me a lot in administrative procedures, especially the translations of various french documents.

Collaboration is a big lesson that I have learned, and also a fun part in my thesis. Meetings and presentations with the company of Hoomano gave me an opportunity to get acquainted with experienced scholars and talented engineers in the industrial fields. I'd like to thank Xavier Basset, Amélie Cordier, Pierre Laurent and Laurianne Charrier for they generously sharing their valuable experiences in applications and new challenges in applying these applications with real robots. Their constructive suggestions helped me a lot when I was designing an algorithm for an autonomous agent and considering

to improve its performance in diverse scenarios. Moreover, the collaboration with the team from Université Catholique de Lyon (UCLy) on the project of INIT give me another opportunity to explain the theory of constructivism to young developers who are interested in studying of cognitive development for autonomous robots. Particularly, I'd like to thank Paul Robertson and Heidi Souibki. Paul gave me many constructive advice on the design of learning algorithms and approaches to improve the agent's performance. Heidi inspired me to think about the design of a cross-platform algorithm architecture and find ways to implement it on the ROS and Gazebo platform.

I would like to thank Mathieu Gillermin and Béatrice de Montera from the Université Catholique de Lyon (UCLy). Thank Mathieu's expertise in philosophy and cognition for providing me with valuable suggestions, which gave me a deeper understanding of the construction of causal models. Thank Béatrice for her caring and encouraging, which gave me enough confidence.

I thank my parents for their constant confidence and support during my studies and to push me to do what I like. I address a special thank to my sister, for her immeasurable thoughtfulness and support. I thank all my precious friends for their kindness and to make me explain my works regularly. It would have never be possible without all of them around me.

Finally, I thank all members of the University Claude Bernard Lyon 1 and people who works in the library of BU Sciences La Doua for providing such a great and comfortable workplace.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Knowledge of the world, [...], is created from the interaction with the environment, rather than existing in an ontic reality, supposedly pre-existing or available for registration from the physical world."

Roesch et al. ,2013, § 1, p1.

Contents

1.1 Context

During the initial phase of cognitive development, infants exhibit amazing abilities to generate novel behaviors with unfamiliar situations and explore actively to learn the best with lacking extrinsic rewards from the environment [1, 2, 3]. With skills and abilities that borne with (such as looking, listening, sucking, touching, and grasping) to interact with the environment, infants continually acquire new information build upon their existing knowledge, and simultaneously adapt previous ideas to accommodate with this new information over a short period of time and involves a dramatic of growth.

Infants, as described of "scientists in the crib" by *Gopnik et all. 1999*[4] who intentionally discover events that are new, informative, then exciting to them [5, 6]. Their abilities of sense-making and knowledge construction of the environments set them apart from even the most advanced autonomous robots.

For most artificial agents (and robots), acquiring such learning abilities that infants have is overwhelming. Learning is usually insufficient [7], with different biases [8, 9, 10], and lacks of flexibility [11, 12]. One of the biggest reasons is that traditional learning approaches heavily rely on the problem specificities and the availability of prior knowledge that is specific to a task proposed by the system designer. Thus how well the agent perceives the state of the environment (for example, through actuators/sensors patterns) determines the performance of the learning models [13]. As Russell & Norvig [14] state that "the problem of AI is to build agents that receive percepts from the environment and perform actions". The approaches based on this statement assumes that the input data for the agent is a direct function of the state of the environment. However, this assumption is not satisfied in situations where the input data of the agent comes from the feedback of actions in the control loop.

In addition, the agent is designed for desired goals and with a well-defined reward function (such as reinforcement learning paradigm), it's thus motivated to maximize the accumulated reward when it achieves these specific goals, rather than focuses on learning behavioral patterns and constructing the knowledge of the environment from interaction experiences. Particularly in cases where the agent is designed to pursue a wide variety of tasks, the reward function needs to be designed sufficiently to include all possible interaction situations. Moreover, it could lead to an unexpected result in situations where the desired reward function is modified [11]. Agent's abilities of self-adaptation and flexibility to tackle with diverse situations in different (or dynamic) and complex environment are limited as well [12] .

Seeking ways to understand the learning mechanisms behind infants' early-stage cognitive development and try to replicate some of these abilities that babies have for an autonomous agent, which allows it to behave in an "intelligent" and flexible manner has become an active research domain as presented in [15, 16, 2, 17, 18, 19, 10, 20, 21, 22]. Over the last decades, a multitude of theories and methods have been devoted to the study of learning in infants' cognitive development and developing various algorithms targeted at designing and implementation a self-developing agent, under various approaches such as learning using constructivist paradigm [23, 24, 25, 13, 12, 26], reinforcement learning paradigm [27, 28, 7, 29, 19, 30, 11], active learning [10, 31, 32, 33], developmental learning [30, 16, 20, 34, 35, 36], intrinsic motivation [19, 18, 31, 17, 37], curiosity-driven learning [38, 16, 39, 20, 40, 41], enactive paradigm [42, 43, 44, 9, 45] and attention mechanism [46, 47, 48, 49].

Among these theories and approaches, the *constructivism* as a knowledge acquisition theory ameliorated by Piaget [50] that describes the information processing mechanisms

behind infants' cognitive development. The constructivist learning paradigm suggests that an autonomous agent iteratively construct the representation of its environment and its self through its experiences of interactions and without a prior knowledge. Specifically, the agent is not designed as a passive observer of reality, but represents its current situation with perceived affordances that developed through the agent's experience. Also as as Georgeon and Ritter [9] mentioned that "this situational representation works as an emerging situation awareness that is grounded in the agent's interaction with its environment and that in turn generates expectations and activates adapted behaviors".

Inspired from the theory of constructivism and recent developments in AI [18, 51, 7, 29, 52], we introduce a new computational model of cognitive architecture in this dissertation, which allows an autonomous agent to acquire the perception of the environment from its interaction experience and obtain capabilities of self-adaptation and flexibility for generating proper behaviors to tackle with diverse situations, as a way to simulate the early mechanism of infants' learning process based on theories of constructivist epistemology, intrinsic motivation and enactive cognition.

Particualrly, the introduced computational model neither initially endows the agent with the prior knowledge of its environment, nor supplies it with knowledge in its learning process. Instead, it propose a way for the agent to autonomously encode the interactional experiences and reuse behavioral patterns based on the its self-motivation (we prefer to call it the *interactional motivation* [53]) as inborn proclivities that drive the agent in a proactive way. Following these drives, the agent autonomously learns regularities afforded by the environment, and hierarchical sequences of behaviors adapted to these regularities [9].

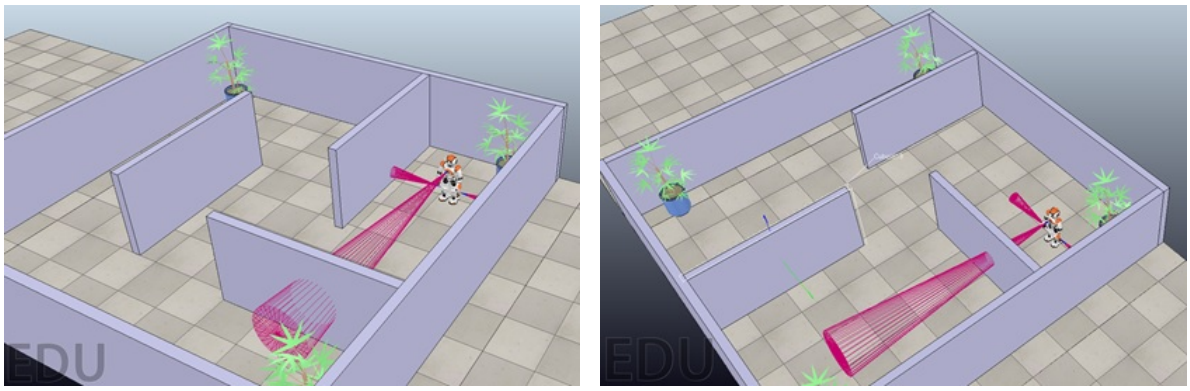
1.2 The challenge for an autonomous agent

1.2.1 Problem statement

Imaging the following scenario: an agent is placed in an unfamiliar environment without any prior knowledge(as shown in Figure 1.1(a)), and it endowed with few innate actions that enable it to perform elementary functions such as moving one step forward, turning its direction and sensing the environment. In the early stage of the agent interacting with the environment, the agent does not know the meaning of these actions and needs it to learn this from its interaction experience. Typically, the difference between this scenario with other traditional ones is in that the agent's input data doesn't directly come from the representation of the environment, nor the final goals for it to achieve. The questions raise as follows: (a) how the agent acquires the perception of the environment and (b) how it proposes appropriate behaviors for interacting. Moreover, (c) how does the agent behave flexibly in the cases where the environment has been changed (as shown in Figure 1.1(b)).

In such scenario, the agent needs to learn regularities of interaction that afforded by the environment and generate structured behaviors for diverse situations[1]. Meanwhile, following a viable learning paradigm that allows the agent to successfully interact with its environment and learn to avoid unfavorable interactions using structured behavior it has learned. In addition, with capabilities of self-adaptation and flexibility that prevent the agent learning from zero for the response of cases where the environment changes or performance degrading in interacting with the environment.

[1]The perception is internal constructed by the agent from its own interaction experience rather than input data directly comes from the state of the environment.



(a) Agent is placed in an unfamiliar environment. (b) The agent is in a changed environment.

Figure 1.1: The interaction scenarios.

1.2.2 Challenges

Designing such an autonomous agent to construct the knowledge of the environment and have flexibility to tackle with diverse interactive situations is one of the greatest and long-standing challenges in Artificial Intelligence. To be autonomous, the agent must learn to master the contingencies from its own sensorimotor experiences in the world [54] (refer to section 3.1.3).

For an autonomous agent, it needs to construct the perception of the environment by autonomously discovering, learning and exploiting regularities of interaction afforded by the environment, but without encoding any prior knowledge. As Roesch et al. [55] (§1, p1) stated that:"knowledge of the world [...] is created from the interaction with the environment, rather than existing in an ontic reality, supposedly pre-existing or available for registration from the physical world".

Furthermore, facing with different interaction scenarios, the agent should have capabilities of adaptation and flexibility for recognizing the context and generate appropriate behaviors. Designing a such learning model for an autonomous agent mainly involves the following challenges:

- The environment-agnosticism challenge. The environment-agnosticism was proposed that the agent should not implement ontological presuppositions about the environment. Instead, the agent should have abilities to learn to construct the perception of the environment from sensorimotor interactions without any predefined knowledge of the environment.

- The autonomous and active learning challenge. The cognitive development should be in an open-ended manner and the agent is self-motivated to learn behaviors that fulfill an innate preference [17], which devoting to forming the core of a system for task-independent learning.

- The progressive and incremental learning challenge. In the progress of cognitive development, the agent continuously interacts with the environment around it and undergo a developmental way to obtain new skills associated with its interaction experience, rather than manually crafted by hand.

- The learning of regularities of interaction challenge. The agent has capabilities to discover, learn and exploit regularities of interaction to master the sensorimotor

contingencies afforded by its coupling with the environment. Regularities of interaction are patterns of interaction that occur consistently and are building blocks of constructing structured behaviors for increasing complexity.

- The acquirement of capabilities of adaptation and the flexibility challenge. The agent has capabilities of continuously absorbing new knowledge from interaction experience and adapting its percept of the environment in the cases where it receives feedback from the environment beyond its expectations. Moreover, the agent is capable of recognizing the context and generating appropriate behaviors for various interaction situations.

1.3 Motivation

Based on the challenges we mentioned above, this dissertation mainly focuses on the following aspects:

Knowledge construction of the environment through agent's sensorimotor interactions. With the cognition development is active and incremental, the agent is self-motivated to discover, learn and explore regularities of interaction from its stream of experiences and to construct knowledge about phenomena, which hypothetical presence in the environment explains these regularities. The agent could construct categories of phenomena, and exploit this knowledge to satisfy its innate preferences, as the way that imitates the humans learning process from experiences.

Learning of structured behaviors with hierarchical sequential learning paradigm. The agent's cognitive development follows a hierarchical progression. With rudimentary lower-level patterns of regularities that have learned from interactions, the agent is capable of autonomously organizing them into a form of higher-level abstraction, which is *hierarchical sequential learning of structured behaviors*. With structured behaviors, agent could gradually learn and exploit it, and simultaneously infer the structure of the environment based on the patterns in the stream of interactions traces. Generating proper behaviors for different situations as well. [2]

Context recognition, adaptation and flexibly generating proper behaviors. The agent enables to recognize current context[3] and effectively proposes intentions with proper behaviors for the next interaction. In particular, the agent has abilities to accurately represent the context while reasonably matching it with its own interaction experience. In addition, an efficient decision-making mechanism is needed for the selection of an appropriate intention. Furthermore, with performance degrading in the enaction of this intention, the agent is capable of acquire new structured behaviors based on the modifications of previously learned behaviors. In this work, we prefer the definition of "context" from Abowd et al. [62] (p306) that "context is the information that can be used to

[2]The "higher-level" in this work indicates that the agent not only enables to learn to organize simple behavioral patterns, but also has a reasoning mechanism to exploit experiences of fallback into a more complex structure with flexibility in various scenarios. The agent could increasingly learn elaborated behaviors and organized them in a hierarchy that reflects how the agent exploits the higher-level regularities afforded by the environment.

[3]The definition of context from [56, 57]is that the context as location, identities of nearby people and objects, and changes to those objects, [58] enumerates context as the user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the user's environment. [59] included the entire environment by defining context to be aspects of the current situation. [60] defines context to be the user's physical, social, emotional or informational state. [61] defines context to be the subset of physical and conceptual states of interest to a particular entity. We prefer the definition from [62].

characterize the situation of an entity, which could be a person, place, or object that is considered relevant to the interaction". Specifically, the context focuses on the reciprocal process of interactions (some of them involve geographical conditions of the environment), which is used to discover behavioral patterns for representing the learning process of the agent.

Through this dissertation, we are going to answer the following questions:

- How can an agent build knowledge of the environment and of itself effectively and efficiently with innate actions?
- How to design an efficient cognitive architecture which fits with agent's continuous interaction with the environment and new learned behavioral patterns with progressive learning? Is there any efficient structure of behavioral patterns exist?
- With the learning is autonomous and progressive, which way can let us effectively organize the behavioral patterns the agent has learned into a form of abstraction to perform the structured behavior?
- Being placed in a new or a more complicated environment, how can the agent aware the changes of the environment and generate proper behavior in it?

1.4 Overview of the dissertation

Following the central themes that we just have discussed above, this dissertation consists of five parts: Introduction and background (Chapters 1, 2 and 3), The structure of CCA (Chapter 4), Applications of CCA (Chapters 5 and 6), Experimental settings and performance evaluation (Chapters 7) and Conclusion, discussion and perspective (Chapter 8).

In Chapter 2 reviews research in early mechanisms of infants' cognitive development and main ideas that are applied in designing an autonomous agent. The research of infants' learning mechanism includes Piaget's theory of cognitive development, the theory of Information Processing Principles (IPPs) and intrinsic motivations (like curiosity, novelty etc.) in infants' cognitive development. Also discussed the recently applications based on these theories in infants' early learning mechanisms.

Chapter 4 describes the design, the structure and the implementation of Constructivist Cognitive Architecture, a computational model to simulate the learning mechanisms behind infants' cognitive development, for designing an autonomous agent constructs the perception of the environment and acquires capabilities of self-adaption and flexibility. This is one of the main contributions of this dissertation.

In Chapter 5, we demonstrate CCA's ability to discover and learn regularities of interaction in its stream of experience and construct causal perception between phenomena whose hypothetical presence in the environment explains these regularities.

In Chapter 6, we introduce a Bottom-up hiErarchical sequential Learning model with CCA, which is also called BEL-CA, as a solution for an autonomous agent continuously learning representations of the environment and acquiring capabilities of self-adaptation and flexibility.

Chapter 7 sets up an experimental scenario and introduce an implementation of analyzing agent's interaction traces to demonstrate CCA's ability of bottom-up hierarchical sequential learning. The experimental scenario is designed based on the classic Small Loop

Problem (SLP), which acts as a benchmark of implementing and demonstrating cognitive emergence for an autonomous agent. Meanwhile, we verify the agent's capabilities of self-adaptation and flexibility by modifying the environment to simulate interaction scenarios that it hasn't experienced before.

Finally, Chapter 8 presents an overview of the work presented in this dissertation and concludes. Meanwhile, we provide several open issues related the design of CCA and its applications. With problems still remain in the CCA and challenges that we have not yet faced, we introduce the perspectives for the future work.

1.5 Contributions

The contributions of this dissertation are summarized as follows:

- We introduced a computational model of Constructivist Cognitive Architecture (CCA) as the way to simulate the early mechanisms of infants' cognitive development based on theories of enactive cognition, intrinsic motivation and constructivist epistemology. Furthermore, the proposed cognitive architecture allows a self-developing agent to autonomously construct the perception of the environment and obtain capabilities of self-adaption and flexibility to generate proper behaviors in tacking with diverse situations.

- We presented a learning model that endows an autonomous agent with two different motivations: (a) the motivation to be in control of one's activity by seeking to successfully enact interactions and (b) the motivation to enact interactions have positive predefined positive valences and to avoid enacting interactions have predefined negative valences. These two motivations spur the agent to learn regularities of interaction afforded by the environment.

- We demonstrated CCA's ability to discover and learn regularities of interaction in its stream of experience and construct causal perception between phenomena whose hypothetical presence in the environment explains these regularities. Moreover, we introduced a Bottom-up hiErarchical sequential Learning model with CCA, which is also called BEL-CA, as a solution for an autonomous agent continuously learning representations of the environment and acquiring capabilities of self-adaptation and flexibility.

- We proposed an implementation of toolkit to analyze the learning process at run time called GAIT (Generating and Analyzing Interaction Traces), which allows us to report and observe the detailed learning process for the agent interacts with environment and the structured behaviors it has learned in each decision-making.

- Finally, we introduced the design and implementation of new simulations of CCA and GAIT for autonomous robots on multiples platforms. We provided methods to precisely control the robots' movement and explained strategies for the robots to maintain alignments with the environment.

Chapter 2

The state of the art

Contents

In this chapter, we suggest a review of related works. We start with studies of early stage learning mechanisms in infants' cognitive development. Specifically, theories of constructivism, Information Processing Principles (IPPs) and intrinsic motivations (like curiosity, novelty and pleasantness etc.) in infants' cognitive development have been gradually developed and improved with considerable success. After then, we introduce the recently applications based on these theories for an autonomous agents and cognitive development with acquiring some abilities that infants have. Furthermore, we did a survey of developments in cognitive architectures and then introduced several successful and classic cognitive architectures, which include the Soar Cognitive Architecture, Constructivist Learning Architecture (CLA), and Enactive Cognitive Architecture (ECA). Finally, we made a summary of this chapter.

2.1 Infants' cognitive development

In the initial phase of cognitive development, in spite of the rapid physical growth, infants also exhibit significant development of abilities in knowledge acquisition, thinking and reasoning. With a number of ways (such as through sight, touch, taste, etc.) of interacting with the environment, infants dynamically acquire the information and construct the perception of the environment, engaging physically with objects in the environment and behaving in novel and surprising ways. The question is how do infants connect and make sense of what they are learning?

One natural idea is that the capacities of world modeling in infants are the result of built-in core systems, including those for object attention and permanence, self-localization, number sense, and intuitive physics [34]. Mentioned by Lipsitt [66] that "once operational, such systems would naturally give the infant a basis on which to make judgments about which sequences of actions would be interesting to perform". Expert system [63, 64, 65] is such a model that emulates the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field. Typically, it incorporates a knowledge about accumulated experience and an inference or rules engine for applying the knowledge base to each particular situation that is described to the program.

Meanwhile, it exists another understanding of that infants have a host of innate responses prepared to interact with the environment [66]. The acquisition of structured behaviors is considered as a model of specific sets of response and response-induced stimulation in which each elemental response may serve also as a stimulus for the next reaction component. Particularly, a radical view from the behaviorist (e.g. Watson [67]) described that "infants with a set of unconditional responses [. . .] environment then begins to shape into patterns of behavior" [66].

However, from nervous system considerations, a view held that young children cognitive development is based on the relationship between the maturation and the environment [66]. It suggests that the maturation-environment relationship is a two-way street that (a) experimental effects awaited maturational changes that would permit the experience to have an effect and for the behavior to occur. And (b) environmental enrichment through the implementation of special experiences can alter maturation rates in certain sphere, which in turn can alter the readiness of the organism to assimilate further stimulation in that modality. Evidences from Globus and Scheibe [68], Schapiro and Vukovieh [69] and Hubel and Wiesel [70] support the contention that experience itself increases dendrite proliferation.

As one of the a first psychologists who makes a systematic study of the beginnings of

mental development and the origins of intelligence in children, Piaget [50] explains the mechanisms and processes of how an infant, and then the child, develops into an individual who can reason and think using hypotheses, and constructs a mental representation of the world. Disagreed with the idea that intelligence was a fixed trait, he regarded cognitive development as a progressive reorganization of mental processes which occurs due to biological maturation and interaction experience with the environment [50, 71]. The mechanism by which infants integrate experience into progressively higher-level representations, which called the "constructivism".

The constructivism as a theory of knowledge acquisition proposes that "learning happens as a result of an internal mental representations and external perceptions from interactions" [72]. According to the constructivism, infants learning progress from simple to complex models of the environment which allow the them to build higher-level representations from lower-level ones. Behavior in constructivism acts as the adaptation to the environment is controlled through mental organizations of sensorimotor scheme, which the individual uses to represent the world and generate corresponding actions. Each *sensorimotor scheme* (refers to section 3.1.1) binds the correlation between the mental and physical actions in knowledge construction from the environment. As the hierarchy of *schemes* grows higher, which representing that they are responsible for more complex behaviors, therefore structures are termed. While structures become sophisticated, they could be organized in a hierarchical manner [73] which represents from general to specific.

For providing a formal model of constructivism, the habituation technique was introduced to explore the details of infant cognition. The habituation technique relies on a novelty preference in infants [77]. The habituation technique is described as thant "when an infant is presented with the same familiar scene repeatedly, the he will grow bored and look away from the scene, presumably in searching of something novel. However, if the he is presented with something new, the infant will stare longer" [85]. Thus, gaze duration could be used as a measure of novelty. The habituation experiment could be designed by using a scene that familiar in one way, and novel in another way. Such experiment can be used to determine how the infant is processing the scene [25].

With studies of infant cognitive development and the habituation technique, Cohen et al. [78, 79] demonstrated that primitives of infant's world model are acquired rather than innate. Specifically, infants organize stimuli into categories based on criteria and build higher-level representations by applying these criteria to lower-level representations. "Given this work, it suggests a possible path to implement a more accurate computational model of developmental cognition" [76].

A related but alternative idea is that the intrinsic motivation of curiosity can itself drive the development of world model making [18]. The idea relies on that the child pushes the boundaries of what its world-model-prediction systems can achieve, giving itself useful data on which to improve and develop these systems [1]. Related to the conception of the "scientist in the crib" [4], in which behaviors learning are an active learning process [10] and could be reorganized into highly structured, driving the self-supervised learning of a variety of representations underlying sensory judgments and motor planning capacities [81, 3, 2, 82].

2.2 Cognitive development of an autonomous agent

2.2.1 Constructivist learning

Constructivism as a knowledge acquisition theory suggests that learning happens as a result of an internal mental representations and external perceptions from interactions [72], rather than existing in an ontic reality, supposedly pre-existing or available for registration from the physical world [55]. In particular, the radical constructivism [83] describes that knowledge of the world comes from the "result of self-organizing construction" based on individual's experience. In the view of constructivist, the agent is not designed as a passive observer of reality, but rather constructs a perception of reality through active interaction experience [9].

Drescher [74] proposes a constructivist schema mechanism attempts to implement Piagetian constructivist learning. The constructed schema associated with context, actions, and expectations. In Drecher's implementation, schemas were neither associated with satisfaction value nor did the agent exhibit self-driven behavior. Particularly, the agent's exploration was rather random and resulted in a combinatorial explosion as the agent encountered increasingly complex environment.

Georgeon et al. [84] presented a self-motivated and hierarchical sequential learning model which inspired by Piaget's theories of early-stage developmental learning. In this learning model, the behavior organization is driven by pre-defined values associated with primitive behavioral patterns. Thus the agent increasingly learns elaborated behaviors through its interactions with its environment. Additionally, these learned behaviors are gradually organized in a hierarchy that reflects how the agent exploits the hierarchical regularities afforded by the environment.

Guériau et al. [12, 13] combined constructivism with reinforcement learning (RL) for autonomous and continuous learning of state space representations and their run-time adaptation, as a way toward fully self-adaptive RL. Following a constructivist learning perspective, the agent iteratively builds a mapping of its environment into a self perception-action states and the knowledge governs the decision process, through learning its experience of interacting with environment.

Based on the Information-Processing Principles (IPPs) introduced by Cohen et al. [78], Chaput et al. [25, 76, 85] combine with a hierarchy of Self-Organizing Map (SOM) and present a hierarchical self-organizing model with constructivist learning, as a way to simulate infants' cognitive development and flesh out the detail of learning process. Particularly, the model is aimed at modeling the constructive process of cognitive development as observed in infants. However, the Constructivist Learning Architecture (CLA) they proposed by using the scheme mechanism in each level of the architecture is depended on goals defined by the designer. Therefore, it relies upon a problem-solving approach that in fact differs from our motivations [9].

2.2.2 Intrinsic motivation

Intrinsic motivation [1, 17, 10, 15] (like curiosity, novelty, and surprise etc.) drives the development of the world-model making, as a way to replicate some abilities of infants' interactions [86]. Examples of intrinsic motivation systems like curiosity [17], search for predictability and control [87], and search for simplification of knowledge or compressibility of data [18]. With intrinsic motivation, the value system is generally not made explicit in the form of numerical values. Instead, it is implicitly characterized by the resulting

behavior of the system.

Driven by intrinsic motivations, Satinder et al. [19] adopt an evolutionary perspective and define a new optimal reward framework that captures the pressure to design good primary reward functions that lead to evolutionary success across environments. The result shows that optimal primary reward signals may yield both emergent intrinsic and extrinsic motivation. Savinov et al. [88] propose an intrinsic motivation model with curiosity and episodic memory for an autonomous agent obtain capabilities of self-adaptation. Specifically, the method stores agent's interactive experiences of the environment in an episodic memory, while also spur the robot for reaching experiences not yet represented in memory. Oudeyer et al. [17] present the mechanism of Intelligent Adaptive Curiosity as an intrinsic motivation system which pushes a robot towards situations in which it maximizes its learning progress. This drive makes the robot focus on situations which are neither too predictable nor too unpredictable, then emerging the autonomous mental development.

With intrinsic motivation of curiosity, Haber et al.[15] mathematically formalize a curiosity-driven intrinsic motivation with neural network and placed the agent in an ecologically naturalistic simulated environment. By combining with an error map, the agent then uses the self-model to adversarially challenge the developing world-model. Twomey et Westermann [20] present a formalization of the mechanism underlying infants' curiosity-driven learning during visual exploration and implement this mechanism in a neural network that captures empirical data from an infant visual categorization task. The results performed that maximal learning emerges when maximize stimulus novelty relative to its learning history, depending on the interaction across learning between the structure of the environment and the plasticity in the learner itself.

2.2.3 Self-adaptation and flexibility

Most studies in adaptive learning distinguish between a learning phase where the knowledge is acquired, and a performance phase where the learning is assessed. Aha [89] highlighted two categories of learning algorithms like: "eager learning and lazy learning". Eager learning algorithms as a way to compile input samples and use only the compilation to make decisions (e.g., reinforcement learning). Lazy learning algorithms sometimes perform little compilation and reuse the stored input samples to make decisions (e.g., schema mechanisms) [9, 10].

Reinforcement Learning (RL) [90] and with its advances [7, 28, 91] as a most popular and successful way to enable runtime adaptation, to learn the appropriate actions by learning to get the maximum cumulative reward of actions, which is used to help making decision of actions in response to interactions with the environment. However, the performance of algorithms become unsuitable and will need further self-adapting during the system execution in situations where environment changes. Various techniques have been developed to enable RL to tackle with such non-stationary environments, basically the main methods aim at continuously detecting and predicting environment conditions [13].

Baranes et al. [10] introduce an self-adaptive architecture with intrinsic motivation of curiosity that allows an artificial robot to efficiently and actively learn distributions of parameterized motor skills/policies that solve a corresponding distribution of parameterized tasks/goals. With result from the experiments, it demonstrated that: (a) exploration in the task space can be a lot faster than exploration in the actuator space for learning inverse models in redundant robots; (b) selecting goals maximizing competence progress creates developmental trajectories driving the robot to progressively focus on tasks of increasing complexity; (c) this architecture allows the robot to actively discover which

parts of its task space it can learn to reach and which part it cannot.

Focus on the limitations of the design of the reward function for complex behavior learning in Reinforcement Learning, Heess et al. [11] tried to explore methods from the perspective of rich environment to promote the learning of complex behavior. In particular, they trained multiple agents on a diverse set of challenging terrains and obstacle, combined with a novel scalable variant of policy gradient reinforcement learning, they demonstrate that this principle encourages the emergence of robust behaviors that perform well across a suite of tasks. Moreover, the agent learns to run, jump, crouch and turns as required by the environment without explicit reward-based guidance.

The enactive paradigm as a viable alternative to traditional computational approaches with respecting to the practical goal of building artificial agents, which allows an autonomous agent to behave in a robust and flexible manner under changing real-world conditions [43]. Based on the enactive paradigm, Georgeon et al. [44] propose an Enactive Cognitive Architecture that allows the agent to "autonomously discover, memorize, and exploit spatio-sequential regularities of interaction". Particularly, the ECA suggests that the cognitive development of an autonomous agent is on the basis of sensorimotor interactions with the environment, which indicates perception and action embedded within sensorimotor schemes, rather than separates them apart. Gay et al. [92] present an architecture for self-motivated agents to organize its behaviors in space according to possibilities of interactions afforded by initially unknown objects. The agent is designed to construct its own knowledge of objects through experience, rather than exploiting precoded knowledge. Experiments from this work with a simulated agent and a robot show that they learn to navigate in their environment, taking into account multiple surrounding objects, reaching or avoiding objects according to the valence of the interactions that they afford.

2.3 Cognitive architecture

Cognitive architecture as a part of research in Artificial general intelligence (AGI) which aims at enabling to reconstruct human-level intelligence in the fields of AI and computational cognitive science. With the definition of cognitive architecture from the *Institute for Creative Technologies* as: "hypothesis about the fixed structures that provide a mind, whether in natural or artificial systems, and how they work together – in conjunction with knowledge and skills embodied within the architecture – to yield intelligent behavior in a diversity of complex environments" [1]. One of the ultimate goals of a cognitive architecture is to summarize the various results of cognitive psychology in a comprehensive computer model.

During the last decades, a large number of cognitive architectures have been developed and have achieved great success both in theory and in application. Classic cognitive architectures, such as Soar [93],ACT-R [94], EPIC [95], MicroPsy [96], CLARION [97], ICARUS [98] etc. have been greatly improved in ease of use and robustness after long-term continuous developments and improvements [99, 100, 101, 102, 103, 104]. At the same time, the newly proposed cognitive architectures, such as CLA [85], LIDA [105], ECA [44], ECRL [88], which integrate various new ideas and cutting-edge implementation technologies in the field of artificial intelligence and cognitive development, and thus has better performance in the flexibility and autonomy. We respectively introduce and explain several typical cognitive architectures below.

[1] Cited from https://en.wikipedia.org/wiki/Cognitive_architecture

2.3.1 The Soar Cognitive Architecture

As a general cognitive architecture, Soar was originally created by Laird et al [93] and integrates a multiple of learning theories related to knowledge-intensive reasoning, reactive execution, hierarchical reasoning, and learning from experience. The goal of the Soar as mentioned that "is to develop the fixed computational building blocks necessary for general intelligent agents – agents that can perform a wide range of tasks and encode, use, and learn all types of knowledge to realize the full range of cognitive capabilities found in humans, such as decision making, problem solving, planning, and natural language understanding" [106]. Since its beginnings of creation and long-term continuous development and improvement, it has been widely used by AI researchers to create intelligent agents and cognitive models of different aspects of human behavior.

The cognitive architecture of Soar is not designed for solving a specific problem. Instead, it is the task-independent infrastructure to bring an agent's knowledge focus on generating appropriate behavior. Furthermore, the learning mechanisms in which constitute the agent's memories upon interaction experience. Over years of development and improvement, the current version of Soar (as shown in Figure 2.1) features major extensions that support multiple long-term memory systems (procedural, episodic, and semantic), multiple learning mechanisms (chunking, reinforcement learning, semantic, and episodic learning), and multiple representations of knowledge (symbolic, numeric, and imagery-based representations) [93].

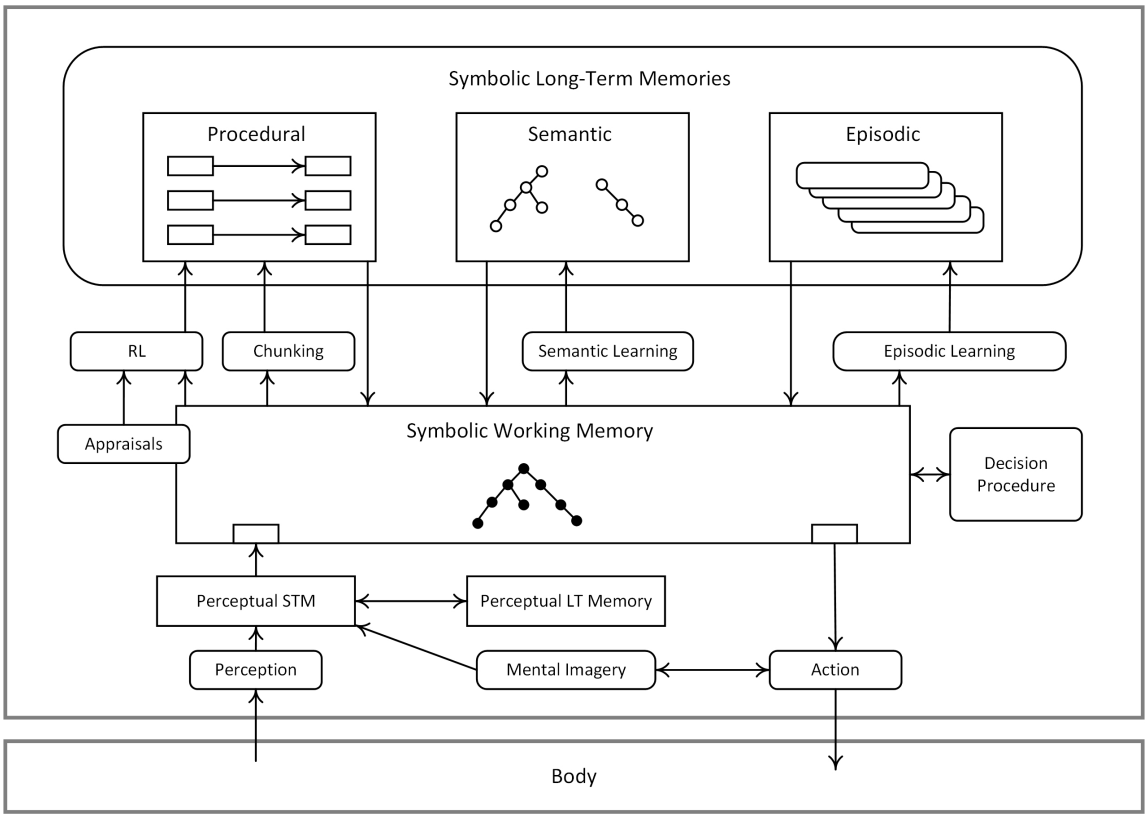


Figure 2.1: The structure of Soar 9 from Laird et al. [106], which features major extensions of reinforcement learning, semantic memory, episodic memory, mental imagery, and an appraisal-based model of emotion.

Above all, the cognitive architecture Soar is both a theory of what computational structures are necessary to support human-level agents and an implementation of that theory. However, it follows a problem solving approach and treats inputs as percepts that

expresses the realist paradigm of Artificial Intelligence (as we mentioned in Section 1.1), which does not match with our requirement.

2.3.2 Constructivist Learning Architecture

The Constructivist Learning Architecture (CLA) is a computational model of infant cognitive development that was built using the Information-Processing Principles (IPPs) as a general design specification. Unlike many other computational models, CLA is not intended to be an "existence proof" which shows that something can be learned. Rather, CLA is an attempt to model the constructive process of cognitive development as observed in infants.

The implementation of the CLA combines with the Self-Organizing Maps (SOM) [107], an kind unsupervised learning style system based on neuroscientific principles. Particularly, the CLA uses a SOM for simulating each layer of unit, and connects multiple layers hierarchically using a Hebbian learning algorithm [108]. Specifically, the lowest level organizes simple perceptual stimuli into a set of prototypes positioned within a topographical map. The next level organizes the information represented by the lower-level map into yet another map of prototypes. Levels may continue to be added as needed. Thus, CLA is a simple-to-compound system that is consistent with the evidence for constructivist learning [85].

Figure 2.2 provides a schematic overview of the different layers of the model. Especially, the launching events are captured with diverse spatial and temporal components. The two input feature vectors represented for each of components of the launching events. The Position Input as the first input by representing the position of the two balls in time. The Speed Input represented the speed of each ball which is provided the temporal information of the events. These two parallel input vectors simulate the stimuli of the model by using features that we know even young infants can perceive. Additionally, each of these input vectors projects to a corresponding layer, the Position Input to the Position2 Layer, and the Speed Input to the Speed2 Layer. These second level layers represent, as prototypes, the different positions and speeds separately over the course of an entire event. Finally, the Top Layer of the model receives inputs from both the Position2 and Speed2 Layers. Its task is to represent the event as a whole and hopefully, after training, be able to distinguish causal from non-causal events.

2.3.3 Enactive Cognitive Architecture

The Enactive Cognitive Architecture (ECA) was introduced by Georgeon et al. as a novel way that "allows the agent to autonomously discover, memorize, and exploit spatial-sequential regularities of interaction afforded by the coupling of the agent and the environment" [44] . The learning model in the ECA is called *Enactive Markov Decision Process (EMDP)*. In ECA, EMDP keeps perception and action embedded in sensorimotor schemes, rather that dissociates them separately. Moreover, instead of seeking specific goals with a well defined reward function (as reinforcement learning paradigm), the EMDP agent actively learns to continuously master the sensorimotor contingencies from its interaction experience with the environment. In doing so, the agent exhibits a form of intrinsic motivation related to the autotelic principle [109], and a value system attached to interactions called interactional motivation. From the perspective of cognitive science, this modeling approach allows the design of agents capable of autonomous self-programming, which provides rudimentary constitutive autonomy—a property that theoreticians of enaction consider necessary for autonomous sense-making.

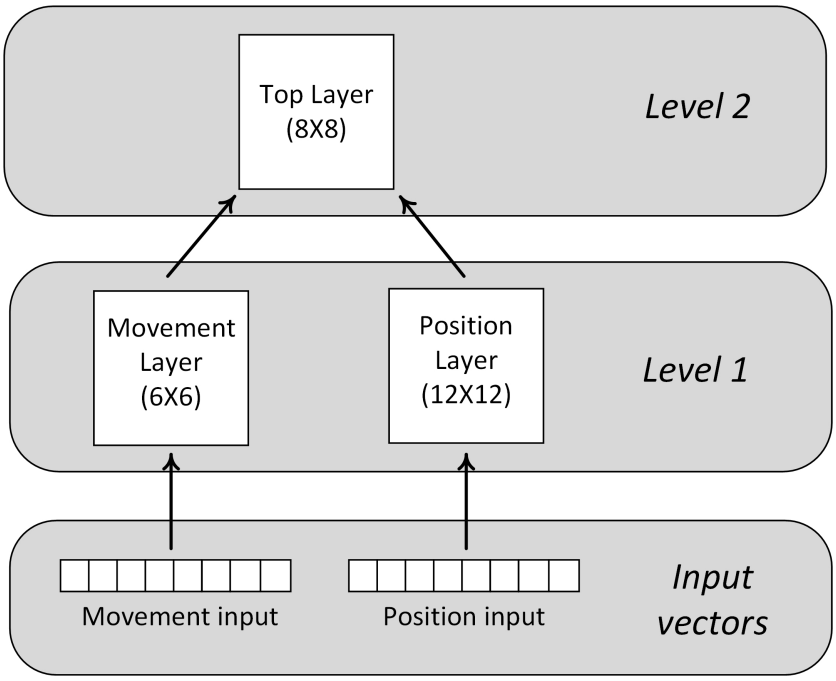


Figure 2.2: A schematic of the layer setup used in CLA model of an infant's development of causal understanding.

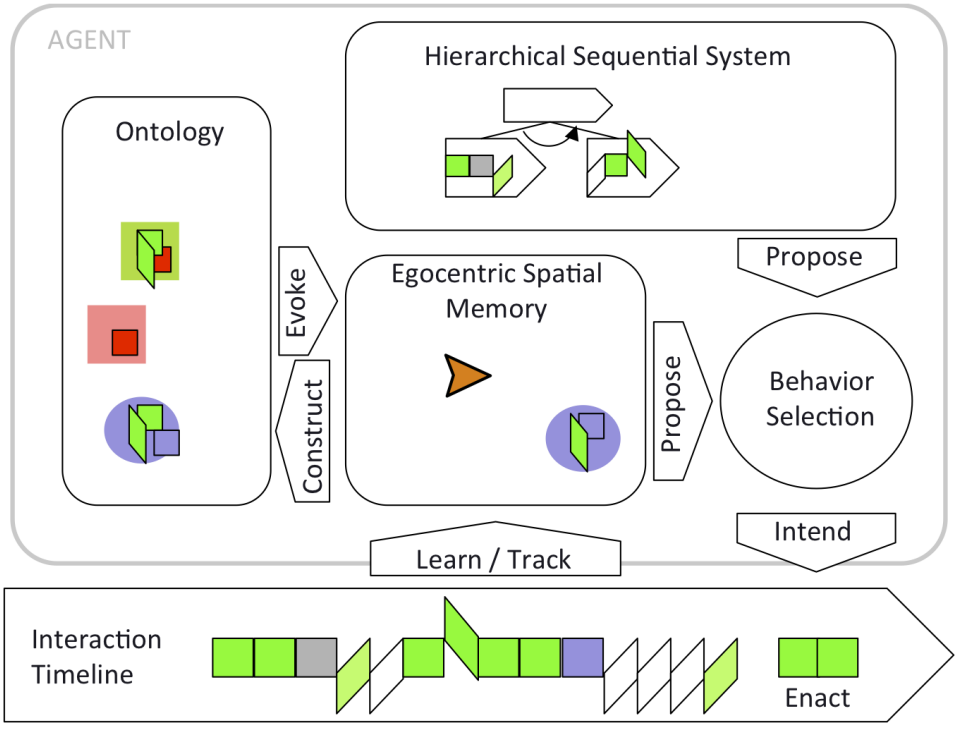


Figure 2.3: The Enactive Cognitive Architecture (ECA).

Figure 2.3 gives an overview of the Enactive Cognitive Architecture (ECA). (a) At the bottom, the Interaction Timeline shows the stream of interactions enacted over time. Enacted interactions are represented by colored symbols to indicate different interaction situations. (b) On the top, the Sequential System represents the hierarchical sequential regularity learning mechanism of EMDP. (c) In the center, Spatial Memory keeps track of the position (relative to the agent) of enacted interactions over the short term. When the agent moves, spatial memory is updated to reflect the relative displacement of enacted interactions. (d) On the left, the Ontology mechanism records bundles of interactions based on their spatial overlap observed in spatial memory. (e) On the right, Behavior Selection mechanism balances the propositions made by the sequential system and by spatial memory, and then selects the next sequence of interactions to try to enact.

2.4 Summary

Focus on designing an autonomous agent to obtain learning abilities that infants have, we suggest a review in this chapter. We start with recent developments in the field of studies in learning mechanisms of infants' early-stage cognitive development. In particular, theories of constructivism, information-processing principles and intrinsic motivations (like curiosity, novelty etc.) in infants' cognitive development have been largely developed and improved with considerable success. Base on these theories, large numbers of models and approaches are being proposed to encourage an autonomous agent to acquire the knowledge of the environment from its experience and learn to organize its behavior to fulfill a from of intentionality which is defined independently of a specific task.

Furthermore, we did a survey of developments in cognitive architectures and introduced several successful and classic cognitive architectures, which include Soar, Constructivist Learning Architecture (CLA), and Enactive Cognitive Architecture (ECA).

Inspired by the theories development in the filed of early mechanisms in infants' cognitive development and recent developments in artificial intelligence, we introduce a new computational model of cognitive architecture (refers to Chapter 4) for designing an autonomous agent to acquire the perception of the environment and obtain capabilities of self-adaptation and flexibility for generating proper behaviors to tackle with diverse situations, as a way to simulate the early mechanism of infants' learning process.

Chapter 3

Foundations

Contents

In this chapter, we provide the foundations behind this computational model of Constructivist Cognitive Architecture (CCA) and pave the way for the next chapter to describe the design, the structure and the implementation of CCA. The foundations are divided into two parts: the theoretical foundations and the implementation foundations. Since the central of the dissertation is that a model based on the constructivist paradigm to simulate the early mechanisms of infant's cognitive development, we started with introducing Piaget's theory of cognitive development and the constructivism. The theories of radical constructivism and radical interactionism provide viable approaches to implement the constructivism into an autonomous agent. Moreover, the enactive paradigm originally emerged as a part of embodied cognitive science endows the self-developing agent with a self-motivation, which spurs the agent to construct regularities of interaction afforded by the environment.

In the part of implementation foundations, we explain the representation and operations of schemes in computer science. Two benchmarks of Small Loop Problem (SLP) and a pedagogical game named "Little AI" are introduced for validating CCA's abilities in Chapter 5 and Chapter 7. In the following chapters, these two foundations are merged to form the computational model that is the main contribution of this dissertation: the Constructivist Cognitive Architecture, or the CCA.

3.1 Theoretical foundations

3.1.1 The theory of developmental psychology

The modern study of infant cognition can truly be said to have started with the research of Piaget [50]. His seminal work in the area of infant cognition presented for the first time a comprehensive picture of infant cognitive development, and provided a theory of infant cognition called the *Developmental Stage Theory*.

To better understand some of the things that happen during cognitive development, it is important first to examine a few of the important ideas and concepts introduced by Piaget. Piaget's cognitive theory mainly includes the following basic components that influence how children learn and grow: (1) schemes as building blocks of knowledge, (2) stages of cognitive development, and (3) adaptation processes that enable the transition from one stage to another.

Schemes

Schemes are described as elementary building blocks in cognitive development, by which organizing knowledge that enables human to perform a mental representation of the environment. In the definition of scheme from Piaget [111], that "The scheme is a cohesive, repeatable action sequence possessing component actions that are tightly interconnected and governed by a core meaning." Specifically, a scheme includes both a category of knowledge and the process of obtaining that knowledge [110]. As experiences happen, the newly obtained information is used to modify, append to previously existing schemes. One of the assumptions comes from that human stores mental representations and applies them when needed [111]. For example, a person might have a scheme about studying in the library. The scheme is a stored form of the pattern of behavior which includes find an available table and a comfortable chair, plug the charger, open the computer and start self-studying. This is an example of a type of scheme called a "script". Whenever they want to study in the library, they retrieve this scheme from memory and apply it to the

situation. The schemes Piaget described tend to be simpler than this, especially those used by infants. He described how as a child gets older his or her schemes become more numerous and elaborate.

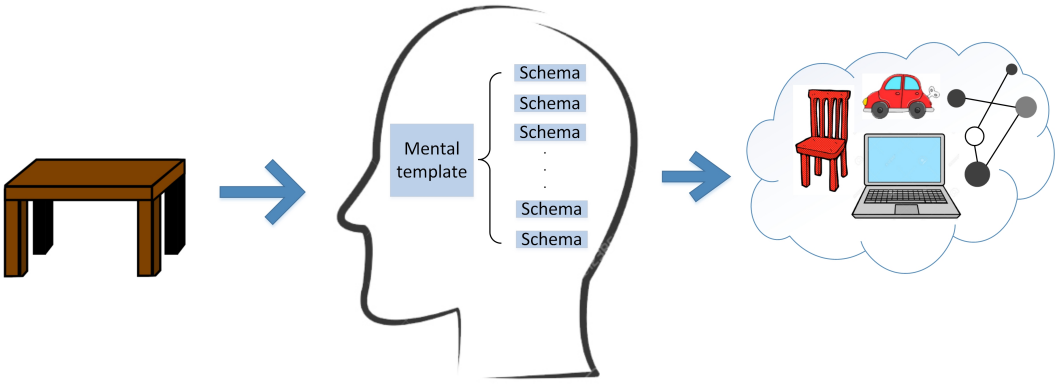


Figure 3.1: The scheme (or "script") of starting self-studying in the library by retrieving schemes from the memory about taking a chair, finding a available table, going to the power socket and plugging the charger, opening the computer and starting self-studying.

With theories from Piaget that newborn babies have a small number of innate schemes-even before they have had many opportunities to experience the world [111]. As the infants use reflexes to adapt to the environment, these reflexes are quickly replaced with constructed schemes. The neonatal schemes are the cognitive structures underlying innate reflexes and are genetically programmed into us. For example, babies have a grasping reflex, which is elicited when something touches the palm of a baby's hand, or the rooting reflex, in which a baby will turn its head towards something which touches its cheek, are innate schemes. Shaking a rattle would be the combination of two schemes, grasping and shaking.

The four stages of cognitive development in infants

With the cognitive development in Piaget's theory, it suggests that children go through 4 distinct stages which reflect the development of sophistication in children's thought, which relate to aspects of mental development including that of reasoning, language, morals, and memory. Specifically, each child goes through the stages in the same order, and the development progress is determined by biological maturation and interaction with the environment. The four stages of cognitive development include [1]:

- Sensorimotor stage (Infancy, Birth-2 years). In this period, intelligence is demonstrated through motor activity without the use of symbols. Infants know the world through their basic actions such as sucking, grasping, looking and listening. Knowledge of the world is limited(but while also developing) because its based on physical interactions/experiences.
- Pre-operational stage (Toddlerhood through Early Childhood, 2-7 years). In this period, intelligence is demonstrated through the use of symbols, language use matures, and memory and imagination are developed, but thinking is done in a nonlogical, nonreversable manner.

[1] Cited from https://www.verywellmind.com/piagets-stages-of-cognitive-development-2795457, Section 2

- Concrete operational stage (Elementary and early adolescence, age 7 to 11 years). In this stage (characterized by 7 types of conservation: number, length, liquid, mass, weight, area, volume), intelligence is demonstrated through logical and systematic manipulation of symbols related to concrete objects.

- Formal operational stage (Adolescence to adulthood, 11 years and over). In this stage, intelligence is demonstrated through the logical use of symbols related to abstract concepts and logically test hypotheses.

Adaptation processes between four stages of cognitive development

In the view of constructivism, children take an active role in the learning process, acting much like little scientists as they perform experiments, make observations, and learn about the world [112]. As kids interacting with the world around them, they continually add new knowledge, build upon existing knowledge, and adapt previously held ideas to accommodate new information. This adaptation processes happen through (a) *Assimilation* which is using an existing scheme to deal with a new object or situation. (b) *Accommodation* which happens when the existing scheme (knowledge) does not work, and needs to be changed to deal with a new object or situation, and (c) Equilibration the force which moves development along. Each process is not developed or operated in isolation, but enables and mutually supports learning and development in the others. Moreover, these processes are used throughout life as the person increasingly adapts to the environment in a more complex manner.

The *assimilation* refers to a part of the adaptation process which initially proposed by Piaget, as one of the processes by using or transforming the environment so that it can be placed in preexisting cognitive structures. Through assimilation, the agent incorporates new information or experiences into already existing cognitive schemes (or knowledge) [113], sometimes that reinterprets these new experiences so that they will fit in with previously existing old information.

While another part of adaptation involves of modifying or altering existing schemes to accept new information from the environment, or called as *accommodation*. This happens when the existing scheme does not work, and needs to be changed to deal with a new object or situation [114, 115]. Accommodation is necessary because it is how people will continue to interpret new concepts, schemes, and more [116]. The process of accommodation involves changing current cognitive structures, as a result of newly acquired external knowledge or new experiences [12, 117]. New schemes could also be developed during this process. With this procedure, schemes become more refined, detailed, and nuanced as new information into knowledge.

Additionally, assimilation and accommodation cannot exist without the other [116]. They are two sides of a coin. For keeping the balance between these two functions of assimilation and accommodation, the *equilibration* is proposed as the force that drives the learning process keeps going. Equilibrium occurs when a child's schemes can deal with most new information through assimilation. However, an unpleasant state of disequilibrium occurs when new information cannot be fitted into existing schemes (assimilation). Equilibration is the force which drives the learning process as we do not like to be frustrated and will seek to restore balance by mastering the new challenge (accommodation). Once the new information is acquired, the process of assimilation with the new scheme will continue until the next time we need to make an adjustment to it [118]. As shown in Figure 3.2, all adaptation processes are used simultaneously and alternately throughout life as the agent increasingly adapts to the environment in a more complex manner.

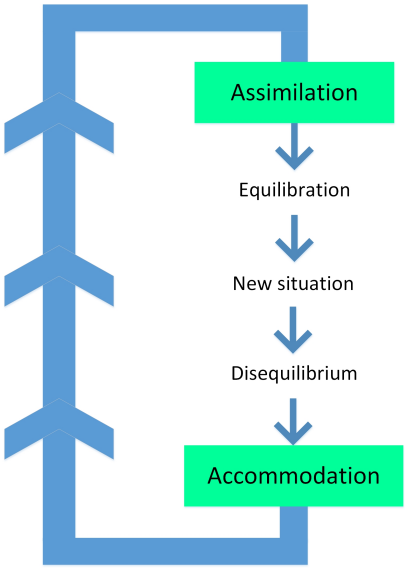


Figure 3.2: Three processes of assimilation, accommodation and equilibration. When the agent faced with a new situation, it takes this new information in existing schemes is known as assimilation. Accommodation involves modifying existing schemes to accept new information. Equilibration is the force moves development along.

3.1.2 Radical constructivism

The radical constructivism as a theory of knowing concerns the relation between the knowledge and the reality, which basically focusing on two classic epistemological problems of: (a) how can acquire the knowledge of reality and (b) how reliable and "true" that knowledge might be. Particularly, the radical constructivism suggests that "the knowledge does not reflect an 'objective'fe ontological reality, but exclusively an ordering and organization of a world constituted by experience" [119].

Based on the fundamental feature of constructivist epistemology, the radical constructivism proposes a prototype of knowledge acquisition in cognitive organisms which describes that the world is constructed as an experiential world that consists of interaction experiences and have no clue about the judgment of "truth" in the sense of feedback from the interaction with the reality. As stated by *Von Glasersfeld* [83] that "the cognitive activity takes place within the experiential world involves a purpose to establish regularities that allows to repeat certain cognitive activities that well serve this purpose and avoid the others based on the evaluation of its experience".

Furthermore, it proposes constructivist approaches that concern the nature of regularities which a cognitive organism looks for invariant experience. Particularly, the experiential sequence was constructed with the comparison between the past perception or experience and the relation between them, or an activity call like "putting-in-relation". And also it makes possible to consider repeated experiences as particular *objects* and to connect them with a space that is "independent of the subject's own motion and into a time independent of the subject's own stream of experience" [120].

3.1.3 Enactive cognition

In cognitive science, there has been a traditional tripartite division of the mind between perception, the control system and motor action. As Georgeon, Manzotti and Marshall [44] (p46) noted that this view has been nicely dubbed as the "classic sandwich model" by

Susan Hurley [121]. Based on this understanding, a large number of control architectures are developed by following this way. Since 1980s, different attempts gradually emerged to face this traditional view, especially in the field of robotics [122] but also from a more psychological and theoretical perspective [123, 124, 125]. In particular, the view conveyed an idea that it might be a mistake to consider sensation independently from action and that cognitive system should be designed on the basis of low-level sensorimotor loops that represent sensorimotor patterns of interaction. This intuition gained momentum from other related views such embodied cognition [126, 127], ecological psychology [128, 129], sensorimotor theories [54, 130], morphological robotics [131, 132, 132], evolutionary robotics [133, 134], developmental robotics [35], and epigenetic robotics [135, 136].

Especially, the enactive approaches were introduced as that "perceptual experiences are not events that are internal to our heads, but are rather something which we enact or bring forth through our engagement and sensorimotor exploration of our environment" [43]. More specifically, perception is an exploratory activity, and in particular that vision is a mode of exploration of the environment that is mediated by knowledge of sensorimotor contingencies. The enactivist approach suggests modeling a cognitive agent on the basis of sensorimotor interactions with the environment.

In this dissertation, we introduce a modeling approach that goes a step beyond the notion of low-level sensorimotor loops by simply considering sensorimotor patterns - also called *sensorimotor schemes* by Piaget [137] - as the atomic elements manipulated by our algorithms.

3.1.4 Motivations in agent's cognitive development

Motivation is generally defined as a driving force that initiates and directs behaviors. In this dissertation, we introduce two forms of self-motivation: the motivation of enjoyment being in control of one's own activity by seeking to successfully enact interactions, and the motivation to enact interactions have predefined positive values and to avoid enacting interactions have predefined negative values. We call the former as *autotelic motivation*, and the latter *interactional motivation*.

In traditional computational models, motivations is presented as an agent's decision making for behaviors selection based on maximizing a value function, typically, the reinforcement learning (RL) algorithms [138, 30] try to simulate an agent learns to perform actions with the best rewards [139]. The designer of the RL interprets a state $s \in S$ as a specific configuration of an agent in a virtual environment, and interprets $a \in A$ as an action that the agent performs in the environment according to the agent's policy. In this case, the agent's motivation is regarded as the *extrinsic motivation* [140] with the reason of that this reward function $r : S \rightarrow \Re$ is defined independently from the agent's policy. Additionally, within designer's view that "agent's motivation seems to come from something rewarding in the environment that is external to the agent itself" [2].

Autotelic motivation

Autotelic motivation as a form of intrinsic motivation related to the autotelic principle, which proposed by which an agent could self-regulate its build-up skills and knowledge without the intervention of the designer to scaffold the environment, stage the reward function , or bring resources progressively online in a maturational schedule [109].

[2]Cited from Implementation of DEvelopmental learning from O.Georgeon, https://projet.liris.cnrs.fr/ideal/mooc/lesson.php?n=022

With the original definition of autotelic motivation from Luc Steels [109], it implies [3]

- Each component must be parameterised so that challenge levels can be self-adjusted based on self-monitoring of performance.
- Each component must have the ability to increase skill to cope with new challenge.
- A global dynamics regulating the adjustment (both increase and decrease) of challenge levels

The reward function in this motivation corresponds to the degree of balance between challenge and skill for each of its components. When the complexity of the agent's behavior increases, an emergent side effect of the system's effort to keep this balance between the challenge and the skill.

Interactional motivation

Interactional Motivation as a new form of self-motivation for artificial agents and robots to specify an inborn perception system without any prior knowledge of the environment. In the interactional motivation, the agent has no specific goal to achieve and even in some cases may lack the state of the environment. However, the agent has a set of predefined primitive interactions (as like schemes) that allow it to enact with the environment.

We call *valence* of interaction $v(i)$ the scalar value associated with the interaction i that represents the value of enacting this interaction and design the agent's policy that tries to enact interactions that have a positive valence. Consequently, the agent must construct knowledge of the world so that it can predict the consequences of enacted interactions and seek situations that lead to positive valences and avoid situations that lead to negative valences. This learning process is open-ended because the agent has no goals predefined as reward states.

While we knew that this type of interactional motivation is not the only possible motivational drives for sensorimotor agents. Recall that we introduced the drive to learn to enact the result of interactions. Bue more importantly, since there is not external data for the agent that directly represents the environment's state, it can hardly can be pre-designed to perform a predefined goal.

3.2 Implementation Foundations

Constructivism suggests that knowledge of reality is constructed from learning regularities of sensorimotor schemes afforded by the environment. And each scheme represents the bind relationships between the mental and physical actions involved in the obtain of knowledge of the environment. Particularly, as *schemes* increasingly become more complex, which represent that they are responsible for more complex behaviors, structures are termed. While structures become complicated, higher-level schemes can be built from the activation of lower-level schemes. Therefore, they could be organized in a hierarchical manner [73] which represents from general to specific.

Inspired by Piaget's constructivist epistemology, a large number of implementations called *schemes mechanisms* are springing up. Most of which concretized the scheme as a triplet of $[observation_1, action, observation_2]$ and referred to them with the term of

[3] Cited from [109], Selection 3, Page 7.

schema, for example Gary Drescher [74], Filipo Perotto et al. [141], and Michael Miller [142]. The agent explores its environment and records schemes with a particular action in a specific state ($observation_1$) would likely obtain a certain outcome ($observation_2$). However, this approach could lead to a combinatorial explosion when the environment's complexity grows gradually [9].

However, more recently, new models have been proposed based on couples (2-tuple) rather than triplets (3-tuple). Georgeon and Ritter [9] modeled the schemes in the form of $\langle interaction_1, interaction_2 \rangle$, which represents that in the context of $interaction_1$, the agent can enact $interaction_2$. A related but alternative idea comes from Kristinn Thórisson [143] in that a Causal-relational models (CRMs) could be created when the controller observes an event α and a subsequent event β that follows. The model can be seen as a hypothesis that the observed event α caused the observed event β, so that when observing again en event α in the future, this model will predict that β will be observed. Moreover, Brett Martensen [144] utilized Binons (binary neurons) to represent and implement the perception-action hierarchy. In particular, there exists two different types of binons: the spatial and the temporal, the spatial binons represent simultaneously occurring patterns of percepts and actions; The temporal binons represent sequential patterns of percepts and actions.

In our work, schemes are modeled as twosomes (2-tuple) and we follow the form of schemes from Georgeon and Ritter [9] of $\langle interaction_1, interaction_2 \rangle$ (as shown in Figure 3.3) , which describes the context in terms of interaction means that the agent learns to "see" its world in terms of affordances [129] related to its own prior experience. In this dissertation, we follow Gibson's definition of affordances as possibilities of interaction afforded to the agent by the environment. With this formalism, schemes natively encode entire sequences of interactions in a hierarchical style. In order to highlight this radical difference with classical scheme mechanisms, we prefer to keep Piaget's term *scheme* rather than using the term of *schema*.

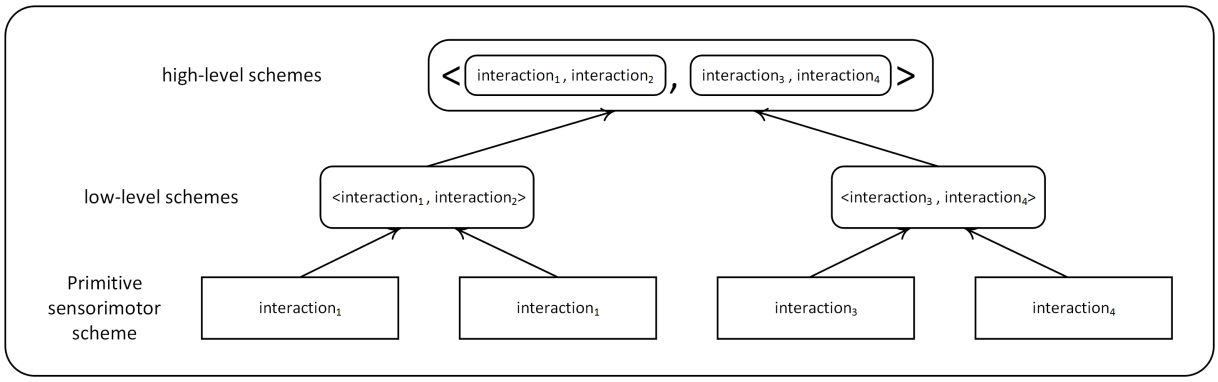


Figure 3.3: The form of schemes and the construction of higher-level scheme from lower-level schemes.

3.2.1 Representation and operations of schemes

Representation of schemes in computer science

As shown in Figure 3.3, a scheme is composed of two interactions in form, and higher-level schemes are also composed of two low-level schemes. In order to better express the construction and operation of schemes, we utilize a type of data structure in computer science which call binary tree (BT) to represent schemes, and through the operations on

this binary tree, we can realize the dynamic generation, construction and modification of schemes.

In computer science, the binary tree as a kind of tree structure, which is characterized that each node has at most two sub-nodes. The sub-nodes of the binary tree are divided into left and right, and its order is not allowed to be modified arbitrarily. In particular, the binary tree has a hierarchical structure. These features of the binary tree meet the requirements of schemes, which are composed of two lower-level schemes on the left and right, and these two lower-level schemes follow the left and right order to be encoded sequentially. The hierarchical structure of the binary tree could intuitively represents the structure of schemes.

In order to facilitate the storage and operation of the schemes in binary tree, we design each scheme node consists six parts (as shown in figure 3.4). In each node of the binary tree, (a) left node specifies the left lower-level scheme that composes the current scheme. For each primitive sensorimotor scheme plays as the basic block of schemes, its left node is initialized as *NULL* (as shown of leaf node at the bottom of Figure 3.4). (b) Layer level represents the level of current scheme. We define the layer level of basic primitive sensorimotor scheme as 0. Whenever a new scheme is constructed, its layer level is the largest layer level among the lower-level schemes that compose it plus 1. (c) Interaction information describes all primitive interactions that constitute the scheme and with its structure, as a convenient way to flat complex interactions down to a series of primitive interactions for facilitating to enact this complex interaction. This part will be described in detailed in *enacting composite interactions* of section 4.3.3 in the next chapter. (d) Upper node indicates its higher-level scheme and (e) upper direction specifies this lower-level schemes belongs to which branch of the higher-level scheme. For the top scheme which represents the whole sequence of complex behavioral patterns of interactions, its upper node is initialized as *NULL* and its upper direction is *NULL* also (as shown of middle-node at the middle of Figure 3.4). (f) Right node specifies the right lower-level scheme that composes the current scheme. Similarly, for each primitive sensorimotor scheme, its right node is initialized as *NULL* (as shown of leaf node at the bottom of Figure 3.4).

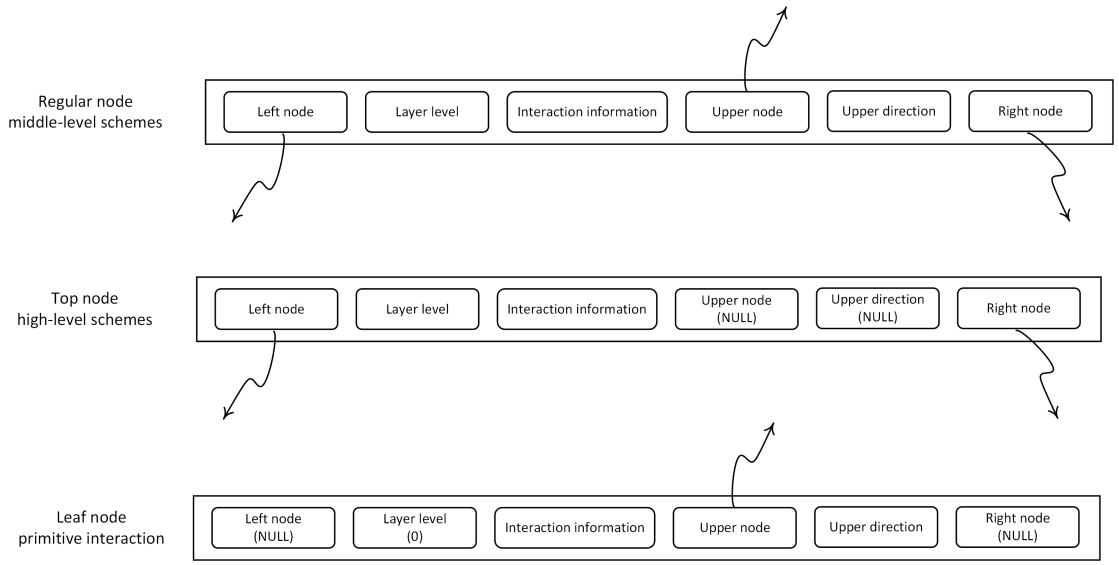


Figure 3.4: The structure of schemes in binary tree.

Operations of schemes

With similarities between the characteristics of scheme and binary tree, we utilize binary tree as a way to represent schemes that are produced in agent's interactions with the environment. Therefore, through operations on this binary tree, we can realize the dynamic construction of new schemes, executing the traversal of schemes in different ways, and locate any interaction in a specific scheme. The operations of schemes include initialization of a binary tree, traversal of this binary tree, and interaction localization.

Given a scheme, a new node is created according to the form of scheme node and initialize its upper node as *NULL* and upper direction as *NULL*. The newly created node is regarded as the root of the binary tree. Meanwhile, we initialize a stack *NodeStack* to iteratively visit each part of the root node and an empty list *NodeList* to store all nodes in this scheme, then we append this root bode in the NodeList and push it at the bottom of the NodeStack. Each time the NodeStack pops an element, we need to check whether this element is primitive scheme. If so, we proceed to pop the elements from the NodeStack. If not, we first create a new node according to the element' right lower-level scheme, setting this newly created node's upper node as this element and its upper direction to be "right". After then, append this newly created node in the NodeList and set it as element's right node. Finally, and most importantly, push the newly created node in the NodeStack. Similarly, we create another new node according to the element's left lower-level scheme, setting this newly create node's upper node as this element and its upper direction to be "left". Then append this newly created node in the NodeList and set it as element's left node, push it in the NodeStack. When the NodeStack is empty and no other elements are added, the presentation of the scheme and the construction of the binary tree are completed.

Note that the right node is first created when a non-primitive scheme is popped from the NodeStack, there exists two reasons: (a) stack as a specific type of data structure which allows the element that first enters the stack but the last to be popped out, and the element that last enters the stack as the first to be popped out. (b) The lower-level schemes that compose the higher-level scheme have a left-to-right order. The right node is pushed into the stack first is the last to be popped out. And the left node is the last to be pushed into the stack but the first to be popped out the stack. In this way, the primitive schemes that appear after continuous iteration are carried out in the order from left to right, which can facilitate the traversal and positioning of the next node.

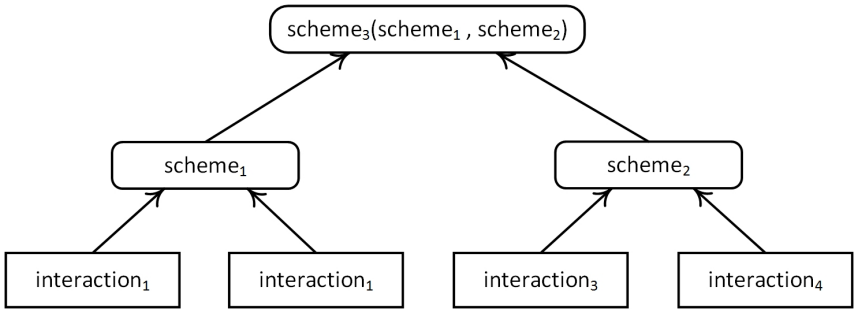


Figure 3.5: An simplified example of the schemes.

As an example, we reuse the Figure 3.4 with a slight simplification (as show in Figure 3.5). The higher-level consist of two lower-level schemes, and each lower-level scheme is composed by two primitive sensorimotor interactions. As illustrated in the Figure 3.6, at the beginning of building the binary tree, the node of $scheme_3$ is the first to be created and pushed into the NodeStack. After the $scheme_3$ is popped out, we find that it is not a

primitive scheme, then the nodes of its right scheme ($scheme_2$) and left scheme ($scheme_1$) are created and pushed into the NodeStack respectively. When the $scheme_2$ is popped out, finding that it isn't a primitive scheme, nodes of its right scheme ($interaction_4$) and left scheme ($interaction_3$) are created and pushed into the NodeStack separately. After popping out the scheme ($interaction_4$), we find that it's a primitive scheme, then continue popping element $interaction_3$ in the NodeStack. Same way with the $scheme_1$, then we receive a series of sequences of primitive schemes: $interaction_1$, $interaction_2$, $interaction_3$, $interaction_4$, which are consistent with their order in $scheme_3$. Algorithm 1 gives an explanation of reconstruction of a scheme in the binary tree.

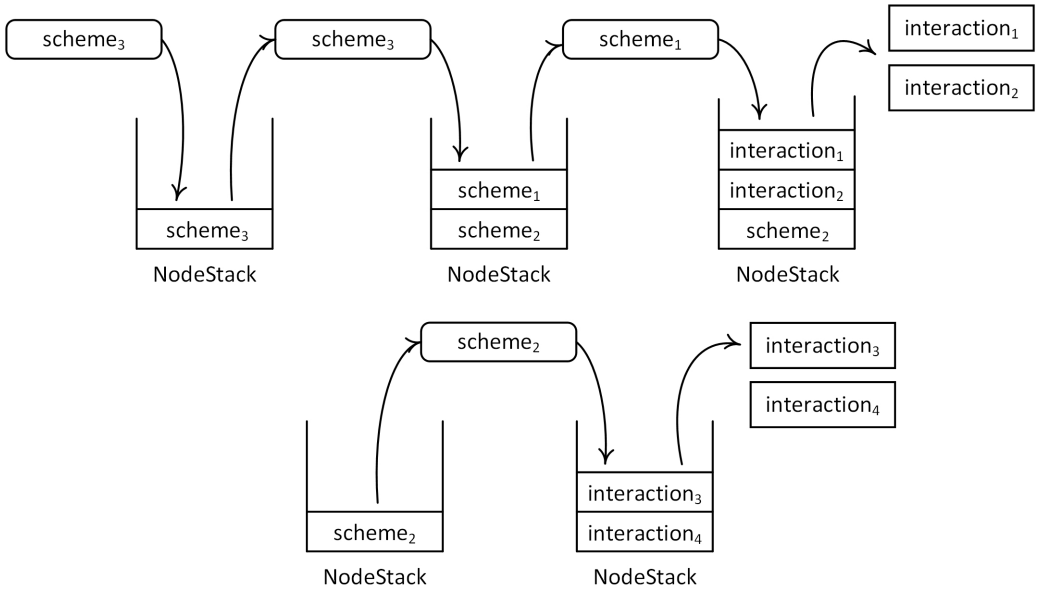


Figure 3.6: The process of construction composite interactions in binary tree.

Algorithm 1 The algorithm of the representation of the scheme and the construction of the binary tree.

```
Initial:
    root = createNode(given scheme);
    root.upperScheme ← NULL; root.upperDirection ← NULL;
    NodeStack = [ ], NodeList = [ ], PrimitiveSchemeList = [ ];
    NodeStack.push(root);
    NodeList.append(root);
While (NodeStack is not empty)
    upperNode = NodeStack.pop()
    if (upperNode is not primitive scheme)
        rightNode = create(upperNode.rightScheme);
        rightNode.upperScheme ← upperNode;
        rightNode.upperDirection ← "right";
        upperNode.rightNode ← rightNode;
        NodeStack.push(rightNode);
        NodeList.append(rightNode);
        leftNode = create(upperNode.leftScheme);
        leftNode.upperScheme ← upperNode;
        leftNode.upperDirection ← "left";
        upperNode.leftNode ← leftNode;
        NodeStack.push(leftNode);
```

```
21:       NodeList.append(leftNode);
22:    else
23:       PrimitiveSchemeList.append(upperNode);
24:       upperNode = NodeStack.pop();
```

In Algorithm 1, Line 2: create a new node with a given scheme and regard it as the root; Line3 and 4: initialize the NodeStack, NodeList, and the PrimitiveSchemeList, set root's upperScheme and upperDirection are *NULL*; Line 8 and 9: pop out an element from NodeStack and check the element whether it is a primitive scheme. Line 10 to 15: if the popped out element isn't a primitive scheme, a new node is created with according to this element's right scheme. Setting the newly created node's upper node as this element and its upper direction to be "right". After then, set the newly created node as element's right node, append it in the NodeList and push it in the NodeStack. Line 16 to 21: a new node is created with the popped element's left scheme, setting this newly create node's upper node as this element and its upper direction to be "left". Then set this newly created node as element's left node, append it in the NodeList and push it in the NodeStack. Line 23, if the popped element is a primitive scheme, then append this continue popping elements from the NodeStack. When the NodeStack is empty and no other elements are added, the algorithm ends.

In the operation of schemes, it is usually required to implement the traversal of all the primitive schemes in a given scheme, localize the first primitive scheme, and find the next primitive scheme given the current primitive scheme. In the traversal of all primitive schemes, we initialize a list of *primitiveSchemeList* as a set of all primitive schemes. With the NodeList which records all nodes from the reconstruction of a given scheme, we iteratively visit each node in it and examine whether it is a primitive scheme. The primitiveSchemeList records each primitive scheme until the end of the NodeList (as shown in the function of *localizeNextNode()* in Algorithm 2) . The first element in the primitiveSchemeList is thus the *first primitive scheme*. Given a primitive scheme $scheme_i$, the localization of the next primitive scheme is $scheme_{i+1}$.

Algorithm 2 Operations in the binary tree.

```
 1: Initial:
 2:    NodeList ← binaryTreeReconsturction(given scheme);
 3:    rootNode ← NodeList.get(0);
 4:    given primitive scheme: scheme_g;
 5:
 6: traversalScheme(NodeList)
 7:    primitiveSchemeList = [ ];
 8:    For (each node in the NodeList)
 9:       if (the node is primitive scheme)
10:          primitiveSchemeList.append(node);
11:
12: getFirstNode (rootNode)
13:    currentNode ← rootNode
14:    While (currentNode is not primitive scheme)
15:       currentNode ← currentNode.leftNode;
16:    return currentNode;
17:
18: localizeNextNode (currentNode)
19:    pointNode ← currentNode
20:    if (pointNode is not the last node of NodeList)
```

```
21:       While (pointNode.upperDirection is "right")
22:         pointNode ← pointNode.upperNode;
23:       if (pointNode.upperDirection is "left")
24:         pointNode ← pointNode.upperNode.rightNode;
25:       While (pointNode is not primitive scheme)
26:         pointNode ← pointNode.leftNode;
27:         return pointNode
28:     else
29:       pointNode ← NULL;
30:     return pointNode;
```

Considering with the hierarchical structure of the scheme, given with the NodeList, the localization of the first primitive scheme could be completed from recursively visiting all left nodes from the root to the leaf node of the binary tree (as shown in the function of *traversalScheme()* in Algorithm 2). Given with current primitive scheme for looking for its next primitive scheme, we should first check the current primitive schemes is the last node of NodeList or not. If the current primitive node is the last node of NodeList, which indicates all primitive schemes have been visited, the next primitive node of the current primitive node is thus *NULL*. Otherwise, the current primitive is not the last node of NodeList, obtaining its next node depends on its upper direction. If current primitive scheme's upper-direction is "left", we just need to find the first primitive scheme of its upper-node's right-node. If current primitive scheme's upper-direction is "right", we need to recursively trace the current primitive scheme's upper-node until its upper-direction is "left", then we go to look for the first primitive scheme of its upper-node's right-node (as shown in the function of *localizeNextNode()* in Algorithm 2).

3.2.2 Benchmarks

Small Loop Problem

Small Loop Problem (SLP) as a new benchmark to the implementation and demonstrate four principles of emergent cognition: "environment-agnosticism, self-motivation, sequential regularity learning, and spatial regularity learning" [145] .

Firstly, the principle of environment-agnosticism was proposed to account for that the agent should not rely on any prior knowledge of the environment. According to this principle, the SLP requires that the designer of the agent must not include predefined knowledge of the environment in the agent.

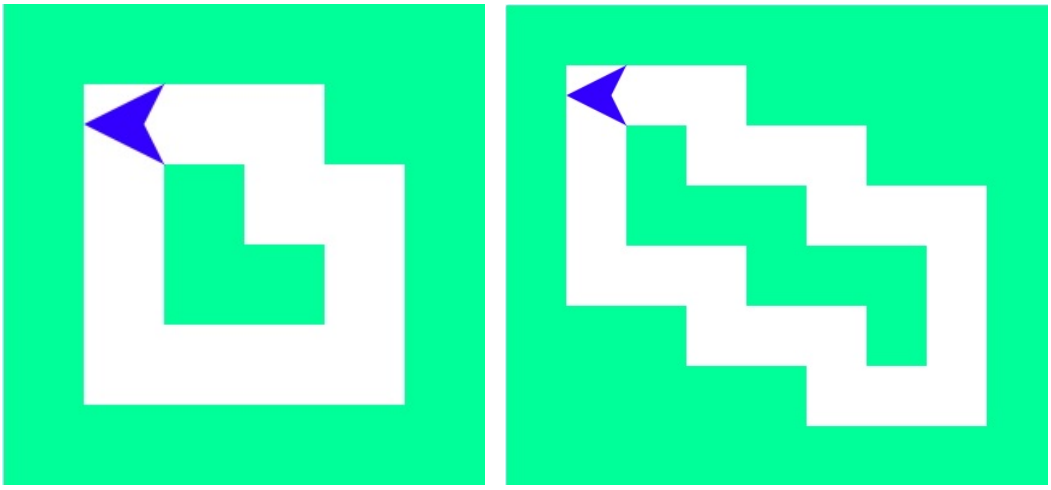
Secondly, the principle of self-motivation implies that learning a internal policy $P(t)$ to maximize a internal reward function $V(t)$, which presents that agent learns to fulfill an innate value system. Assumptions on this view describe that such an innate value system goes through an evolution-style according to natural organisms to favor the survival of the organism and of its species.

Thirdly, the principle of sequential regularity learning aims at that the agent need to discover, learn, and exploit interaction patterns from its experiences to maximize self-motivation internal function $V(t)$. Additionally, this principle follows the principle of principle of environment-agnosticism and remains an open challenge, which the goal that SLP intends to address.

Finally, the agent could use principle of sequential regularity learning for spatial regularities. Basically, natural organisms have inborn structures that prepared for dealing

with space, thus the spatial regularity learning is a key feature which the SLP also intends to integrate with a biologically inspired cognitive architecture.

In the design of SLP, it consists of squares surrounded by walls shown in Figure 3.7 and the agent is presented as a head arrow with a random initialization direction. Note that the SLP differs from benchmarks traditionally used in AI (e.g., [146]) by the fact that the environment does not offer a final goal to reach.



(a) An example of SLP environment (b) Another example of SLP environment

Figure 3.7: Examples of Small Loop Problem environment

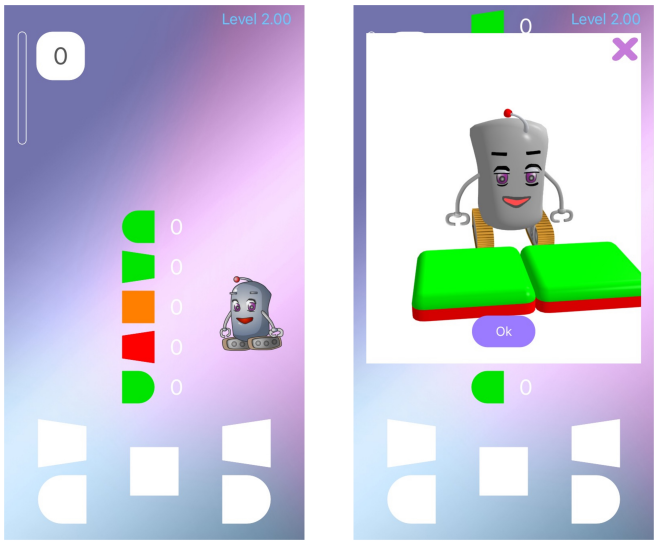
In the SLP, the agent has a endowed with a set of possible actions $A = \{a_1, \ldots, a_i\}$ with possible observations $O = \{o_1, \ldots, o_j\}$. The set of primitive interactions is defined as $I = A \times O$ but the agent has no initial knowledge of these interactions' meaning. Paticularly, each primitive interaction $i = [a_p, o_q], 1 \leq p \leq i, 1 \leq q \leq j$ associated with numerical value v_i for simulating its nature preference, hence the agent prefers to enact interactions have larger value and avoid enacting interactions that have smaller values. The value function $V(t)$ equals to the value v_i of the interaction i at step t mentioned above in the principles. The policy function aims to select the action a_j that would maximize the value function in an infinite steps. Additionally, within each decision cycle, the best action is selected usally does not depend on a single previous interaction but also rely on a sequence of interactions and on the possibility of enacting several next interactions. This makes the SLP more suitable for demonstrating the structured behaviors construction process detail, also as the highest-level which most satisfying sequence can be repeated indefinitely.

Little AI

Little AI is a pedagogical game aimed at presenting the fundamental concepts in constructivism learning and developmental learning [147]. In this game, players can control a simulated artificial agent(a "baby robot") by pressing several buttons. At the start of playing, the agent designed without any prior knowledge about the effects of the commands and the environment. The player cannot directly observe the agent, and also the environment it interacts with . The only information received from the interactions and supported for the player is the feedback from the agent's interactions with the environment by pressing these commands. At the same time, the player needs to learn the function of different command and the structure of the environment from patterns in the stream of interaction feedback traces. The learning process is analogous to how infants engage in

early-state developmental learning [147].

To play the game, players need to press the commands at the bottom of the Screen. The new interactions will automatically generate at the bottom of the trace as illustrated in Figure 3.8. The traces is scrolled one step upward with a numerical value display on the right and also in digits as a bar-graph(green means positive while red means negative) which represents how much the agent "likes" or "dislikes" in this interaction [147]. The score on the top left is the sum of the values of the last interactions and the level on the top right is the current agent's play-level. Levels are organized in groups with regard to the kind of research questions they illustrate and the teams that study them. The level will be completed when the score reaches 10 and the player can move the next or to the previous level(if unlocked) by sliding the robot to the left or to the right. The level numbers are generally in the form of *g.nn*, where *g* is the group number and *nn* is the level number in group g, The player can slide Level Screen horizontally to access other group of levels, or slide downwards to resume playing the current level. The player can press the level link at the top left to access the Level Screen.



a) The initial interaface b) The 3D interaface

Figure 3.8: The Little AI interface.

3.3 Conclusion

In this chapter, we provide the foundations behind this computational model and pave the way for the next chapter to describe the design, the structure and the implementation of CCA. The foundations are divided into two parts: the theoretical foundations and the implementation foundations. Since the central of the dissertation is that a model based on the constructivist paradigm to simulate the early mechanisms of infant's cognitive development, we start with introducing Piaget's theory of constructivism and with its basic ideas and concepts, which include the definition of schemes, the theory of stages in the infants' cognitive development, and the adaptation processes between theses stages. The theories of radical constructivism and radical interactionism provide viable ways to implement the constructivism into an autonomous agent. In addition, we introduce the embodied paradigm which suggests that the agent is not a passive observer of reality, but rather constructs a perception through its active interaction with the environment. The enactive paradigm as an approach originally emerged from embodied cognitive science

endows the autonomous agent with a self-motivation, which spurs the agent to construct regularities of interaction afforded by the environment.

In the part of implementation foundations, we explain the representation and operations of schemes in computer science. Specifically, with similarities between the characteristics between schemes and binary tree, we utilize a data structure of binary tree as a way to represent schemes that are produced in agent's interaction with the environment. We can thus realize different kinds of operations in schemes by operating with a binary tree. Two benchmarks of Small Loop Problem and Little AI are introduced for validating CCA's abilities in the flowing chapters. In the following chapter, these two foundations are merged to form the computational model that is the main contribution of this dissertation: the Constructivist Cognitive Architecture, or the CCA.

Chapter 4

The Constructivist Cognitive Architecture

Contents

The constructivist cognitive architecture (CCA) is such a learning system that simulates the early mechanisms of infants' cognitive development based on theories of enactive cognition and principles of constructivist epistemology. This chapter contains a detailed description of the design, the structure, and the implementation of CCA, which is one of the main contributions of this dissertation. Particularly, the design and the implementation of of CCA is based on the analysis of Georgeon et al. [44]'s Enactive Cognitive Architecture (ECA), which presents my understanding and re-implementation of it.

The structure of this chapter is as follows: the section 4.1 describes the design of CCA, which includes the interaction cycle between the agent and the environment, sensorimotor interaction, representation of schemes and self-motivation in CCA. The section 4.2 introduces the structure of CCA with detailed explanation of each modules involved and the workflow between them. Section 4.3 presents the implementation of CCA. The final section makes a conclusion of this chapter.

4.1 The CCA design

4.1.1 Interaction cycle between the agent and the environment

Cognitive architectures and computational machine learning models generally represent the interaction between the agent and the environment as a cycle, in which the agent alternatively receives input data from the environment and sends output data to the environment, this interaction cycle as depicted in Figure 4.1.

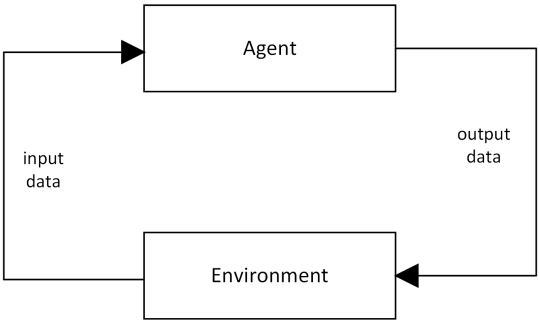
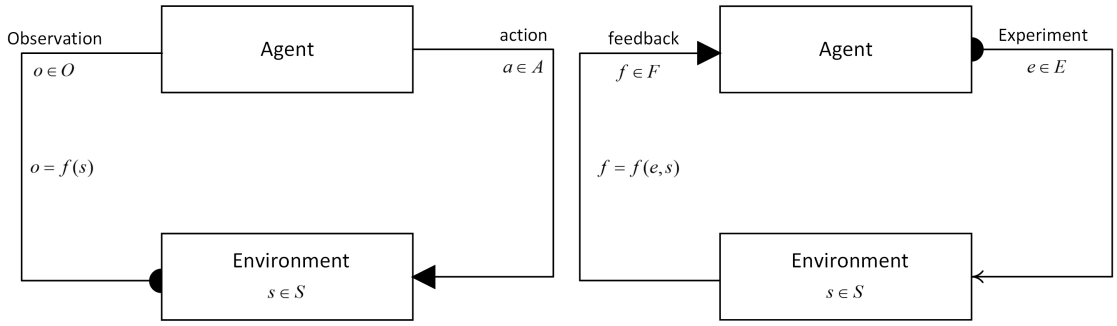


Figure 4.1: Interaction cycle between an agent (top) and an environment (bottom).

As presented in Figure 4.1, the model implies no particular commitment about the description of the input and output data. The lack of this explanation usually has a significant impact on the design of learning system, therefore forms different learning paradigm. From the traditional view, most models usually make an additional commitment that they arrange the input data (or called "observation" o in Figure 4.2(a)) so that it directly represents the environment' state, as if the input data partially made the environment's state accessible [148]. Furthermore, they agent's algorithm implementation follows this assumption. In the case of simulated environments, the observation $o = f(s)$ as a function of s, where s is the state of the environment (as shown in Figure 4.2(a)). While in cases of robots, input data presented by sensor data which is processed by the designer.

However, in the constructivist learning paradigm, agent is not a passive observer of the environment but the participant in the interaction between them that constructs a perception of reality through its active interaction [9]. This implies that the learning model should derive perception as a secondary construct resulting from experience of interaction, rather than considering the input data as the agent's perception. Fortunately, there exists



(a) The interaction cycle of traditional model. (b) The interaction cycle of embodied model.

Figure 4.2: The interaction cycle of embodied model (right) compared to the traditional model (left). In the traditional model, the cycle conceptually starts with observing the environment (half black circle on the environment) and ends by acting on the environment (black arrow on the environment). The embodied model cycle conceptually starts with the agent performing an experiment (half black circle on the agent), and ends by the agent receiving the result of the experiment (black arrow on the agent).

other possibilities proposed that input data for the agent is not the representation of the environment. A typical alternative is an *inversion of the perception-action cycle* [149, 148], in which the input data can represent the result of an experiment initiated by the agent. In the same environment's state, experiments may produce different results and the result itself thus not represent the environment's state, not even partially or with noise [148].

As shown in Figure 4.2(b), the agent's input data (called *feedback* $f \in F$) described as a feedback of an experiment $e \in E$ initiated by the agent, F as a set of all possible feedback and E as a set of all experiments. In the simulated environments, feedback f as a function $f(e, s)$ of the experiment e and the state s of the environment. Within a given state of the environment, the feedback may vary according to the experiment. In addition, in the cases where the environment is implemented without any states, agent could still interact with the environment and receive corresponding feedback, this situation will be explained in the next section. In the case of designing robots, sensor data can be taken as representing the result of an experiment initiated by the robot.

Meanwhile, another point that needs to pay attention is the starting point and the end point in the interaction cycle between the agent and the environment. Particularly, most works often ignore to make it explicit but it has a significant impact on the designs of agent's learning system, therefore forms the robot's sensors or simulated environment differently.

4.1.2 The sensorimotor interaction

As introduced in the last section, the inversion of the perception-action cycle ensures that agent's input data as a feedback of an experiment initiated by the agent [148]. In this section, I introduce the sensorimotor interaction in the CCA. The sensorimotor paradigm suggests that the input data should be taken in the association with output data, by combining both of them into a single entity called a *sensorimotor interaction*.

With the formalism introduced in last section, the sensorimotor interaction is defined as a tuple of $i = \langle e, f \rangle$ which consists of an experiment e with its corresponding feedback f from the environment, also it can be presented as $i = \langle experiment, feedback \rangle$. Noted that in this definition of the sensorimotor interaction, I utilize the term of *experiment* rather than the *action* (as previously mentioned in Section 1.2.1 of the Problem statement) is

in the reasons that the experiment provides an intended interaction that could flexibly represent a single primitive interaction or a complex interaction that consists of a series of primitive interactions. While for an action, it is an one-way process and not sufficient to express the interaction cycle (more detail explanation will be discussed in Section 6.1). In the theory of autonomous mental development, Piaget [118] coined the term *sensorimotor scheme* refers to a pattern of interaction between the agent and its environment. In the CCA, an interaction is as a pair of experiment and feedback $\langle e, f \rangle$ that represents a *primitive sensorimotor scheme*.

In this section and latter in this dissertation, the expression of "*to enact an interaction* $i : \langle e, f \rangle$" refers to performing the experiment e and receiving the feedback f that compose a given interaction i. Meanwhile, the expression of "*to intend to enact interaction* $i : \langle e, f \rangle$" means that the agent performs experiment e while expecting the corresponding feedback f. As a result of this intention of interaction $i = \langle e, f \rangle$, the agent may actually enact interaction $\langle e, f' \rangle$ if it receives the feedback of f' which is different from the expected feedback f. In this situation, the agent newly received enacted interaction $\langle e, f' \rangle$ as the feedback from the environment represents the differences of the interaction context, which reminds the agent that it may produce different feedback when performing an same experiment next time.

Overall, the sensorimotor paradigm allows designing self-motivated agents without modeling the world as a predefined set of states. Instead, the agent is left alone to construct its own model of the world through its individual experience of interaction. Since there is no predefined model of the world, the agent is not bound to a predefined set of goals. For example, it can discover/categorize new edible entities in the world.

4.1.3 Schemes in the CCA

The constructivist epistemology [120, 55] suggests that ·"sensorimotor patterns of interaction constitute the basic elements from which the agent constructs knowledge of the world". Accordingly, in the CCA, the agent's input data constitutes the feedback of its interactions iteratively construct a representation of the unknown environment [150]. In Piaget's view, the distinction between the inner self and the external world is not innate but is learned by the subject: "Intelligence (and therefore knowledge) begins not with the knowledge of the self, nor with the knowledge of things as such, but with the knowledge of their interaction; intelligence organizes the world while organizing itself by simultaneously considering the two poles of this interaction".

In the CCA, schemes are modeled in a form of $\langle interaction_1, interaction_2 \rangle$ (refers to Section 3.2), which expresses that the agent can enact $interaction_2$ in the context of $interaction_1$. Since the sensorimotor interactions are atomic elements in the constructivist learning model, I redesign the coupling of agent/environment to be more consistent with the cognitive coupling. As shown in Fgure 4.3, we replace the concepts of primitive experiments and feedback with the concepts of intended interactions and enacted interactions. This helps us address the constructivist learning problem of constructing knowledge of reality from regularities of sensorimotor interactions.

However, there exists several differences in terminology between the sensorimotor paradigm (as shown in Figure 4.2(b)) and the constructivist learning paradigm (as shown in Figure 4.3). (a) The agent's output data has been renamed i for intended interaction instead of e for experiment. (b) The agent's input data has been renamed e for enacted interaction instead of f for feedback. Besides the terminology, the only formal difference is that the agent's output data and the agent's input data now are all belong to the

same set of interactions I, instead of two different sets E of all experiments and F of all possible feedback. This means that the primitive data of the model is only of one kind: sensorimotor interactions. This is the key concept of constructivist learning paradigm and of the CCA that: using sensorimotor interactions as primitives of the model.

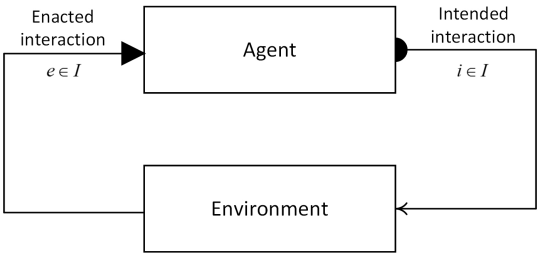


Figure 4.3: The interaction cycle of the constructivist learning paradigm.

From the perspective of constructivism, intended interaction i_t as it represents the sensorimotor scheme that the agent intends to enact, and constitutes the agent's output that is sent to the environment. While the enacted interaction e_t represents the sensorimotor scheme that the agent records as actually enacted, which constitutes the agent's input received from the environment. At the beginning of interaction cycle t, the agent chooses an intended interaction i_t to enact with the environment. The attempt to enact i_t may change the environment or not. At the end of cycle t, the agent receives the enacted interaction e_t. If the enacted interaction equals with the intended interaction ($e_t = i_t$), then the agent's attempted enaction is considered as a *success*. Otherwise, it is considered a *failure*.

As an example, the primitive interaction it may correspond to actively feeling (through touching) an object in front of the agent, involving both a movement and a sensory feedback. The tentative enaction of it may indeed result in feeling an object, in which case $e_t = i_t$. It may, however, result in feeling nothing if there is no object in front of the agent. In this case the enacted interaction e_t corresponds to a different interaction: moving while feeling nothing. The agent constructs knowledge about its environment and organizes its behavior through regularities observed in the sequences of enacted interactions.

4.1.4 Self-motivation in CCA

In CCA, there exists two forms of self-motivation: the motivation to be in control of one's own activity to by seeking to successfully enact interactions, and the motivation to enact interactions have predefined positive values and to avoid enacting interactions have predefined negative values. The former is called as *autotelic motivation*, and the latter is *interactional motivation* (refer to Section 3.1.4).

With the formally defined a *successful enaction* before, the *autotelic motivation* could be explained as the tendency to learn to successfully enact interactions. This tendency involves neither a maximizing reward function nor a specific goal to achieve, it is intrinsically encoded in the agent through discovering, recording, and re-enacting sequences of interactions that capture regularities in the coupling with the environment. To an external observer, the agent seems to enjoy being *in control of* its activity, which relates to the autotelic principle proposed by Steels [109] that "an agent could self-regulate its build-up skills and knowledge without the intervention of the designer".

Additionally, each sensorimotor interaction is associated with an "innate" value as a nature behavioral preference. For example, a positive value with interactions represents that an agent wants to move forward one step and it gets succeed, and a negative value

with interactions represents that the agent intends to move forward but it gets bumped with the wall in front of him. Interactional motivation is meant to "underdetermine the agent's behavior so that the agent can use neutral interactions to place itself in situations in which it can successfully enact positive interactions and to stay away from situations in which negative interactions cannot be avoided" [9].

4.2 CCA structure

In the previous sections, I explained the interaction cycle between the agent and the environment and specified the start point of this cycle, after then I introduced the sensorimotor paradigm with the definition of sensorimotor primitive interaction. With the redesigned coupling of agent/environment, the interactions are treated as primitives in the interaction cycle and introduce two forms of self-motivations: the autotelic motivation and the interactional motivation. In this section, I will introduce the structure of CCA, which includes the detailed explanation of each module involved and the workflow between them.

4.2.1 The CCA structure

As shown in Figure 4.4, the CCA is mainly composed of the following parts: the stream of enacted interactions timeline, the episodic memory of interaction experiences, the implementation of episodic memory with interactions, the hierarchical sequential system and the behavior selection mechanism.

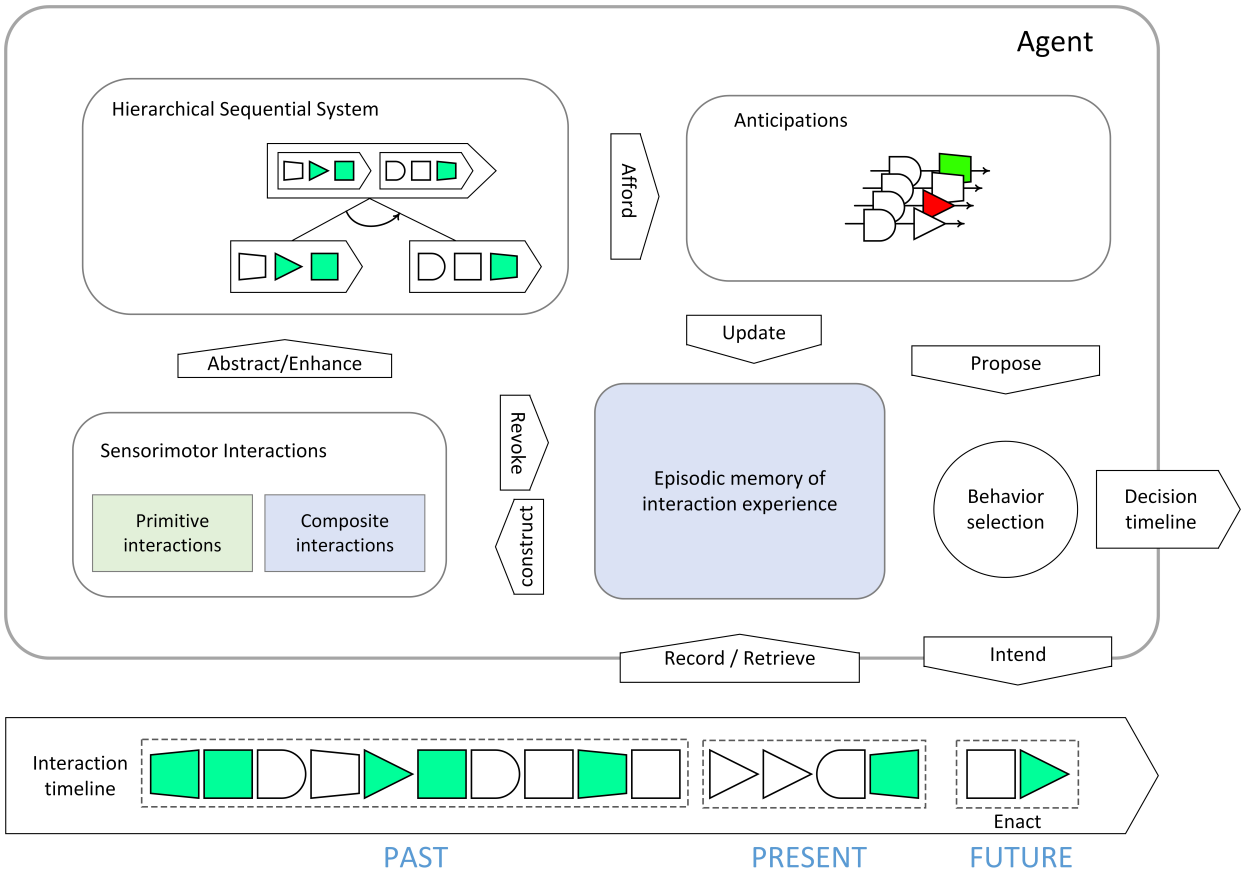


Figure 4.4: The Constructivist Cognitive Architecture (CCA).

At the bottom, the interaction timeline shows the stream of enacted interactions that occur over time as the agent interacting with the environment. Enacted interactions are represented by colored symbols indicate different interactions. The right green trapezoids represent touching right side and it's a wall, green squares represent touching front side and it's a wall, right half circles represent turning right, white square represent touching front side and it's empty, left white trapezoids represent touching left side and it's empty, green triangles represent moving forward and success, left green trapezoids represent touching left side and it's a wall, left half circles represent turning left.

As the agent receives enacted interactions in the interaction timeline, the episodic memory records all the agent's interaction history in the form of hierarchical sequential schemes. Each time the agent receives an enacted interaction, it will compare this enacted interaction with its corresponding intended interaction. According to our previous statement, if the intended interaction equals with the enacted interaction, this enaction of the intended interaction is considered as a*success* and this interaction experience will be marked as "stable" and reinforced in the episodic memory. The next time when the agent in the same context, the successful interaction experiment will be probably activated to enact again. However, if the intended interaction is different with the enacted interaction, the agent will retrieve this newly received enacted interaction in the episodic memory. If this enacted interaction already exist, then the agent reinforces this interaction experience in the memory and marks it as "unstable". Otherwise, if the enacted interaction isn't exist in the episodic memory, it will be appended in the memory and marked as "new learned". In essence, the agent's encoding of its interaction experience reflects the way the agent has learned to understand this experience. Then the agent gradually learn to pre-encode its experience for the future reuse.

One of the key ideas of CCA is to store the agent's interaction experience with the environment in an episodic memory, while also spurring the agent for enacting interactions not yet presented in memory. Being "not in memory" is the definition of *novelty* in CCA — seeking such enacted interaction means seeking the unfamiliar. Such a drive to seek the unfamiliar will lead the agent to experience new interactions (or schemes) for exploring new optimal behaviors in interacting with the environment. Agent in the interaction among the new enacted interaction, as the agent in interaction of study to the new scheme.

In the CCA, the episodic memory is implemented in the form of interaction. There are two different kinds of interactions exist in it: the one is primitive interaction and the other is composite interaction. Primitive interaction as a way to represent primitive sensorimotor scheme, the composite interaction is used to represent hierarchical sequential scheme. As mentioned in previous section, schemes are modeled as a couple of $\langle interaction_1, interaction_2 \rangle$, means that in the context of $interaction_1$, the agent can enact $interaction_2$. Composite interaction follows the same formalism but combines with two enacted interactions, the definition of composite interaction will be introduced in next section and explain how composite interactions represent schemes with episodic memory.

With agent's continuously interacting with the environment, new patterns of interaction and higher-level behaviors are gradually constructed, which enhances the hierarchical sequential system on the left top of CCA. Hierarchical Sequential System represents the mechanism of learning hierarchical regularities of interactions. It includes the rudimentary learning of regularities of interaction and the recursive learning of composite interactions. The rudimentary learning of regularities of interactions constructs the basic patterns of interaction with previous enacted interaction and current enacted interaction. While the recursive learning of composite interaction give a bottom-up way to learn higher-level patterns of interaction which are made of lower-level patterns of interactions (or rudimentary

composite interactions), which is bottom-up hierarchical sequential learning.

Behavior Selection mechanism balances the propositions made by the sequential system and by episodic memory, and then selects the next sequence of interactions to try to enact. For example, if an interaction of moving forward and succeeded is evoked in episodic memory in front of the agent, then the behavior selection mechanism may select this interaction as the next intended interaction to try to enact.

4.3 CCA implementation

4.3.1 Learning of of regularities: the composite interaction

Regularities of interaction (in short as regularities) are patterns of interaction that occur consistently. Specifically, regularities depend on the coupling between the agent and the environment. That is, they depend both on the structure of the environment, and on the possibilities of interaction that the agent has at its disposal.

At the beginning of interaction cycle at time t, the agent decides an intended interaction $i_t^i = \langle e_t, f_t \rangle$ and tries to enact with reference with the reactive part of the environment. Enacting i_t^i means the agent performs the experiment e_t and receives a feedback f_t that compose a given interaction i_t^i (refer to the expression of "enact an interaction" at section 4.1.2) at time t. As a result, the agent receives the enacted interaction i_t^e and memorizes the two-step enacted interaction sequence $c_t = \langle i_{t-1}^e, i_t^e \rangle$ as a tuple of $\langle contextInteraction, enactedInteraction \rangle$ made by the previously enacted interaction i_{t-1}^e of i_t^e. The sequence of these two enacted interaction $\langle i_{t-1}^e, i_t^e \rangle$ called a *composite interaction*, as the pattern of structured behaviors corresponds to the *assimilation* process in constructivism. The interaction i_{t-1}^e is called c_t's *pre-interaction*, noted as $pre(\langle i_{t-1}^e, i_t^e \rangle)$, and i_t^e is called c_t's *post-interaction* and is noted as $post(\langle i_{t-1}^e, i_t^e \rangle)$. The tuple of composite interaction expresses that in the context of i_{t-1}^e, the agent learns to recognize its interactive situation in terms of affordances related to its own prior experience, then enacts the proposed enacted interaction in the future to verify its assumptions. For the enacted interaction i_{t+1}^t at interaction cycle $t+1$, the composite interaction $c_t : \langle i_{t-1}^e, i_t^e \rangle$ also called i_{t+1}^t's *super-interaction*. From now on, low-level interactions $i = \langle e, f \rangle$ will be called *primitive interactions* to differentiate them from composite interactions.

As interaction continuing, more complex and higher-level composite interactions will be emerged with combinations of different kinds of pre-interactions and post-interactions. To better reflect the closeness of pre-interaction and post-interaction in composite interactions, each composite interaction associated with a *weight* (initialized as "1") and it will be incremented when the same composite interaction has learned again (coincide with the *accommodation* process in constructivism). Moreover, composite interaction as a complex type of interaction, its *valence* is the sum of its pre-interaction's and post-interaction's *valence*, which meaning that enacting a sequence of interaction has the same motivational valence as it enacting all its elements successively.

4.3.2 Selection mechanism

Note that the enacted interaction may activate more than one composite interaction, each of activated composite interactions could propose its post-interaction's experiment, these experiments may be the same, may be different, or partially the same. An *anticipation* is created for each activated composite interaction and associates with their post-interaction's experiment.

For making a decision among multiple anticipations, a notion of *proclivity* is introduced as a way to reflect the agent's *intrinsic satisfaction* for enacting each anticipation. The *proclivity* value comes from the *weight* of activated composite interaction multiplied by the *valence* of its post-interaction. As a result, the anticipations that are the most likely to result in the experiment that have the highest valence receives the highest proclivity. A selection mechanism is designed to sort the list of anticipations by decreasing proclivity value of their proposed interactions. Then, the agent takes the fist anticipation, which has the highest proclivity in this anticipations' list.

4.3.3 The enaction of intended interaction

As introduced in the last section, each anticipation corresponds to an experiment associated with a proclivity value for performing this experiment. The proclivity is computed on the basis of the possible interactions that may actually be enacted as an effect of performing this experiment, as far as the system can tell from its experience. The selection of anticipations is implemented by a decision mechanism that sorts the list of anticipations by decreasing their proclivities. Then the experiment of the first anticipation will be select to perform.

If the chosen experiment corresponds to a primitive interaction, the agent directly enact this primitive interaction and returns the enacted primitive interaction. However, if the experiment corresponds to a composite interaction, the agent needs to recursively enact the intended composite interaction all the way down to a sequence of primitive interactions, and return the enacted interaction according to the structure of the intended composite interaction.

As shown in Figure 4.5, at the beginning of decision cycle t (dashed loop in Figure 4.5), the agent's decision-making mechanism chooses the intended composite interaction i^i_{ct} from the set C_t of composite interactions known at time t. The enaction of i^i_{ct} consists of trying to enact the n primitive interactions $i^i_{p1}, \ldots, i^i_{pn}$ that constitute i^i_{ct} one after another (solid loops). If the enaction of i^e_{pk} fails ($i^i_{pk} \neq i^e_{pk}$) then the enaction of i^i_{ct} is interrupted. The decision making mechanism then receives the actually enacted composite interaction i^e_{ct} corresponding to the sequence $\langle i^e_{p1}..., i^e_{pk}\rangle$. From the perspective of the decision making mechanism, i^e_{ct} thus seems to be enacted as a single interaction in a virtual "environment known by the agent at time t" (dashed-line box). Because the primitive loop (plain line) and the decisional loop (dashed line) use the same entities (interactions), the learning mechanism that applies to the primitive loop can apply in the same way to the decisional loop, which allows recursive learning of increasingly complex composite interactions.

4.3.4 Learning of structured behaviors.

As introduced in the last section, the agent memorizes the two-step enacted interaction sequence, $c_t = \langle i^e_{t-1}, i^e_t\rangle$, i^e_{t-1} as c_t's pre-interaction and i^e_t as c_t's post-interaction. At the beginning of the interaction, the pre-interaction and post-interaction that make up the composite interaction are both primitive interactions. However, as the interaction continues, the agent gradually learns the patterns of interaction, therefore the pre-interaction and post-interaction that make up the composite interaction will gradually become composite interaction.

The system learns the composite interaction $\langle i^e_{ct-1}, i^e_{ct}\rangle$ made of the sequence of the previous enacted composite interaction i^e_{ct-1} and the last enacted composite interaction i^e_{ct}. Additionally, the system learns the composite interaction $\langle i^e_{ct-2}, \langle i^e_{ct-1}, i^e_{ct}\rangle\rangle$, which

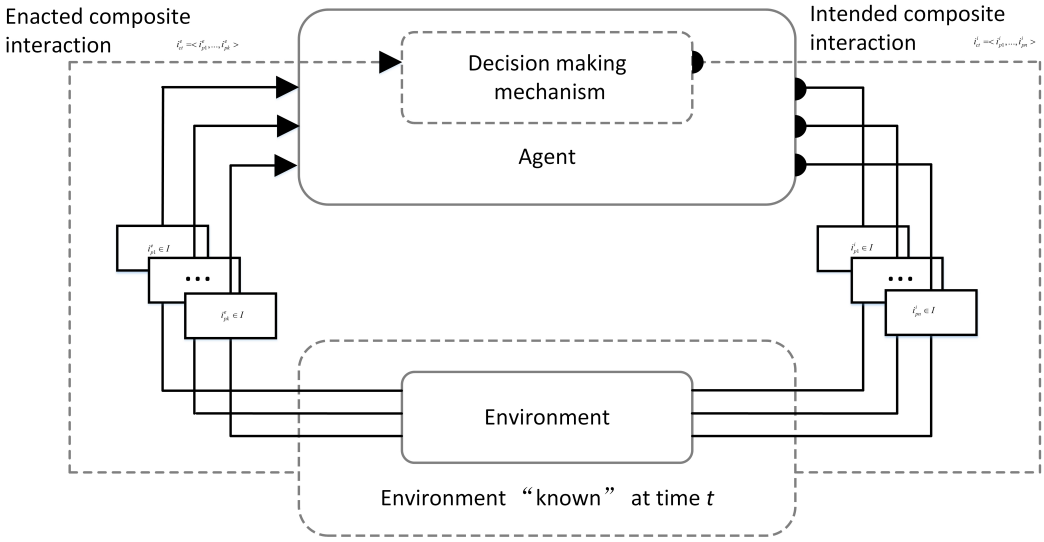


Figure 4.5: (adapted from [44]) The decision making mechanism and the enaction mechanism of intended interaction, particularly the intended composite interaction. At the beginning of each decision cycle t, the behavior selection mechanism decides a most likely intended composite interaction $i_{ct} = \langle i_{p1}, \ldots, i_{pn} \rangle, (i_{pj} \in I, 1 \leq j \leq n)$ from activated composite interactions whose context interaction belongs to the interactional context. The tentative enaction of i_{ct} refers to the enaction of a sequence of n primitive interactions from i_{p1} to i_{pn}. At the end of the decision cycle t, the agent receives the enacted composite interaction $e_{ct} = \langle e_{p1}, \ldots e_{pk} \rangle$, $1 \leq k \leq n$, $(e_{ph} \in I, 1 \leq h \leq k)$, corresponding to the enactions of multiple primitive interactions in the intended composite interaction.

represents that if i^e_{ct-2} is enacted again, $\langle i^e_{ct-2}, \langle i^e_{ct-1}, i^e_{ct} \rangle \rangle$ will be re-activated and will propose to enact its post-interaction $\langle i^e_{ct-1}, i^e_{ct} \rangle$. The system has thus learned to re-enact composite interaction $\langle i^e_{ct-1}, i^e_{ct} \rangle$ as a sequence. Moreover, the higher-level composite interaction $\langle \langle i^e_{ct-2}, i^e_{ct-1} \rangle, i^e_{ct} \rangle$ is also learned so that it can be re-activated in the context when $\langle i^e_{ct-2}, i^e_{ct-1} \rangle$ is enacted again, and propose its post-interaction i^e_{ct} for enacting.

In addition, once a new composite interaction i^i_{ct} is added to the set J_t of known interactions at time t, a new abstract experiment e^a_t is created to the set E_t of known experiments at time t, and its intended interaction is initialized as this newly learned composite interaction i^i_{ct}. In addition, the composite interaction's experiment is set as this newly created experiment e^a_t. Abstract experiments are called *abstract* because the environment cannot process them directly. The environment is only programmed to interpret a predefined set of experiments that now called *concrete*.

To perform an abstract experiment e^a_t, the agent must perform a series of concrete interactions and check their feedback. That is, the agent must try to enact the composite interaction i^i_{ct} from which the abstract experiment e^a_t was constructed. If the agent chooses experiment e^a_t, then the system tries to enact i^i_{ct}. If this tentative enaction fails and results in the enacted composite interaction $i^e_{ct} \in J_{t+1}$, a composite interaction i^c_{ct} is thus formed as $\langle i^e_{c(t-1)}, i^e_{ct} \rangle$, the $i^e_{c(t-1)}$ is composite interaction i^c_{ct}'s pre-interaction. If this composite interaction i^c_{ct} hasn't learned before, then the system creates a new the abstract experiment e^a_{t+1} and set its intended interaction as i^c_{ct}, the i^c_{ct}'experiment is set as this newly created experiment e^a_{t+1}. As a result of this learning mechanism, each composite interactions can have two forms: the sequential form $\langle pre{-}interaction, post{-}interaction \rangle$ and the abstract form $\langle experiment, feedback \rangle$. In order to differentiate these two forms, the abstract experiments and feedback in initial caps are separated by the "|" symbol: $\langle experiment \mid feedback \rangle$.

4.3.5 Episodic memory, "surprise", and "novelty"

As described in section 4.2, the CCA provides episodic memory for recording all agent's interaction history and in a hierarchical sequential form. The *novelty* is defined as "not in the memory". Once the agent receives an enacted interaction, new composite interactions will be learned or reinforced in conditions that the composite interactions have previously learned in the episodic memory. The newly learned composite interaction as the "novelty" in the memory. As mentioned above, the current enacted interaction acts as the context to retrieve interactions from the memory and activate composite interactions that their pre-interaction matches with this enacted interaction for proposing anticipations.

Furthermore, for each enacted interaction, the episodic memory also records it with its intended interaction. When the agent receives an enacted interaction, it will compare this enacted interaction with its corresponding intended interaction. With our previous statement, if the enacted interaction equals with the intended interaction, this enaction of intended interaction is considered as a *success*. If the enacted interaction is different with the intended interaction, therefore "surprise" comes up.

The agent needs to retrieve all enacted interactions with this intended interaction from the memory and check whether the newly received enacted interaction has appeared before. If it did appear, the "surprise-level" of this enacted interaction will be decrease to indicate that the agent is less surprised of this enaction. However, if the enacted interaction does not appear in the history of enacting this intended interaction, the "surprise-level" will be increased based on the differences between the intended interaction and the enacted interaction, particularly, the number of primitive interactions in the intended interaction subtracts the number of primitive interactions in the enacted interaction that equal with the ones in the intended interaction.

As interaction continues, the agent gradually learns the regularities of interaction afforded by the environment and build perception of the environment to explain such regularities. The construction of composite interactions becomes stable from the rapid development at the beginning of interaction. Similarly, novelties and surprises will disappear with the process of agent constructing the perception of the environment.

4.4 Conclusion

In this chapter, I introduced the design, the structure, and the implementation of the proposed Constructivist Cognitive Architecture (CCA). This is a contribution of my dissertation that was done in close collaboration with Olivier Georgeon.

In the first section, I start with the interaction cycle with between the agent and the environment. In particular, the Experiment-Result Cycle (ERC) ensures the input data of the agent is not present the states of the environment and I emphasize the importance of the start point and the end point in the agent/environment interaction cycle. Furthermore, I gave the definition of sensorimotor interaction and the representation schemes in CCA, which play as the basic blocks in this computational model. In the CCA, there exits two different forms of self-motivation, the autotelic motivation and the interactional motivation, the former spurs the agent to successfully enact interactions, and the latter implements the agent's natural preference to enact interactions have predefined positive values and avoid interactions have predefined negative values.

In the second section, I present the structure of CCA, which contains five main parts: (a) the stream of enacted interactions timeline, (b) the episodic memory of interaction experiences, (c) the implementation of episodic memory with interactions, (d) the hierar-

chical sequential system and (e) the behavior selection mechanism. After then, I explained each part involved and the workflow between them.

In the third section, I described the implementation of each part in CCA according to the workflow, which includes the rudimentary learning of regularities of interaction, selection mechanism, recursive enacting intended interaction, recursive learning of composite interaction and the episodic memory in the CCA.

Above all, the constructivist cognitive architecture (CCA) as the way towards simulating the learning mechanism of infants' early-stage cognitive development based on theories of enactive cognition, intrinsic motivation, and constructivist epistemology. Different with traditional cognitive architecture, the CCA neither initially endows the agent with the prior knowledge of its environment, nor supplies it with knowledge during its learning process. If it did, the agent's knowledge about the environment will not be grounded in the agent's interaction experience and motivations (it would not be the agent's knowledge, but the designer's knowledge). Instead, the CCA proposes a way for the agent to autonomously encode the interactional experiences and reuse behavioral patterns based on the agent's self-motivation as inborn proclivities that drive the agent in a proactive way. Following these drives, the agent autonomously learns regularities afforded by the environment, and hierarchical sequences of behaviors adapted to these regularities.

Chapter 5

Causality reconstruction with CCA

Contents

The previous chapter introduced the design, the structure, and the implementation of CCA, a learning system that allows the agent to encode the interactional experiences and reuse behavioral patterns based on the agent's self-motivation as inborn proclivities that drive the agent in a proactive way. Particularly, one of the most basic and the most critical parts in CCA is that the agent has abilities to learn regularities of interaction afforded by the environment and construct causality from interaction experiences.

The study of acquisition of causal perception is an important and challenging area in cognitive development. This chapter demonstrates CCA's ability to discover and learn regularities of interaction in its stream of experience and construct causal perception between phenomena whose hypothetical presence in the environment explains these regularities. The agent starts with looking for interactions that can be repeated several times in a row (or called *persistent interactions*) and obtains higher-level causal relations between these interactions. Results show that the agent develops a rudimentary form of causality, along with active perception as it learns to master the sensorimotor contingencies afforded by its coupling with the environment [150]. In the following chapters, the CCA will be applied to a self-motivated agent for bottom-up hierarchical sequential learning (Chapter 6) and applied to an autonomous robot in diverse simulations (Chapter 7).

5.1 Causal Perception

The capability of understanding causal relations is essential for sense-making and interacting with the environment [151]. Particularly in infants who shown by behavioral psychology studies that discover the underlying causal mechanisms from their interaction with the world [152], and also their causality knowledge in turn facilitates subsequent learning process. [153, 154, 155]. Accordingly, for a self-motivated agent, the causal acquisition allows the it learns to predict the effects of interactions and affect goal-directed change on the physical environment.

5.1.1 Causal Perception in Adults

The study of causal perception was famously inaugurated by Michotte [158] on exploring casual perception in adults with experiment which called *launching events* (as shown in Figure 5.1 of a schematic of these events). The experiment is in the scene that on billiard ball strikes another stationary ball, resulting in the launching of the stationary ball and the halting of the moving ball (or called the "launching event"). Michotte found that adults presented with a simple direct launching event would describe the event as causal [85].

By using the launching event, Michotte could affect a subjects' likeliness of perceiving causality [85]. One situation is that we can alter the event by introducing a gap between the two balls to keep objects ever touch. Also can alter the temporal component by introducing a delay between the moment of contact and launching as events presented in Figure 5.1.

5.1.2 Causal Perception in Infants

Since the study of Michotte on causal perception in adults, researchers have combined these launching events with habituation techniques to demonstrate the presence of causal perception in infants. One study by Leslie [159] demonstrated that infants' ability to

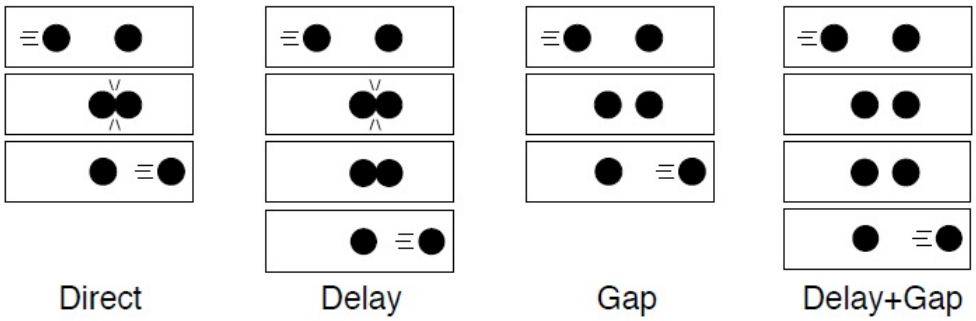


Figure 5.1: Launching events (Cited from *Chaput 2004* [85] Ch4, Section 1, p20). Four different launching events used by [158] to study causal perception in adults.

discriminate between different launching events, especially their responses were on the causality of the event.

However, recent studies tend to refute these claims and have new results demonstrated that infant develops his own causal perception and follows a hierarchical style. Particularly, Cohen and Amsel [160] performed a similar experiment and they found that different with older infants, younger ones would respond to the introduction or removal or either a delay or gap, regardless of how this change impacted the causality of the event. These result indicated a developmental shift in causal understanding that progresses from a component view to a higher-level causal view. The development of causality is just one instance of a more general part-to-whole progression that can be see in a variety of cognitive developmental domains.

Nevertheless, a related but alternative view from Georgeon et al. [161] introduce design principles that implement the construction of phenomenal knowledge and causal perception in an unknown noumenal reality. Particularly, the view conveyed an idea that discovers and learns regularities in its stream of experience and to construct knowledge about phenomena whose hypothetical presence in the environment explains these regularities. Thus the causality emerges between phenomena.

5.2 Modeling Causal Acquisition with CCA

The studies described above suggest a simple progression stages of causal understanding from low-level component views to a higher-level causal view. These stages do not occur sequentially and independent of one another. Rather, the component view is used to build the causal view. Following with these theories, I have designed and implemented a causality reconstruction model with CCA as presented below. In particular, the design principles are introduced for a self-motivated agent followed by constructivist paradigm to learn regularities of the environment and construct causality consists of learning feedback from interactions with the environment and organizing its behaviors to fulfill a form of intentionality defined specific-tasks independently with the environment. This work suggests a new approach to cognitive modeling that focuses on the agent's internal stream of experience. In addition, for facilitating to explain this model, an interaction scenario is provided to demonstrate the causal acquisition with CCA in the following section.

5.2.1 Interaction scenarios

As an example, an interaction scenario is presented in which the agent can experience 11 different sensorimotor interactions (in short as interactions, as shown in Figure 5.2) represented by shapes of left trapezoid/left-half circle/square/right trapezoid/right-half circle with colors of red/green. Specifically, the green-square interaction has a positive valence (+10), the rest interactions have a zero valence. These valences are arbitrary and remain constant during the system's existence. That is, the agent as if it tends to enjoy experiencing green-square interaction, and was indifferent to experience other interactions. For the sake of demonstration, agent is initialized ignorant of the meaning of interactions, which is consistent with the constructivist paradigm that the agent isn't endowed with any prior knowledge of the environment, nor supplies it with knowledge during its learning process.

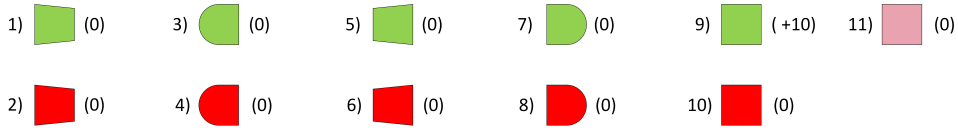


Figure 5.2: Eleven different sensorimotor interactions.

At any given moment, the agent needs to make a decision to select an interaction for enacting. Assuming that the environment affords regularities in the way it generates the actual interaction i_t^e. These regularities may depend both on the current intended interaction i_t^i and on previous interactions. Moreover, to anticipate actual interactions, the agent must construct causality that represents the regularities afforded to it by the environment.

5.2.2 Principles of the learning

In the casual acquisition model, there exists an implementation of the mechanism that between two kinds of regularities: *immediate regularities* and *sequential regularities*. Immediate regularities usually present the correlation between enacted interactions with intended interactions, regarding to its interaction experience. However, sequential regularities mostly depend on the interaction history. A sequential regularity is usually performed in the form like that: in a given interactions have experienced, a corresponding interaction could be experienced next. With this implementation of the sequential regularities, it is complied with the definition of *schemes* as a tuple of $\langle interaction_1, interaction_2 \rangle$ that has mentioned in the Section 3.2, which describes the context in terms of interaction $interaction_1$ that the agent learns to "see" its world in terms of affordances [129] $interaction_2$ related to its own interaction experience. As shown in Figure 5.3, it lists partial 12 first-order (two-step) regularities afforded by the environment (the complete two-step regularities afforded by the environment refer to Appendix A.2).

With these two kinds of regularities, the agent is expected to discover the patterns between interactions and exploit them to construct the causality, which allows the agent to obtain more experiences that enact interactions have positive valence and avoid interactions that have zero valence. As a result, the agent could develop the causal acquisition model that is depicted in Figure 5.4. As illustrated in Figure 5.4, the agent is expected to learn to use square interactions to inform subsequent behavior. If an intended square interaction yields a green-square, the system will subsequently experience green-square interaction with the reason that it has a positive valence (Figure 5.4-a, exploiting Regularity 7 of Figure 5.3). If the intended square interaction yields a red-square, the system

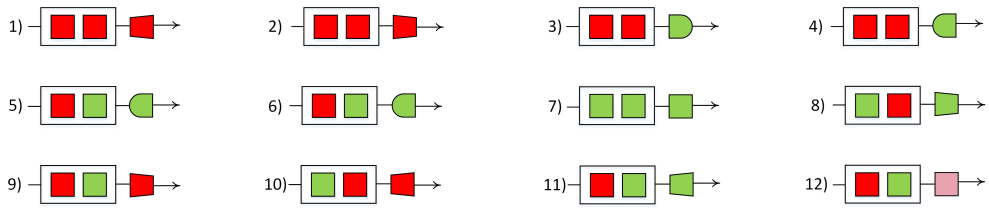


Figure 5.3: Partial 12 two-step regularities afforded by the environment, consisting of a pre-interaction followed by a post-interaction. Regularity 1: in the context when two red squares have just been experienced, it is likely that the interaction of right-red-trapezoid can be experienced again; i.e., if the agent intends to experience any of the two-red interaction again, then the agent will be more likely experience right-red-trapezoid than other interactions. Similar with regularity 5, in the left-red-square and the right-green-square context, the agent is likely to experience a green-left-half-circle interaction (immediate regularities prevail over sequential regularities).

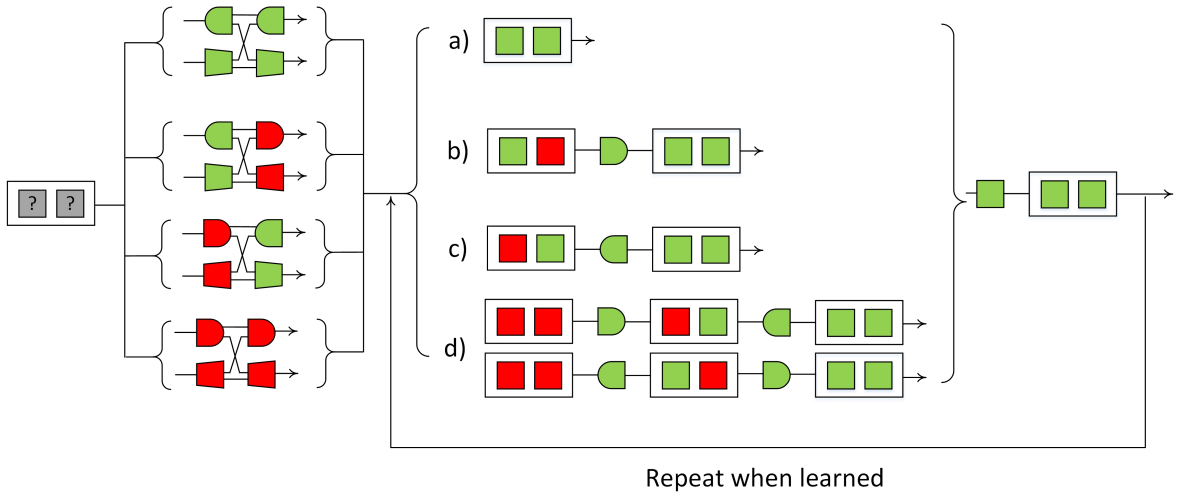


Figure 5.4: Patterns of interaction experience that expect the causal acquisition model to develop. The curly brackets represent a conditional scheme that results in sequence a), b), c) or d) depending on the current state of the environment. Once the agent has learned these schemes, it applies them indefinitely.

should subsequently experience green-left-circle interaction so that it can then experience green-square interaction (Figure 5.4-d, exploiting Regularities 3 and then 4 of Figure 5.3). After this, I expect the agent to intend a square interaction again and to repeat the same conditional choice again (Figure 5.4-b or 5.4-c again).

5.2.3 The algorithm of the causality reconstruction

In this section, I introduce the algorithm of the causality reconstruction that allows the agent to explicitly interpret the patterns of behavior as illustrated in Figure 5.4. I divide the learning process into two parts: the node construction stage and the arc construction stage. In the node construction stage, the algorithm tries to learn which stable phenomena exist. In the arc construction stage, the algorithm learns how to transition from one phenomena to another.

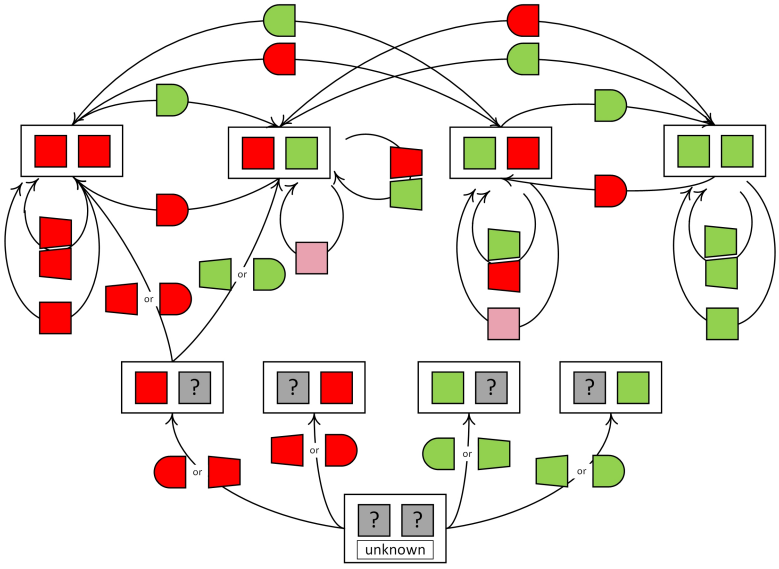


Figure 5.5: Partial representation of the Petri-net constructed by the algorithm. White rectangles represent nodes. Arcs with feeling experiences (trapezoids and squares) are self-loops that do not change the state. Arcs with swapping experiences (half-circles) cause transitions between belief states.

The node construction stage

To model the part-to-whole progression of causal acquisition, the agent is designed to start with looking for interactions that can be repeated several times in a row and then use these solid interactions to construct the causality. When the same interaction is enacted in a certain number (or call as the *excitement threshold* k) of times in a row, which confirms that this interaction is a solid experience, then the algorithm assumes that the interaction is persistent and creates a new node combines with this interaction (as shown in the left figure of Figure 5.6). Otherwise, this interaction is a sporadic experience.

Once the enaction of an interaction exceeds the excitement threshold, the agent holds a belief of it. Particularly, the algorithm implements a *BeliefState* class with an array of *triedNumber* for each interaction and a adaptive method *getLeastTriedInteraction()*. When a new node is created, a new BeliefState object associated with this node is instantiated. Its triedNumber attribute is used to count the number of trials of an interaction in this belief state. When the algorithm is in the "curious" mood, it uses the function of

getLeastTriedInteraction() to select the least tried interaction based on the *triedNumber* within the current BeliefState. In the initial phase of the interaction, the BeliefState is initialized as *unknown* and all interactions have not been tried before, the agent randomly selects an interaction begins the interaction.

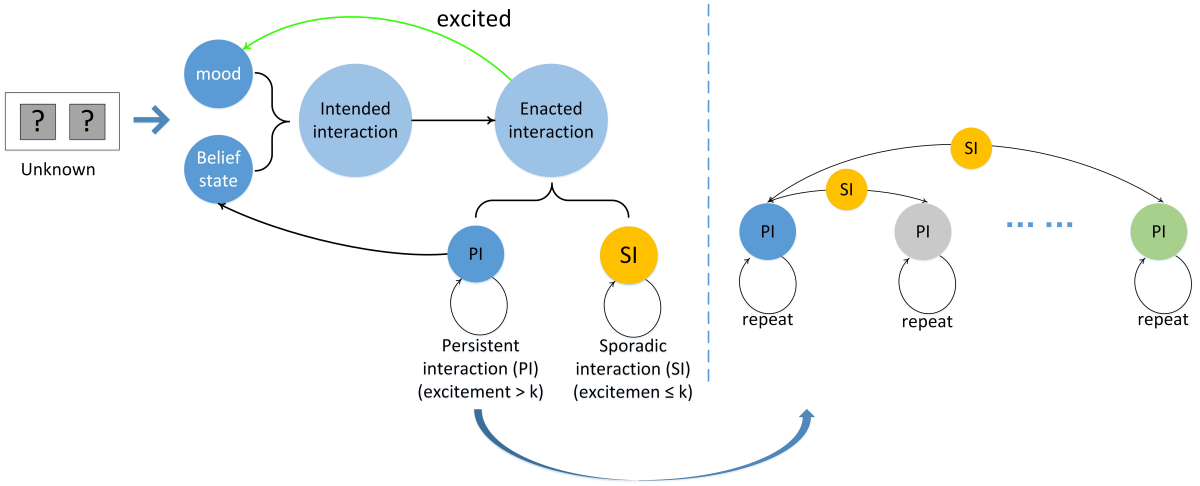


Figure 5.6: Modeling Causal Acquisition with CCA .

The arc construction stage

When the agent finds a candidate persistent interaction, it creates a self-loop based on this interaction, and a node attached to this self-loop. This node thus represents a hypothetical phenomenon that can be consistently observed through this interaction. Next, the agent learns arcs between nodes which represent the causality between these persistent interactions. When the agent learns a new stable BeliefState, then it goes into the arc construction stage (as shown in the right figure of Figure 5.6).

The arc construction combines with the feedback that the agent receives from its past interactions and the previous patterns in the stream of interaction traces. With the changeable environment, it needs to figure out the difference of the context between previous and post persistent interactions after the agent experiences a sporadic interaction. If the two persistent interactions are different, then an arc between these two persistent interactions will be created through this sporadic interaction. Otherwise, the agent continues try another sporadic interaction until all sporadic interactions have all been tested with all persistent interactions. When all interactions are known to the agent and no changes happen in the causal structure, the agent is in the "confident" mood. Overall, I expect the algorithm to learn the Petri-net as shown in Figure 5.5, which constitutes a valid representation of the structure of the coupling of agent/environment.

The algorithm

Algorithm 3 A proof-of-concept algorithm to infer phenomena from regularities of interaction afforded by the environment.

```
1: Initial:
2:     currentBeliefState ← "unknown", mood ← "curious";
3: loop
4:     if mood = "curious"
5:         intendedInteraction = getLeastTriedInteraction(currentBeliefState);
```

```
6:    if mood = "hedonist"
7:       intendedInteraction = intentionWithMaxValence(currentBeliefState)
8:    if mood = "confident"
9:       finish the knowledge construction and learning is done;
10:   if mood = "excited"
11:      intendedInteraction = enactedInteraction;
12:   enactedInteraction = ReactiveSubsystem(intendedInteraction)
13:   if enactedInteractionis"unknown"
14:      mood = "excited";
```

Alg 3 summarizes this algorithm at the highest level. The algorithm has four possible motivational moods: *curious*, *excited*, *hedonist*, and *confident*. At the beginning of the interaction, the current belief state is initialized as "unknown" and the agent is in a *curious* mood. Lines 04 and 05: if the agent is in a *curious* mood, it selects an interaction that is the least tried in the current BeliefState as the next intendedInteraction. Line 06 to Line 7: if the agent is in a *hedonist* mood, it will intend an interaction with the maximum valence in the current BeliefState. Line 08 and 09: if the agent is in a *confident* mood, then the algorithm has already finished the reconstruction of the whole Petri-net which represents the causality of interaction experiences, the learning process is thus done. Lines 10 and 11: if the agent is in an *excited* mood, then it intends to repeat the previously enacted interaction. Line 12: the algorithm provides the intended interaction to the subprogram that implements in the environment. This subprogram processes the action associated with the intended interaction and returns the enacted interaction, which may be same with the intended interaction or not. Lines 13 and 14: if the enacted interaction is "unknown", which means this interaction has neither been marked persistent nor sporadic, then the agent gets excited about this interaction and the mood becomes "excited".

Algorithm 4 Causality reconstruction.

```
1:  if mood = "excited"
2:     if intendedInteraction ≠ enactedInteraction
3:        intendedInteraction is sporadic
4:        currentBeliefState = knowledgeUpdating(intendedInteraction)
5:     else if excitement ≥ excitementThreshold
6:        enactedInteraction is persistent
7:        create new beliefState and added in the beliefStateList
8:        currentBeliefState = knowledgeUpdating(enactedInteraction)
9:     else
10:       excitement++
11: updateTriedNumberOfInteraction(intendedInteraction)
12: knowledgeUpdating(currentBeliefState)
13: if (all interactions have not been tried yet in the current BeliefState)
14:    mood = "curious"
15: if (all interactions have been tried and knowledge isn't updated)
16:    mood = "confident"
```

Lines 01 to 10: if the agent is in the *excited* mood and the intendedInteraction differs from the actually enactedInteraction, the intendedInteraction is marked as *sporadic*. Otherwise, the algorithm increments its excitement level. When the excitement reaches the preset threshold, this intendedInteraction is marked as *persistent*, then a new belief state is instantiated. Line 11: the current BeliefState updates its intendedInteraction's tried numbers. Line 12: learning regularities between persistent interactions and sporadic

interactions, and update the current belief state. Lines 13 and 14: if all interactions have not been experienced before, the mood becomes curious. Lines 15 and 16: in the situations that all interactions are known to the agent and there is no other change in the procedure of knowledge reconstruction, the agent is thus in the "confident" mood. This means that the algorithm has reconstructed a valid representation of the structure of the agent/environment coupling. The arc construction is thus finished.

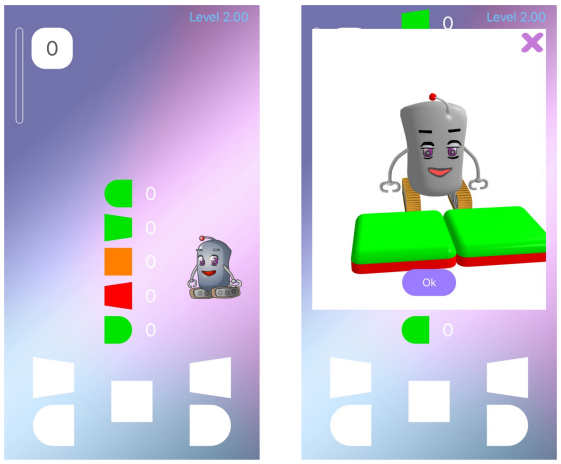
5.3 Experiment

Little AI is a pedagogical game as mentioned before in Section 3.2.2, in the experiment, I mainly focus on the Level2.00 of Little AI. In this level, the system initially provides five commands with same shape and same color (as shown in Figure 5.7(a)). The effects of these five commands and the structure of the environment are unknown to the agent and the player as well [161], so the players need to construct the knowledge gradually by their interactions with the agent and its environment. For the sake of demonstration, I intentionally place the reader in the same situation as the system: initially ignorant of the meaning of experiences.

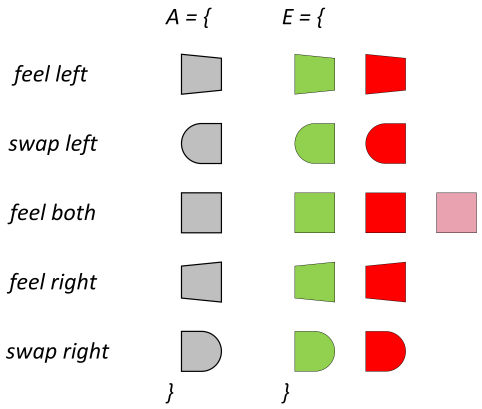
The self-motivated agent is designed to execute five actions affiliated with different numbers which are $A = \{feelleft = 1, swapleft = 2, feelboth = 3, feelright = 4, swapright = 5\}$. These numbers in A are just used to identify the actions in the later experimentation more conveniently. In a given time, the agent chooses an action amongst the set of five possible actions A, then it receives an interaction corresponded with this action and the state of situated environment. Actions of *feel left* and *feel right* touch the environment and report the state of the environment, they have two different results: true (marked 1) or false (marked 0), action *feel both* touch the left and the right at the same time and gives the situation of current environment, it thus has three different results: left and right both are false (0,0), left and right only one are true (0,1 or 1,0) and left and right both are true (1,1). Only actions of *swap left* can change the left state of the environment and *swap right* can change the state of right. Above all, the agent has eleven different kinds of interactions which are based on corresponding actions and the state of situated environment. When the agent experiences several interactions, then the environment will give the response result of the actions. In order to present these five actions more clearly, I prefer to use different icons and colors to identify five actions and eleven interactions respectively (as shown in Figure 5.7(b)). With different situations the agent faced with and the changeable environment the agent situated in, I use three colors (red means false, green means true and pink only used for the action of *feel both* means only one side is true) combined with five actions to illustrate these eleven interactions.

The experimentations are conducted following two main steps. Firstly, the agent selects the intended interaction which is the least tried in the current belief state and uses the persistent interactions to observe the following interactions. Secondly, once the agent confirms the intended interaction is persistent or sporadic, it then tries to construct the connections between this new learned interaction with previously confirmed persistent interactions and sporadic interactions.

When interactions are all known to the agent and there are nothing changes in the procedure of creating connections between all persistent and sporadic interactions, which represents that the agent has already learned regularities of Little AI and the causality reconstruction goes to stable. Then it is in the *confident* mood and then in the *hedonist* mood and trying to organize its behaviors to fulfill a form of intentionality defined in the environment. This work suggests a new approach to cognitive modeling that focuses on the



(a) Little AI interface.



(b) Five actions and eleven interactions.

Figure 5.7: The Little AI interface and five actions with their eleven interactions.

agent's internal stream of interaction. The experimentation helps clarify the distinction between the designer's goal and the agent's intentionality. while the agent remained unaware of the underlying structure of the environment, it learned to master sensorimotor contingencies.

Behavioral analysis of a rudimentary constructivist agent

Our experiment shows that the algorithm was able to construct the Petri-net in 350 interaction cycles (Figure 5.8) Our system's 350 interactions. Line 1: the intended interaction. Line 2: the enacted interaction. Line 3: current belief: unknown(grey pointed rectangle)/persistent interactions/ allKnown(green pointed rectangle). Line 4: mood: curious ("?"), excited (black bargraph), or hedonist (green circle). Step 1: the algorithm intends a red left trapezoid and obtains a same red left trapezoid. Since the red left trapezoid interaction is neither yet marked sporadic or persistent, the agent gets excited (black bar in Line 4). Step 2 to 5, the agent repeat the red left trapezoid and gets increasingly excited. Step 6: the agent reaches the excitement threshold; it marks the red left trapezoid is persistent, and creates a new belief state associated with this interaction (Line 3: the current belief becomes red left trapezoid; Line 4: the agent becomes curious to play with the newly created belief). Step 7: the agent tries red rectangle. Step 15: the agent tries red left half-circle and obtains green left half-circle. Step 16: green left circle-rectangle is learned to be sporadic since it failed on the second attempt. Then the agent enters the arc reconstruction stage, The agent will test all the effects of different sporadic interactions to persistent interactions consecutively. Step 17, the agent encounters again with the red left trapezoid, it knows this interaction that experienced before and recognizes this interaction is persistent, then the current belief associates with this persistent interaction. Step 77: similar, green right half-circle is sporadic. Step 350: The interactions are all marked to the agent and there are no more changes in the Petri-net, the agent's mood becomes confident (green circle) and in an all-known belief state(green pointed rectangle). Arrived at this step, the algorithm can use the constructed Petri-net to predict the consequences of interaction interactions.

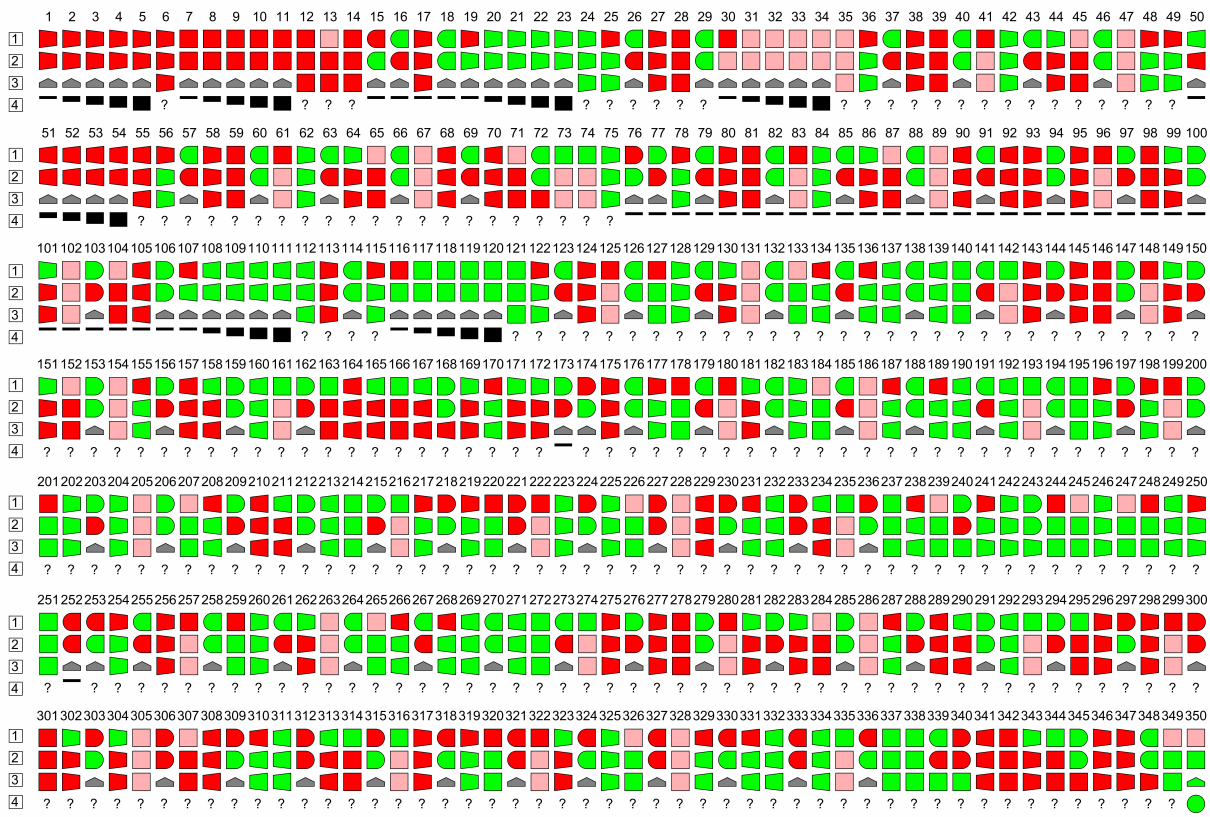


Figure 5.8: Trace of the first 350 interaction cycles in our experiment. Line 1: intended interactions.Line 2: enacted interactions. Line 3: belief states: unknown (grey triangle) / known state represented by its corresponding persistent interaction. Line 4: mood: curious (question mark), excited (increasing black bars), or confident (green circle).

5.4 Conclusion

In this chapter, I introduced the design principles with CCA to let an self-motivated agent discover and learn regularities of Little AI(in level2.00) in this stream of experience and construct causality that consists of learning feedback from interactions with the environment and organizing its behaviors to fulfill a form of specific-tasks intentionality defined independently with the environment. In the game of Little AI, the agent learns the meaning of their own actions and infer the structure of the environment simultaneously based on the patterns in the stream of interactions feedback traces.

An intriguing philosophical question is whether the learned model of the coupling between the agent and the environment constitutes the best possible model accessible to the agent. When transposed to humans and their knowledge, Kantian-like positions affirm that we can only know the world as we experience it. By contrast, those admitting a more (scientific) realist epistemology believe that knowledge can be pushed further, toward the world in itself. Returning to our discussion, this suggests the following question: would it be possible and interesting to design an agent that would try to infer theories of the world in itself (the complete implementation of the agent and of the environment)?

Technically, the question remains how the algorithm could construct a theory that involves a representation of its environment with tiles and their sides. This may be feasible if the agent presupposes the existence of space. The agent could try to construct a simpler model based on the assumption that states of the agent/environment coupling are caused by the presence of objects in some locations in space. Such simplification of knowledge will be necessary when moving on towards more complex tasks. In more complex tasks, the Petri net will be too large and complex. Explaining the regularities in terms of the presence of objects in some locations in space will provide a powerful means to deal with such complexity.

Chapter 6

Bottom-up hierarchical sequential learning in CCA

Contents

In the constructivist view, infant's cognitive development can be described as a bottom-up and a hierarchical processing according to which infants initially process simple perceptual units. These building blocks of simple units then are organized into a more complex, higher-level abstractions at the next level, which themselves in turn become integrated to yield more higher units at the third level, and so on. Furthermore, once infants have access to multiple levels, they will tend to process information at the highest possible level unless the system is somehow overloaded. In that case, learning will drop down to a lower level of processing and attempt to rebuild the system. In brief, the infant cognitive development follows a hierarchical and constructive paradigm.

In the previous chapters, I presented the structure of CCA and described the implementations of rudimentary learning of regularities of interaction and recursive learning of composite interactions. In this chapter, I introduce a Bottom-up hiErarchical sequential Learning model based on the CCA, which is also called BEL-CA, as a solution for an autonomous agent continuously constructing the knowledge of the environment and acquiring capabilities of self-adaptation and flexibility. In addition, for facilitating to observe the agent's learning process in interacting with the environment and the emergence of structured behaviors after each enaction of intended interaction, I have designed and developed an implementation of toolkit for agent autonomously generating and analyzing interaction (GAIT) at run-time, which will be introduced in the next chapter of methodology.

This chapter is structured as follows. The section 6.1 gives the definition of interaction in BEL-CA and with its valence allocation. Section 6.2 explains the hierarchical sequential learning process in CCA. Section 6.3 gives the formal model of bottom-up hierarchical sequential learning based on CCA, as well as the BEL-CA. In section 6.4, I introduce the algorithms for implementing each part in the BEL-CA. Section 6.5 describes the comparison with related work and Section 6.6 gives a conclusion of this chapter.

6.1 The interaction and its valence allocation

The definition of interaction in the BEL-CA is followed by the sensorimotor paradigm (refers to Section 4.1.2), which suggests that the input data should be taken in association with output data, rather disassociated. Also this complies with constructivist epistemology, which suggests that knowledge of reality is constructed from regularities observed in sensorimotor experience. As shown in Figure 4.3, agent's output data and input data are all interactions and belong to the same set of I, which represents the set of primitive interactions. The primitive interaction is defined as a tuple of an experiment e_t with its corresponding feedback f_t at time t, $i_t = \langle e_t, f_t \rangle$.

Additionally, each primitive interaction associated with an innate scalar valence v_t as a way to simulate agent's inborn behavioral preference and quantify the agent's "feeling" of each interaction experience. In a sense, such innate valences relate to fundamental constraints that involve the agent's risk aversion. Examples of such constraints allow the agent to move around and avoid being bumped with the wall. This innate valence, nonetheless, provides a reason why the agent should even learn to cope with the environment in the first place: the agent should be able to efficiently enact interactions that favor its moving activity and avoid interactions that have a risk of collisions with the wall.

The quantification of valences (or the valences allocation) for various primitive interactions can reflect the strength of the agent's desire for each primitive interaction. For example, if the agent decides to "move forward", there exits two possible situations afforded for the agent: moving forward successfully or bumping with the wall. Suppose

associating positive valence for the interaction of "move forward successfully", the agent will be satisfied with this interaction and continues enacting interactions that have experiment of "move forward" and expecting the feedback of "success". Otherwise, the agent enjoys receiving the feedback of "bumping". With the commonsense, positive valence is usually associated for interaction of "moving forward successfully" and negative valence for "bumping" interaction. With these predefined primitive interactions, the agent will find ways to elegantly organize them to experience more interactions for "moving forward" with the feedback of "success" and reduce the interactions with "bumping". For primitive interactions with negative valences, except the "bumping" interaction, the valence of experiencing interaction of "touch front side and it is empty" is better than "touch front side and it is a wall ", then the agent is expected to construct the connection between the interaction of "feeling front side and it is empty" first and making decisions for enacting interaction of "moving forward successfully".

The valence allocation for primitive interactions is an important topic in constructivist learning. The optimal allocation strategy could apparently accelerate the learning process and improve the agent's performance in interacting with the environment. Otherwise, it could slow down the agent's interaction and even interfere with the learning process. Currently, there exists several approaches [162, 163, 164, 165, 166, 167] dedicate to finding the optimal valence among various combination possibilities. However, these approaches have certain limitations, and usually obtain the partial optimal valence. In this dissertation, valence is mainly allocated based on experiences, this issue of valence allocation will be detailly discussed in the Section 8.2.3 of Chapter 8.

With the explanation from Section 4.1.2 that enacting an interaction $i_t : \langle e_t, f_t \rangle$ means that the agent intends to perform an experiment e_t and receives feedback f_t that composites a given interaction at step t. Note that the experiment may contain a single primitive interaction or a series of primitive interactions, thus the performance of this experiment will be different. In the case of a single primitive interaction, the agent enacts it directly and receives the corresponding feedback. While in cases where the experiment contains an interaction that is composed by a multiple of primitive interactions, the agent needs to flat this interaction down to a series of primitive interactions and enact these primitive interactions recursively. The feedback of enacting this type of interaction comes from the results of enacting each primitive interaction and re-form them with this interaction's hierarchical structure. Furthermore, the agent intends an interaction i_t which expresses that it performs an experiment e_t while expecting its corresponding feedback f_t at step t. Within diverse situations, this "intention" could be that the agent actually enacts interaction $\langle e_t, f_t^{'} \rangle$ if it receives feedback $f_t^{'}$ instead of f_t.

From the perspective of constructivism, intended interaction as it represents the sensorimotor schema that the agent intends to enact, and constitutes the agent's output that is sent to the environment. While the enacted interaction represents the sensorimotor schema that the agent records as actually enacted, which constitutes the agent's input received from the environment. If the enacted interaction equals the intended interaction, then the attempted enaction of intended interaction is considered a *success*, otherwise *failure*.

6.2 The hierarchical sequential learning process in CCA

Following the structure of CCA in Section 4.2, the hierarchical sequential learning system in CCA as shown in Figure 6.1. At the beginning of each interaction cycle t, the agent de-

cides an intended interaction $i_t^i = \langle e_t, f_t \rangle$ and tries to enact with reference to the reactive part of the environment. After the enacted interaction i_t^e has received, new composite interactions are constructed or reinforced with their pre-interaction belonging to the context and their post-interaction i_t^e, forming the set of learned or reinforced interaction c_t to be included in C_{t+1}, which represents the set of composite interaction at time $t+1$. For supporting affordance of more complicated interaction situations in the future. The set of composite c_t is defined as $c_t = \{\langle i_{t-1}^e, i_t^e \rangle, \langle i_{t-2}^e, \langle i_{t-1}^e, i_t^e \rangle \rangle, \langle \langle i_{t-2}^e, i_{t-1}^e \rangle, i_t^e \rangle\}$, $C_{t+1} = C_t \cup c_t$.

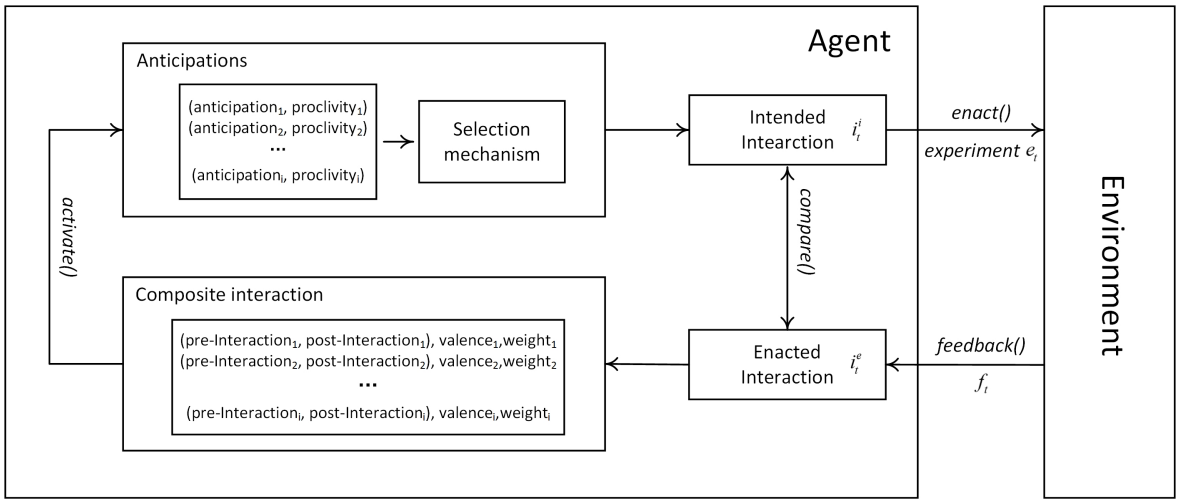


Figure 6.1: The hierarchical sequential learning system in CCA.

Thus the set of composite interactions known by the agent at time t is C_t and the set $J_t = I \cup C_t$ indicates all interactions known to the agent at time t. For the next step interaction, the enacted interaction i_t^e activates all previously learned composite interactions C_t as it matches with their pre-interaction, then the agent forms the *activated composited interaction* set $A_t = \{a_i | a_i \in C_t, pre(a_i) = i_t^e\}$. For example, if $i_t^e = a$ and the composite interaction $\langle a, b \rangle$ has been learned before time t, then the composite interaction $\langle a, b \rangle$ is activated, like it's recalled from the memory. Activated composite interactions propose their post-interaction as anticipations for the next round, in this case: the interaction of b. The agent's decision making comes from these anticipations.

For each post-interactions, *anticipation* is created with a scalar value *proclivity* $p_i \in \Re$ which is computed from the weight w_{a_i} of the activated composite interaction a_i multiplied by the valence of the proposed post-interaction $v(post(a_i))$. The proclivity value as a way to reflect the regularity of the interaction based on its probability of occurrence and the motivations of the agent.

$$p_i = w_{a_i} \times v(post(a_i)) \tag{6.1}$$

As a result, the anticipation which is the most likely to result in the primitive interaction that has the highest valence receives the highest proclivity, and that have the biggest proclivity are the most likely to be enacted in the next round.

6.3 The BEL-CA

6.3.1 The structure of BEL-CA

Based on the CCA, I propose a Bottom-up hiErarchical sequential Learning model, which is also called BEL-CA, as a solution for an autonomous agent continuously learning representations of the environment and acquiring capabilities of self-adaptation and flexibility.

As shown in Figure 6.2, it consists of nine parts: (a) pairs of intended interaction with its enacted interaction at the bottom, (b) sequential learning with composite interactions, (c) abstraction of experiments in the memory, (d) activation of composite interactions by enacted interactions, (e) anticipations from activated composite interactions and afford its propositions , (f) mapping from propositions to experiments, (g) selection mechanism, (h) enacting intended interactions and (i) the construction of higher-level behavioral sequences.

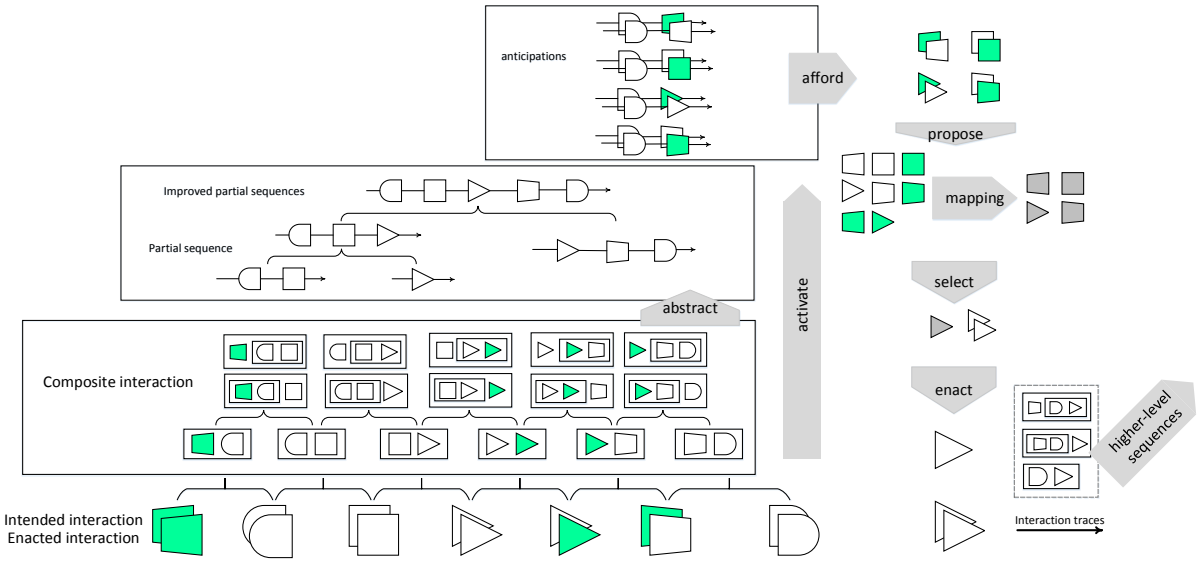


Figure 6.2: The hierarchical sequential learning model of BEL-CA with CCA

Following the timeline at the bottom of CCA, the timeline in the BEL-CA represents the stream of selected intended interactions with its enacted interactions that occur over time as agent interacts with the environment. Particularly, colored symbols in the timeline indicate different interactions, right green trapezoid represents that the agent touches right side and finds it's a wall, left half-circle represents turn left, white rectangle represents touching front and the result is empty, white triangle represents moving forward one step and it succeeds, green triangle represents moving forward but bumping with the wall, left blue trapezoid represents touch the left side and find it's a wall, white left trapezoid represents touch the left side but it's empty, right half circle represents turn right.

Meanwhile, same with CCA, composite interactions in the BEL-CA represent the sequential learning system, which are composed by enacted interaction and with its previous enacted interactions. Each time the agent receives an enacted interaction, it records this enacted interaction with previous enacted interaction as a composite interaction. Moreover, the agent recalls the two-step previous enacted interaction with new learned composite interaction to form a newly third level composite interaction. Besides, the agent combines previous super-interaction with enacted interaction to form another third level composite interaction. For example, the agent receives an enacted interaction i^e_t at time t, a composite interaction is formed as $\langle i^e_{t-1}, i^e_t \rangle$. Combined with two-step previous enacted interaction and previous super-interaction, another two composite interaction are built as: $\langle i^e_{t-2}, \langle i^e_{t-1}, i^e_t \rangle \rangle$, $\langle \langle i^e_{t-2}, i^e_{t-1} \rangle, i^e_t \rangle$. As new composite interactions have learned, new abstract experiments are created as described in Section 4.3.4.

The enacted interaction as a way to recognize the current context is used to propose anticipations from the episodic memory for the following interactions. Especially, composite interactions are retrieved from the memory and activated in cases where their pre-interaction matches with the enacted interaction. All activated composite interactions propose their post-interaction's experiment as anticipations with a proclivity value which

is calculated by activated composite interactions' weight and their post-interaction's valence.

Based on anticipations from CCA, the BEL-CA mainly focuses on the classification of anticipations that start with same primitive experiment. Based on this, anticipations are classified into different groups and identified by the experiment of first primitive ineraction in their intended interaction. The proclivity value of different groups is the sum of their anticipations' proclivities. Thus the selection mechanism is mainly divided into two steps: firstly, sorting these groups with their proclivities and choose one group has the biggest proclivity; secondly, sorting the anticipations in the chosen group with their proclivities and select one anticipation with the biggest proclivity. The intended interaction of the selected anticipation's experiment will be enacted for the next interaction.

The enaction of an intended interaction depends on which type it is (primitive interaction or composite interaction) and how much weight it has (compared with the given threshold). If the intended interaction is a primitive interaction, then directly enact it and receive the corresponding primitive enacted interaction. However, if the intended interaction is a composite interaction, the enaction of this intended interaction should consider its weight with the threshold. If the weight is greater than the threshold, called the *regularity sensibility threshold*, then it is effectively proposed for being enacted as a whole. If its weight is lower than or equal to the threshold and if its proclivity value is positive, then the intended interaction is not proposed but its proposition is propagated to its pre-interaction. In essence, this mechanism makes sure that higher-level schemes are sufficiently rehearsed before being enacted as a whole. During each rehearsal, higher-level schemes are reinforced, which tends to pass the reinforcement from one hierarchy level to the next. A lower regularity sensibility threshold results in a faster adoption of potentially less satisfying higher-level schemes. A higher regularity sensibility threshold results in a slower adoption of potentially more satisfying higher-level schemes.

This selection mechanism shows that proclivity values do not operate as a reward but rather as inward drives that either push the agent toward or restrain him from enacting certain behavior. Moreover, the reinforcement does not operate as a form of reward propagation but as a mechanism for experience counting. By driving the agent's behavior, this selection mechanism also shapes the agent's experience and consequently the agent's learning. The agent does not learn all that can be learned in the environment—which would be overwhelming—but only what its motivations make it experience. In addition, this mechanism guarantees that higher-level regularities will only be constructed upon lower-level regularities that have been effectively tested.

6.4 Algorithms

Following with the hierarchical sequential learning system in CCA and nine parts in the learning process of BEL-CA, a high-level overview of BEL-CA is presented as in Algorithm 5 below.

Algorithm 5 Bottom-up hierarchical sequential learning with CCA.

1: Initial:
2: the set of experiments $E = \{e_1, e_2, e_3, \ldots, e_n\}$;
3: the set of valences $V = \{v_1, v_2, v_3, \ldots, v_m\}$;
4: the set of Feedback $Fe = [f_1, f_2]$;
5: E.resetAbstract();
6: the set of primitive interactions I = addOrGetPrimitiveInteraction(E, Fe, V);

```
    the set of default intended Interactions De = defaultInteractions(I);
    E.setIntendedInteraction(De);
    the set composite Interaction C_0 = φ,
    the set of all interactions J_0 = I.

while interactions continues do
    anticipations = anticipate();
    selectedAnticipation = selectInteraction(anticipations);
    intendedInteraction = selectedAnticipation.intendedInteraction;
    enactedInteraction = enact(intendedInteraction);
    if intendedInteraction = enactedInteraction then
        enaction of intended interaction is a success;
    else
        enaction of intended interaction is a failure;
    end if
    c_t = learnCompositeInteraction(intendedInteraction, enactedInteraction);
    if c_t ∉ C_t then
        Initial c_t's weight as 1;
        Add c_t in C_t;
    else
        Increase c_t's weight;
        Reinforce c_t in C_t;
    end if
end while
```

6.4.1 Initialization

At the beginning of interaction with the environment, the agent is pre-defined with a set of primitive interactions, which is composed of innate experiments and the possible feedback of the interaction with the environment. These primitive interactions enable the agent to interact with the environment without any prior knowledge about the environment, nor desired goals for the agent to achieve. Noted that each innate experiment contains a type of action, and also pre-defined with a primitive intended-interaction which is composed of this experiment and a possible interaction feedback. In addition, each primitive interaction is associated with a scalar *valence* pre-defined with positive or negative values that represent agent's nature reference. For example, if the agent decides an experiment of "moving forward", there exists two different feedback: moving successfully or bumping with the wall. In order to distinguish these two kinds of feedback, a positive valence could be used as "1" to perform moving successfully and "-1" for bumping with the wall.

Since there exists no enacted interaction and interaction experiences at this moment, the set of composite interaction C_0 is empty as performed with ϕ, the set of all interactions J_0 equals with the set of primitive interactions I. With no composite interactions are activated for proposing anticipations, the agent randomly selects one from default anticipations which are formed by innate experiments, and intends the default intended-interaction of its experiment to start the interaction process.

6.4.2 Context construction

On each decision cycle, an intended interaction is selected through an activation mechanism which is triggered by the context. At the end of decision cycle t, the agent records

or reinforces the composite interaction composed by the pre-interaction which belongs to C_{t-1} and post-interaction is the actually enacted interaction i_t^e. Thus, the newly learned or reinforced composite interactions set is formed as: $L_t^c = \{\langle i_{t-1}^e, i_t^e \rangle | i_{t-1}^e \in C_{t-1}\}$. Each learned or reinforced composite interaction thus represents that when then context i_{t-1}^e is enacted, i_t^e will subsequently enacted in the future. The set of composite interactions known by the agent at decision cycle t thus become $C_t = C_{t-1} \cup L_t^c$. The weight of the newly learned composite interactions is initialized as 1. For reinforcing previously constructed composite interactions, their weight is automatically increased by 1.

Before making the decision for the next interaction, the agent saves the current context B_t as the set of interactions were enacted at the end of decision cycle t and that are sufficiently reinforced.

$$B_t = i_{t-1}^e \cup post(i_{t-1}^e) \cup i_{t-1}^{super} \tag{6.2}$$

Equation 6.2 implies that B_t includes interactions of various lengths that have been enacted at the end of decision cycle t. As interaction continues, the agent selects an intended interaction to enact that capture sequential regularities of interaction afforded by the environment, B_t tends to represent the agent's situation in terms of the sequences of interactions that are the most representations of the situation at time t.

6.4.3 Activation of composite interactions and the construction of anticipations

Given with the context B_t, the agent retrieves the memory and searches for composite interactions whose pre-interaction is contained in the current context B_t. The qualified composite interactions will be activated and form the set of activated interactions A_t which is defined as $A_t = \{a_i | a_i \in C_t, pre(a_i) \in B_t\}$ (as shown in Algorithm 6). After then, the activated interactions in A_t propose their post-interaction's experiment to form weighted anticipations. The weight of each anticipation equals to the weight of the proposing activated interaction and its proclivity value will be calculated from the weight of activated interaction a_i and the valence $v(post(a_i))$ of its post interaction. In this way, the set of anticipations is formed as: $AN_t = \{anti_i \in AN_t | a_i \in A_t, proclivity_{anti_i} = w_{a_i} \times v(post(a_i))\}$.

Algorithm 6 Activation of composite interactions and the construction of anticipations

```
1: Initial:
2:    the set of primitive interactions I;
3:    the set of context interaction B_t;
4:    the set of activated composite interaction A_t = [ ];
5:    the set of default anticipations Default_t = [ ];
6:    the set of anticipations AN_t = [ ];
7:
8: for (each i_t ∈ I)
9:    anti^i_default = createAnticipation(i_t.experiment, 0);
10:   anti^i_default.anticipationsList = [ ];
11:   add anti^i_default in Default_t;
12:
13: for (each c_i ∈ C_t)
14:   if(pre(c_i) ∈ B_t )
15:      add c_i in A_t;
```

16:
17: for (each $a_i \in A_t$)
18: $proclivity_{anti_i} = w_{a_i} \times v(post(a_i))$;
19: $anti_i = createAnticipation(post(a_i).experiment, proclivity_{anti_i})$;
20: $anti_i.intendedInteraction = post(a_i)$;
21: $anti_i.weight = weight(a_i)$;
22: if ($anti_i \notin AN_t$)
23: Add $anti_i$ in AN_t;
24: else
25: $anti_{old} = AN_t.getAlreadyExistAnticipaiton()$
26: $anti_{old}.addProclivity(proclivity_{anti_i})$
27: $anti_{old}.addWeight(anti_i.weight)$
28:
29: for (each $anti^i_{default} \in Default_t$)
30: for (each $anti_i \in AN_t$)
31: $firstPrimitiveIntearction$ = getFirstPrimitiveInteraction($anti_i$);
32: if ($firstPrimitiveIntearction.experiment = anti^i_{default}.experiment$)
33: Add $anti_i$ in $anti^i_{default}.anticipationsList$;
34: $proclivity_{anti^i_{default}} += proclivity_{anti_i}$;

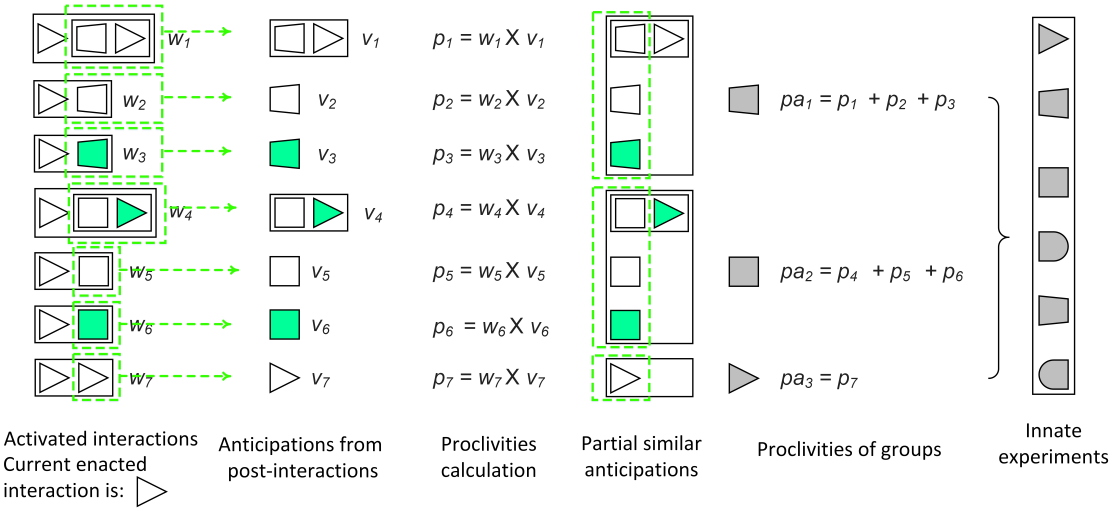


Figure 6.3: Mapping anticipations to experiments. With the previous enacted interaction is a white triangle, seven composite interactions are activated for proposing their post-interaction. After calculating each anticipation's proclivity, I classify these seven anticipations into three different groups according to their anchor, which is the innate experiment of the first primitive interaction in the intended-interaction of each anticipation's experiment. The proclivity value of each group is the sum of anticipation's proclivity which is classified in it.

Meanwhile, with anticipations obtained from the activation mechanism, I classify them into different groups according to their *anchor*, which is the innate experiment of the first primitive interaction in the intended-interaction of each anticipation's experiment. For anticipations sharing with the same anchor, or called *partial similar anticipations (PSAs)*, they will be classified into the same group and form an *anticipationsList* identified by this anchor (as shown in Figure 6.3. The proclivity of each group is the sum of the proclivity of all anticipations in it. The process of the classification of anticipations and the proclivity calculation of each group is presented in following equations:

$$group_i = \{anti_i \in group_i | anti_i \in AN_t, firstExp(anti_i) = anchor(group_i)\} \tag{6.3}$$

$$proclivity_{anti^i_{default}} = \sum_{i=1}^{n} w_{a_i} \times v(post(a_i)) \tag{6.4}$$

The $proclivity_{anti^i_{default}}$ is the proclivity of the group i, n refers to the number of anticipations in this group, w_{a_i} is the weight of activated composite interactions a_i and the $v(post(a_i))$ is the valence of a_i's post-interaction.

6.4.4 Selection mechanism

The intended interaction i^i_t is selected through the function of *selectInteraction()* (as shown in Figure 6.4). First, sorting all groups with their proclivities in descending order and selecting one with the biggest proclivity. Second, once again sort all anticipations in the selected group in descending order according to their respective proclivity and select the anticipation with the biggest proclivity. Therefore, the intended-interaction of the selected anticipation's experiment is selected to enact for the next interaction.

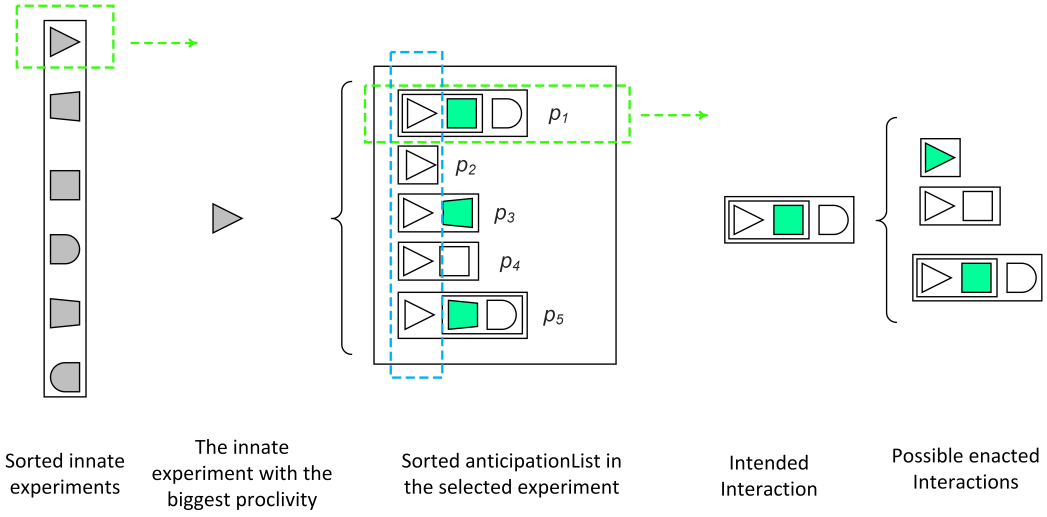


Figure 6.4: Selection intended interaction. The selection mechanism sorts all groups of innate experiments by their proclivity and selects one group has the biggest proclivity. The selected group sorts its anticipationList according to their proclivity and selects the anticipation with the biggest proclivity. The intended-interaction of the selected anticipation's experiment is selected to enact for the next interaction.

The moment when the intended interaction is selected, a little heuristic applies (as shown in Algorithm 7). If the intended interaction is a composite interaction, the intention of this composite interaction is subject to its anticipation's weight w_{a_i} and the threshold $d \in \Re$ (or called as the *regularities sensibility threshold*). The parameter d is the threshold which encodes the limitation of intending the composite interaction as a whole. If the weight of the proposing anticipation is greater than d and its proclivity value is positive, then the agent will effectively enact all primitive interactions within this composite interaction according to the hierarchical sequential structure recursively. If the weight is lower or equal with d but its proclivity is positive, the agent will try to enact the pre-interaction of this composite interaction. Otherwise, the agent just needs to enact the

first primitive interaction of this composite interaction. In essence, this intention mechanism of composite interactions ensures that higher-level schemes are sufficiently rehearsed before being enacted as a whole. During this rehearsal, higher-level schemes are enough reinforced and gradually establish believes from one hierarchy level to the next. If the sequence of intended interaction corresponds to the regularity of interaction, then it is possible that the sequence of this intended interaction can be enacted again. Therefore, the agent can thus base its choice of the next interaction on this anticipation.

As shown in Algorithm 7, in the selection mechanism, proclivity values as an inward drives the agent towards or restrains him from enacting certain behaviors. Moreover, this mechanism shapes agent's experience and consequent learning, which guarantees the higher-level regularities are constructed based on lower-level regularities that have been effectively proven to be trustworthy. In addition, with different levels in the intentions of composite interactions, the selection mechanism allows the agent to capture different levels of regularities. Over time, the schemes are established by the most robust level of regularity afforded by the environment and that best fulfill the agent's satisfaction become popular. For example, in cases where the agent tries to move forward, the decision-making will not rest upon the last primitive step but rather exploit regularities that propose a sequence of experiences.

Algorithm 7 The selection mechanism of intended interaction.

```
1: Initial:
2:    default anticipations ← Default_t;
3:    intendedInteraction ← NULL;
4:    threshold ← d;
5:
6: function selectInteraction(Default_t):
7:    sortedDefaultAnticipations = sort(Default_t);
8:    selectedDefaultAnticipation = sortedDefaultAnticipations.get(0);
9:    if (selectedDefaultAnticipation is primitive)
10:      experiment = selectedDefaultAnticipation.experiment;
11:      intendedInteraction = experiment.intendedInteraction;
12:      return intendedInteraction;
13:   else
14:      detailAnticipationList = selectedDefaultAnticipation.anticipationsList;
15:      sortedDetailAnticipationList = sort(detailAnticipationList);
16:      selectedAnti = sortedDetailAnticipationList.get(0);
17:      experiment = selectedAnti.experiment;
18:      intendedInteraction = experiment.intendedInteraction;
19:      if (intendedInteraction is primitive)
20:         return intendedInteraction;
21:      else
22:         if (selectedAnti.weight >threshold and selectedAnti.proclivity >0 )
23:            return intendedInteraction;
24:         else if (selectedAnti.weight ≤ threshold and selectedAnti.proclivity >0)
25:            intendedInteraction = intendedInteraction.preInteraction;
26:            return intendedInteraction;
27:         else
28:            intendedInteraction = firstPrimitive(intendedInteraction);
29:            return intendedInteraction;
30:
31: function firstPrimitive(intendedInteraction):
```

```
32:   firstPrimitiveInteraction = intendedInteraction;
33:   while (firstPrimitiveInteraction is not primitive)
34:     firstPrimitiveInteraction = firstPrimitiveInteraction.preInteraction;
35:   return firstPrimitiveInteraction
```

6.4.5 The enaction of intended interaction

Once the intended interaction is proposed, the agent tries to enact it. To do so, the enaction mechanism recursively flats the intended interaction down to a sequence of primitive interactions according to its hierarchical structure (as shown in Algorithm 8 Following with this sequence, the agent enacts each primitive interaction and compares the corresponding enacted primitive interaction with it. If the obtained enacted primitive interaction matches with the intended primitive interaction, then the enaction procedure follows the same way to the next intended primitive interaction until the end of the sequence, unless a special situation that the enacted primitive interaction and the intended primitive interaction are different. In this special situation, the enaction progress is interrupted and the actually enacted interaction is constructed based on the part of the hierarchy that has been actually enacted until the interruption. The enacted interaction will be used to reflect the situation where the current intention ends up for the next interaction, both at the hierarchy level of intended interaction and the hierarchy level where the enaction fallback.

Algorithm 8 Enaction intended interaction

```
1: Initial:
2:   Given with intendedInteraction i_t^i;
3:   Nodes = [];
4:   nodeStack = [];
5:   enactedInteraction = NULL;
6:
7: if (intendedInteraction is primitive)
8:   enactedInteraction = enact(intendedInteraction);
9: else
10:  add intendedInteraction in nodeStack;
11:  while (nodeStack is not null)
12:    topInteraction = nodeStack.pop();
13:    if (topNode is not primitive )
14:      nodeStack.push(topInteraction.postInteraction);
15:      nodeStack.push(topInteraction.preInteraction);
16:    else
17:      newNode = createNode(topInteraction);
18:      add newNode in Nodes;
19:  node = getLeftNode(Nodes);
20:  while (the root node is not visited)
21:    intendedPrimitiveInteraction = node.getInteraction;
22:    enactedPrimitiveInteraction = enact(intendedPrimitiveInteraction);
23:    if (intendedPrimitiveInteraction = enactedPrimitiveInteraction )
24:      previousReInteraction = recordWithStructure(enactedPrimitiveInteraction);
25:      node = getNearestRightNode(node);
26:    else
27:      previousReInteraction = recordWithStructure(enactedPrimitiveInteraction);
28:      break;
```

29: enactedInteraction = previousReInteraction;
30: learnCompositeInteraction (intendedInteraction, enactedInteraction);

In effect, the enaction mechanism associated with the selection mechanism favors the agent to adopt unsatisfying experience to form various hierarchical interactions, which allows the agent to enact more satisfying or elegant interactions in the future. Therefore, the agent not only simply reflexes interactions toward the highest immediate proclivity, but also learns from unsatisfying interactions for more satisfying interactions.

With the enacted interaction, composite interactions are learned or reinforced with their pre-interaction belongs to the context and their post-interaction i_t^e, forming the set of learned or reinforced composite interaction c_t to be included in C_{t+1}, for supporting affordance of more complicated interaction situations in the future. The set c_t is presented as $c_t = \{\langle i_{t-1}^e, i_t^e\rangle, \langle i_{t-2}^e, \langle i_{t-1}^e, i_t^e\rangle\rangle, \langle\langle i_{t-2}^e, i_{t-1}^e\rangle, i_t^e\rangle\}$, $C_{t+1} = C_t \cup c_t$. A new context B_{t+1} is constructed based on i_t^e and $post(i_t^e)$.

6.5 Comparison with related work

The first needs to note that the proposed model of BEL-CA differs from studies in robotics for autonomously learning the correlation between actuators and sensors. Mostly, these studies conform to the "perception-cognition-action" (as mentioned in Section 3.1.3) loop with a mechanism that implements the sensor's signal as the temporary perception of the environment. Instead, I follow the sensorimotor paradigm that associates the input data with the output data. The agent is initialized with a set of primitive interactions that associate the agent's innate experiments with their corresponding possible feedback. The agent thus does not need to learn the relation between its actions and its inputs, but is self-motivated to generate proper intended interactions based on its perception of the environment for establishing various higher-level schemes. Nevertheless, with implementing more complex perceptual behaviors, a hybrid model needs to be considered in the future.

Meanwhile, the model uses episodic memory as a way to record agent's all interactions which include intended interactions with their corresponding enacted interactions, and all obtained composite interactions in each decision cycle. Inspired from mechanisms of learning through experience, especially the Trace-Based Reasoning [168], the agent retrieves the memory and activates composite interactions whose pre-interaction is contained in the context B_t. Particularly, the Trace-Based Reasoning usually follows an approach of knowledge-representation that requires the designer to provide the process of encoding episodes, and drive the process of reusing episodes [9]. However, the model differs from the Trace-Based Reasoning's knowledge-representation approach in that the I neither initially endow the agent with the prior knowledge of its environment, nor supply it with this knowledge during the learning process. Instead, an alternative mechanism has been proposed for the agent autonomously encoding and reusing episodes based on the agent's self-motivation. With this mechanism, it provides a possible way to tackle with issues that related to non-Markovian sequence modeling which includes automatically delimitates episodes, organizes episodes in a hierarchy, and encodes context in a way that supports the appropriate reuse of episodes.

In addition, during each decision-making, this work differs with the reinforcement learning paradigm. As mentioned above, in the selection mechanism, proclivity values do not operate as a reward in reinforcement learning, but as an inward that drives the agent towards or restrains him from enacting certain behaviors. In particular, the intention mechanism ensures the higher-level composite interactions are sufficiently rehearsed

before being enacted as a whole. During this rehearsal, the mechanism shapes agent's experience and consequent learning, which guarantees the higher-level regularities are constructed based on lower-level regularities that have been effectively proven to be trustworthy. Thus our work is also differs from approaches of statistical hierarchical temporal learning [169] but relates to pragmatic epistemology [170] and evolutionist epistemology [171] that suggest knowledge evolves on the basis of usage and to constructivist epistemology [172], which suggest that knowledge selection is driven by the subject's intrinsic motivation. Moreover, the reinforcements for composite interactions in our model are not to obtain the maximum accumulated reward, but as a mechanism to strengthen the regularities afforded by the environment and best fulfill the agent's satisfaction.

Currently, there exists two major approaches to implement intrinsic motivation for artificial agents (and robots). The one that consists of implementing motivation as behavioral rules that directly represent either emotions [173] or drives [174]. The another implements intrinsic motivation as curiosity [17, 16] and searches for novelty [88]. In this work, the proposed approach endows the agent with two forms of self-motivation: successfully enacting sequences of interactions (autotelic motivation), and preferably enacting interactions that have predefined positive values (interactional motivation). Nonetheless, the three approaches could be complementary in the case of higher-level cognition.

6.6 Conclusion

In this chapter, I present a Bottom-up hiErarchical sequential Learning model with CCA, which is also called BEL-CA, as a model for an autonomous agent continuously constructs the knowledge of the environment and acquiring capabilities of self-adaptation and flexibility. Following with three processes of assimilation, accommodation and equilibrium in constructivsm, the agent autonomously organizes schemas it learned from interaction into hierarchically structured behaviors, which let the agent gradually understand the meaning of divers experiments and infer the structure of the environment simultaneously based on the patterns in the stream of interactions feedback traces.

Different with traditional approaches, the proposed model neither initially endows the agent with the prior knowledge of its environment, nor supplies it with knowledge during its learning process. Instead, the agent autonomously encodes the interactional experiences and reuses behavioral patterns based on its intrinsic motivation. In BEL-CA, the agent is driven by two forms of self-motivation: successfully enacting sequences of interactions (autotelic motivation), and preferably enacting interactions that have predefined positive values (interactional motivation). Therefore, the agent gets the perception of this world and generates proper behaviors in different and complicated situations. Meanwhile, our agent can discover a long sequence of "correct" actions to find a configuration of the environment that yields the non-stationary valence.

Nevertheless, the agent has to retrospect all previous learned composite interactions to retrieve the ones whose pre-interactions are matched with the current enacted interaction. With interactions continuing, the recorded enacted interaction traces progressively grow longer, then the agent will spend a long time to activated all eligible composite interactions for anticipations. In addition, the utility rate of composite interactions needs to be improved. The agent activated almost all composite interactions but few are proposed for intending. Although the agent can be easily qualified for the work in the environment designed in this paper, when the environment becomes more complex, this shortcoming will be easily shown.

Valence initialization is an another issue. Different valence allocation strategies are

crucial to the performance of agents in interactions. According to the commonsenses of human beings, assuming that the agent can successfully take a step forward, it should get positive hints, and collisions with the wall should get negative feedback. As for the agent, it starts interaction without any prior knowledge, which means it should understand the feedback of different behaviors through their own interaction with the environment. For the initialization of Valence, it is inevitable to have a certain influence on the cognitive process of the agent to some extent. The better valence allocation strategy could get better interaction performance of the agent. In the following research, I need to study to reduce human intervention as much as possible, let the agent explore for itself, and discover and optimize how to find the best valence allocation strategy from the interaction.

In the following chapters, I will introduce the experimental scenarios to demonstrate the model that proposed and discuss open issues of this work, like concerns of the allocation of valence and the combinational explosion of composite interactions.

Chapter 7

Methodology and experimental scenario of the BEL-CA

Contents

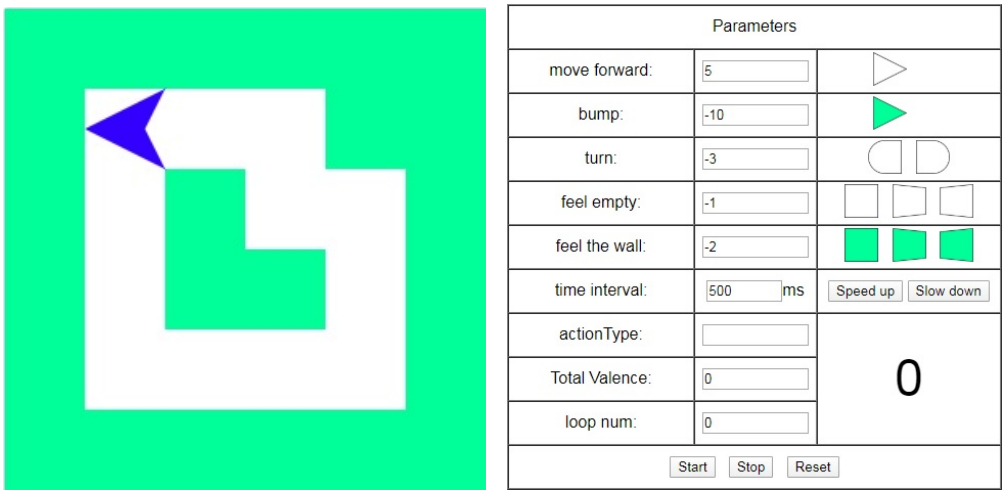
In this chapter, I set up an experimental scenario and introduce an implementation of analyzing agent's interaction traces to demonstrate CCA's ability of bottom-up hierarchical sequential learning. The experimental scenario is designed based on the classic Small Loop Problem (SLP) [45, 175] as mentioned in Section 3.2.2, which acts as a benchmark of implementing and demonstrating cognitive emergence for an autonomous agent. Meanwhile, I verify the agent's capabilities of self-adaptation and flexibility by modifying the environment to simulate interaction scenarios that it hasn't experienced before.

This chapter is structured as follows. The section 7.1 describes the experimental settings which include the simulation of the agent and the environment, the experiments and the initialization of parameters. Section 7.2 explains the implementation of the GAIT which facilitates to analyze agent's interaction traces. Section 7.3 explains the detail learning process for the agent interacts with the environment and the structure behaviors it has learned in each decision-making. Section 7.4 reports the results from the GAIT. In section 7.5, I present new simulations for which demonstrating BEL-CA's performance in autonomous robots. Section 7.6 gives a conclusion.

7.1 Experimental settings

The agent is presented as a blue head arrow and initialized with a random direction. At the beginning of the interaction, the agent is positioned in the upper left corner of the environment and oriented to the left. Different from classic Small Loop environment, our environment is designed changeable for simulating agent's performance between familiar and unfamiliar environments, and dealing with different levels of complexity of interaction scenarios. The changeable environment means that if you click any walls in the environment (except the boundaries wall in the environment), this wall will be removed and the location becomes accessible. Otherwise, it will be filled with a wall and become impassable. And also this environment's modification will be synchronized with the proposed constructivist cognitive architecture in time.

In order to distinguish different experiments, I utilize divers icons like a triangle, left and right half-circle square, left and right trapezoid and square to represent moving forward, turn left and turn right, touch left, touch right and touch front respectively. With colors of green and white to indicate interactions that the agent enacts with the same experiment but receives different possible feedbacks from the environment, but it ignores the meaning of these interactions. The experimenter can preset the valences of primitive interactions on the "Parameters Panel" (as shown in the right figure of Figure 7.1). The interaction interval could be changed by afforded buttons or edited directly to speed up or slow down the speed of interactions. The "actionType" indicates the current experiment the agent enacts, the "Total valence" represents the cumulated valence the agent has received from current and previous all interactions, and "loopNum" presents the times that the agent makes decisions. With buttons of "Start", "Stop" and "Reset" could control the interaction to start, suspend or reset to the initialization state. In order to better observe the interaction between agents and the environment, as well as the agent's gradually learning process, I have proposed and developed a toolkit named "Generating and Analyzing Interaction Traces toolkit" (GAIT) as a way to investigate the detailed learning process for agent interacting with the environment and each structured behaviors it has learned within each decision-making.



(a) The environment for the agent to interact

(b) Parameters to control the interaction

Figure 7.1: The environment and experimental settings.

7.2 Generating and Analyzing Interaction Traces toolkit (GAIT)

In this section, I introduce the implementation of GAIT to explain the detail learning process for the agent interacts with the environment and the structure behaviors it has learned in each decision-making on cross-platforms. The framework of GAIT records all information in agent's each interaction with the environment, which includes intended interaction, enacted interaction, anticipations with its selection process and all learned or reinforced composite interactions, forming a continuous interaction traces followed with the timeline.

One of the classic developments of GAIT is based on a Client-Sever architecture. The animation of the interaction scenario that presents the dynamic interaction between the agent and the environment is designed by the Canvas of HTML5 and the interaction traces are drawn by SVG components in the front page as the client side. In the server side, the algorithm of the BEL-CA is implemented in a Java Servlet and a server of Tomcat9.0 is prepared for responding requests from the front page. In order to be able to selectively display the agent's interaction traces and to facilitate interactive operations on the interaction traces, I have designed different layers to display various information in the interaction (as shown in Figure 7.2). By default, except the Layer 1, other layers always remain invisible. After starting the an operation, the corresponding layer will be displayed accordingly. Meanwhile, in order to facilitate to display of the information of concern and keep the page tidy, corresponding methods are also provided to hide unnecessary information. The implementation of these layers is also drawn by SVG but the implementation of functions behind operational buttons are based on Javascript and the CSS3.

Furthermore, in the area of interaction traces, there exists several inducing fields that facilitate to observe the structure of interactions and processing procedure (as shown in Figure 7.3). One of the inducing fields in the interaction traces shows that each time move the mouse on the primitive intended/enacted interaction symbols, a tip window will pop out to present the all newly learned or reinforced composite interaction with its hierarchical structure and layers information (as shown in Figure 7.3). The composite interactions surrounded by green rectangles indicate these composite interaction were new

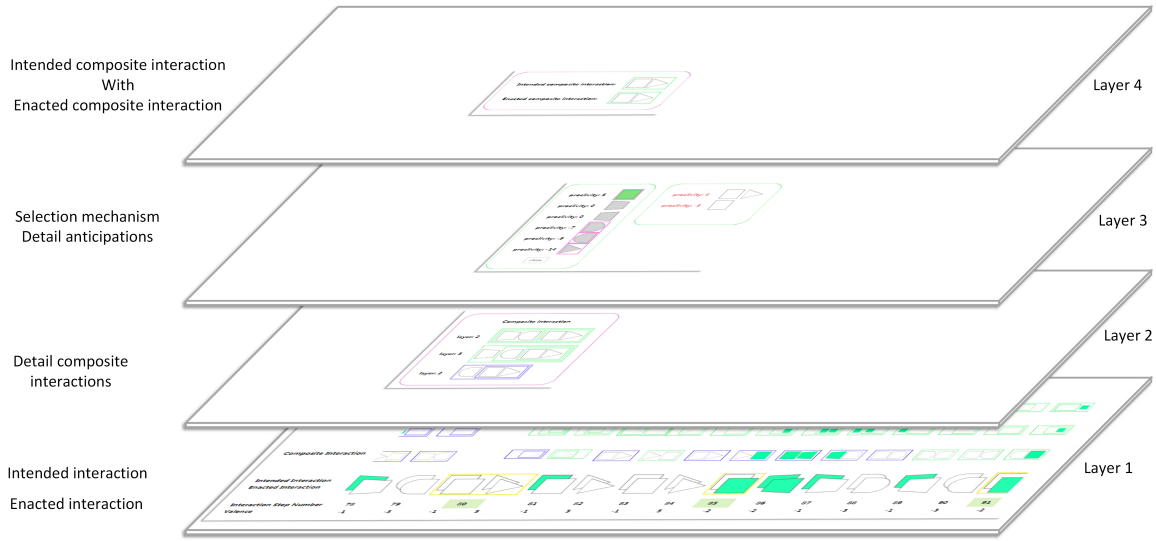


Figure 7.2: The layers of the GAIT in the front page that is designed to display various information in the interaction. Particularly, the bottom Layer 1 displays all enacted interactions affiliated with their intended interaction, and a pair of thumbnails of composite interactions followed by the timeline. The Layer 2 shows the detailed structure of composite interactions which is shrunken in the Layer 1. Layer 3 was used to present the selection mechanism in BEL-CA and the anticipations for each interaction. Considering the situations that the agent enacts an intended composite interaction, the Layer 4 displays the comparison between the intended composite interaction and the enacted composite interaction.

learned and the ones surrounded by blue rectangles indicate these composite interactions already learned before but were reinforced in this interaction.

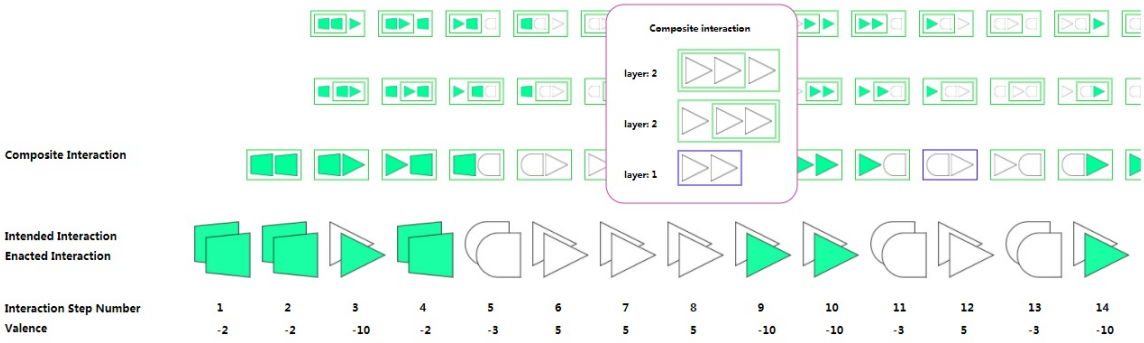


Figure 7.3: The inducing field of primitive intended/enacted interaction.

After moving out the mouse from primitive intended/enacted interaction symbols, the composite interaction tip window will simultaneously disappear. Operations like left-clicking any enacted interactions, a tip window with a list of experiments (icons with grey color) will pop out and sorted by its proclivities (as shown in Figure 7.4). For the cases where experiments could propose anticipations (surrounded with the pink rectangle), pick one experiment and continue left-clicking on it, the select experiment fills with light green to specify it has been chosen and pop up an another tip window with a list of detailed anticipations, also this list is sorted by their proclivities. The proclivity of detailed anticipations with red color indicates its weight is lower than the threshold, while the green color indicates its weight is greater or equals with the threshold. Also, there exists

a “close” button at the bottom of the experiments’ tip window for close this experiments’s tip window with their detailed anticipation tip window, which facilitates to display other interaction information.

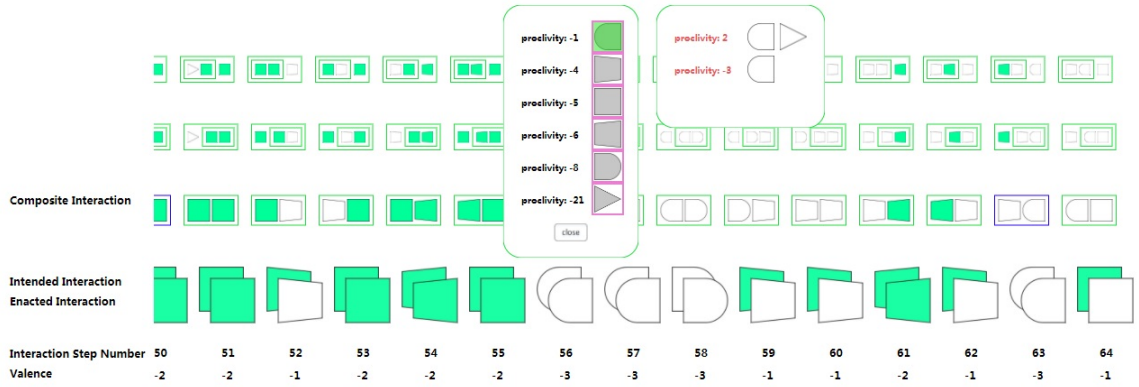


Figure 7.4: The tip windows experiments with their anticipation tip window.

While an agent is enacting composite interactions, there are several inducing fields (illustrated with light green rectangle) on the “loop number” (as shown in Figure 7.5). Moving the mouse on this area, a tip window that includes the intended composite interactions with its enacted composite interaction will pop out to show the detailed interaction process. Similarly, when moving out the mouse, this tip windows will simultaneously disappear. In order to identify the range of intended/enacted composite interactions, all primitive interactions that the agent have enacted in the intended/enacted composite interactions are surrounded with a yellow rectangle.

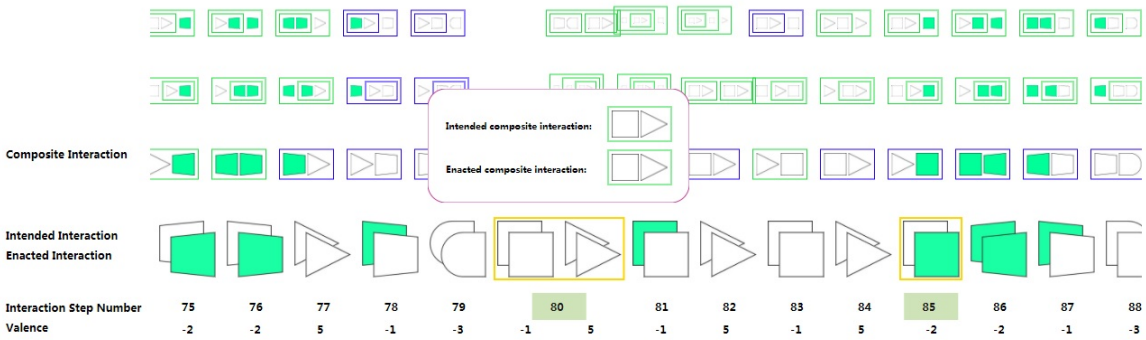


Figure 7.5: The inducing field of enacting composite interactions.

The scroll button at the bottom of the interaction traces window could be used to show all previous interaction traces. In our test, the proposed toolkit could support tens of thousands of interactions which makes it easy to look back at all previous interactions.

7.3 Interaction traces analysis

At the beginning of interacting with the environment, the agent starts the journey of “perception of the world” without endowing any prior knowledge, nor specific goals for it to achieve. The agent randomly selects an experiment (feel right) and intend this experiment’s default primitive intended interaction (the green trapezoid). With feedback from the environment, agent receives the same green trapezoid (as shown in Figure 7.6). At step 2, the agent memorizes the previously enacted interaction with current enacted

interaction forms a composite interaction. Particularly in step 3, combined with previous enacted interaction and super-interaction, the agent constructs one composite interaction that composed by previous enacted interaction with current enacted interaction and two third level composite interaction that composed with previous two-step enacted interaction and super-interaction.

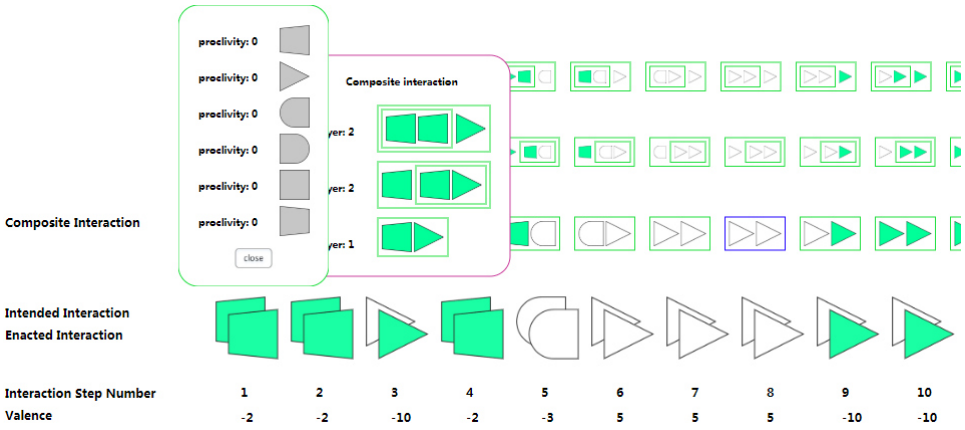


Figure 7.6: The first several interactions and composite interactions construct process.

The proposed intended interaction appears at step 9 and step 14 (in Figure 7.7), the experiment "move forward" has the highest proclivity and its intended interaction (the white triangle) is proposed with the highest proclivity (the proclivity value is 15). Then the agent intends this white triangle (move forward) and gets the same white triangle (the agent successfully moves forward a step), then this attempt intended interaction as a *success*. While the opposite situation happened in step 14, the agent intends the same white triangle while bumping with the wall, it gets the a green triangle, hence this interaction is considered a *failure*.

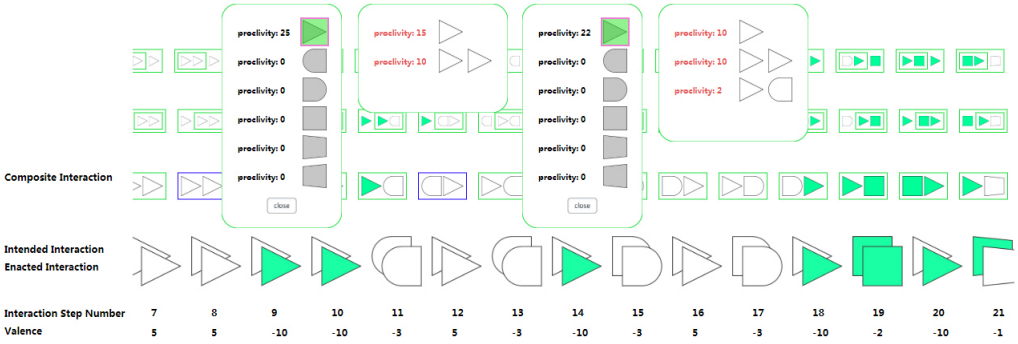


Figure 7.7: Enact the same intended interaction with different feedback.

At step 23, the agent is going to enact a composite interaction, with the reason that this anticipation's weight is less than the threshold, then the agent intends the first primitive interaction (left half-circle, turn left) of this composite interaction and receives the same enacted interaction (see Figure 7.8). In this implementation, I use different colors to identify whether the anticipations' weight beyond the threshold or not, the color red means its weight less than the threshold then the agent just intends the first primitive interaction of the intended interaction (in this case, the white left half circle), while the green signifies its weight is greater or equal with the threshold which means the agent will enact all primitive interactions in this composite interaction sequentially as a whole.

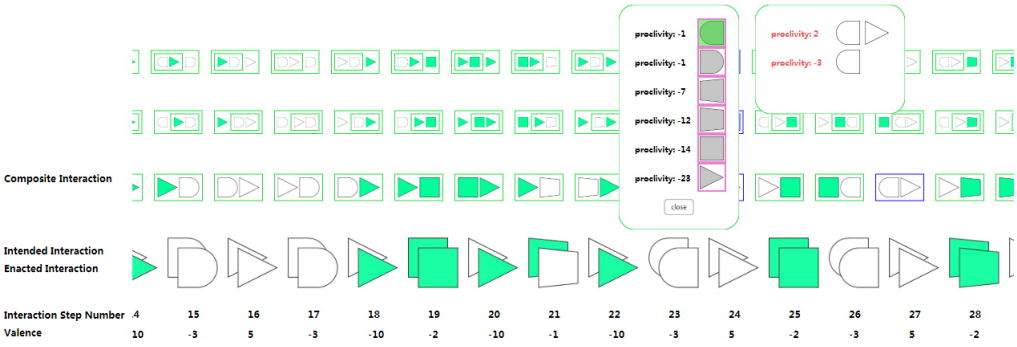


Figure 7.8: The enacted composite interaction's weight less than the threshold.

At step 119, the agent gets an intended composite interaction with its anticipation weight beyond the threshold, then the agent sequentially intends all primitive interactions in it and successfully receives the same enacted composite interaction (see Figure 7.9). The agent combines this enacted composited interaction with previous enacted interaction and previous learned composite interaction construct higher-level and more complex composite interaction, which present much more complicated behavioral patterns have learned by the agent for the case that it could generate proper behaviors in the future.

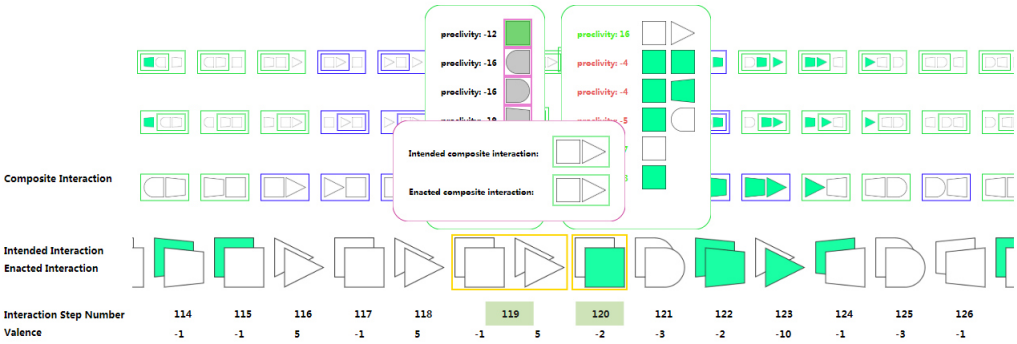


Figure 7.9: The agent enacts composite interaction and constructs higher-level composite interaction.

As interactions continue, the agent gradually constructs higher-level composite interactions which represent complex structured behaviors that allow the agent to successfully interact with the environment and learn to avoid unfavorable interactions (bumping with the wall) by using regularities it has learned. More complicated behavioral patterns have constructed and it could generate properly with different situations (see Figure 7.10). The learning process goes stabler with agent's interaction with the environment and the growth of constructing composite interactions will become slower from the rapid development at the beginning of interaction.

7.4 The results

7.4.1 The agent's learning process exported from the GAIT

The interaction traces exported from the GAIT provide us a possible path to understand agent's cognitive development from interactions with the environment. As shown in Figure 7.11, at the beginning of interaction, the agent tries to enact predefined primitive interac-

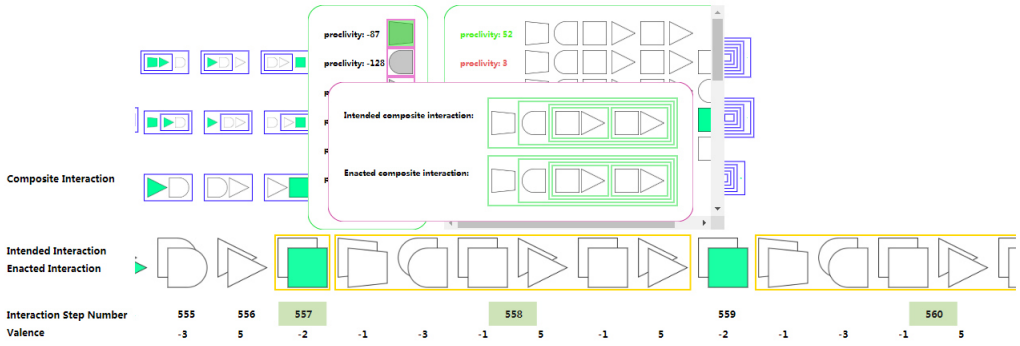


Figure 7.10: Enacting complicated composite interaction.

tions and occasionally bumps with the wall. As interactions continue, the agent gradually learns regularities from its interaction experiences and organizes them into structured behaviors. With the agent constructs perception of the environment, it can generate proper behaviors and reduce the collision with the environment. Especially from the 357th interaction, the bumping phenomenon starts to disappear and the agent enables to generate proper behaviors based on accurately recognizing the context. Moreover, the agent could successfully interact with its environment and start preventing unfavorable interactions using regularities that it has learned.

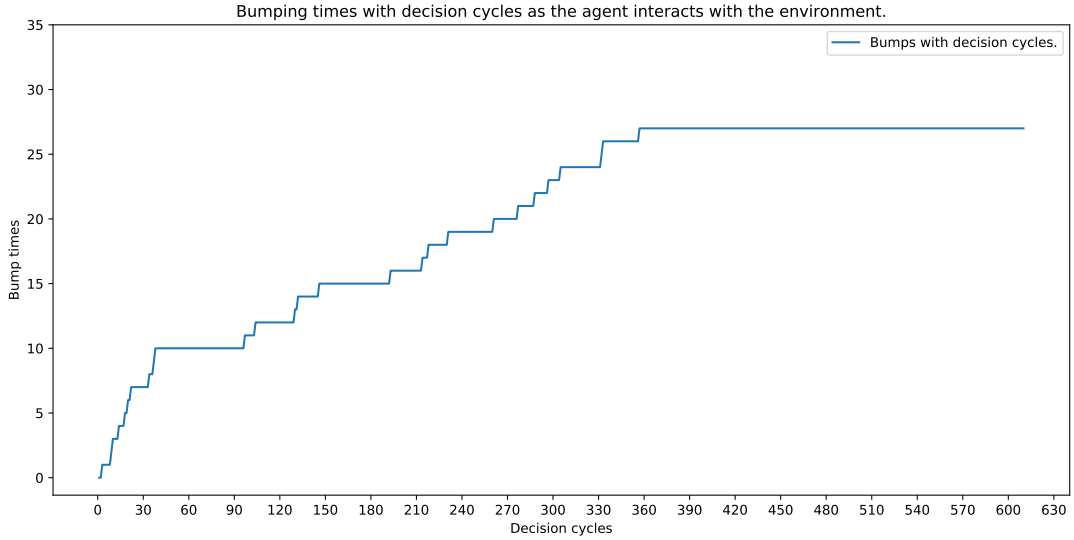


Figure 7.11: Bump times with decision cycles as agent interacts with the environment.

From the perspective of total valences can also demonstrate agent's learning process in cognitive development of the environment. Total valence, which represents the cumulated valence the agent has received from enacted interactions. As shown in Figure 7.12, in the initial interactions, the total valence continued to reduce with the agent continuously bumps with the wall and tries to find ways to enact interactions have positive valence. In this period, the agent occasionally move forward successfully, but in most cases where the agent is making various explorations and attempts to prevent bumping with the wall. Since 357th interaction, the total valence starts to rise, and the increase goes faster until the increase rate tends to be stable, which as a way to prove the emergence of sense-making in agent's interactions with the environment. This could also be confirmed from the Figure 7.13 that shows the average valence with decision cycles in agent's interactions

with the environment.

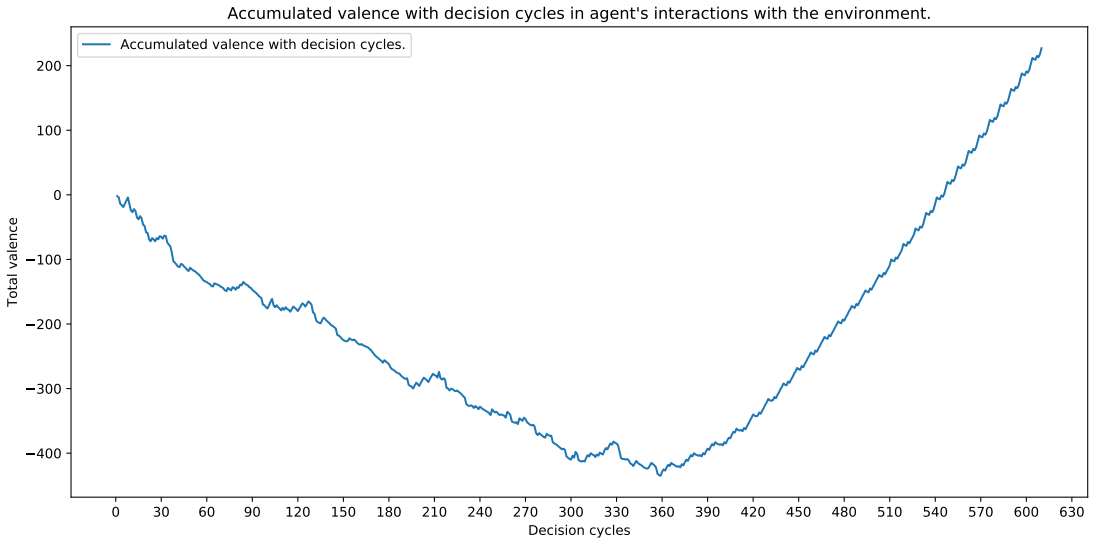


Figure 7.12: Accumulated valence with decision cycles in agent's interactions with the environment.

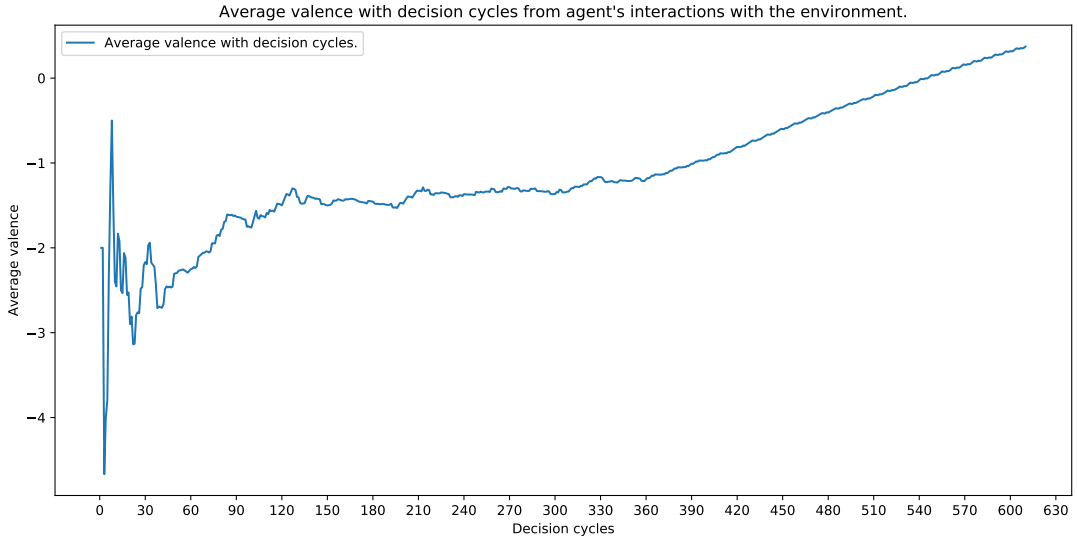


Figure 7.13: Average valence with decision cycles in agent's interactions with the environment.

7.4.2 The threshold of regularity sensibility in the interactions

Meanwhile, from the result reported from the GAIT, the threshold of regularity sensibility (as mentioned in Section 6.3.1) results in the adoption of potential satisfying higher-level schemes. As shown in Figure 7.14, the lower regularity sensibility threshold results in a faster adoption of potentially less satisfying higher-level schemes. A higher regularity sensibility threshold results in a slower adoption of potentially more satisfying higher-level schemes.

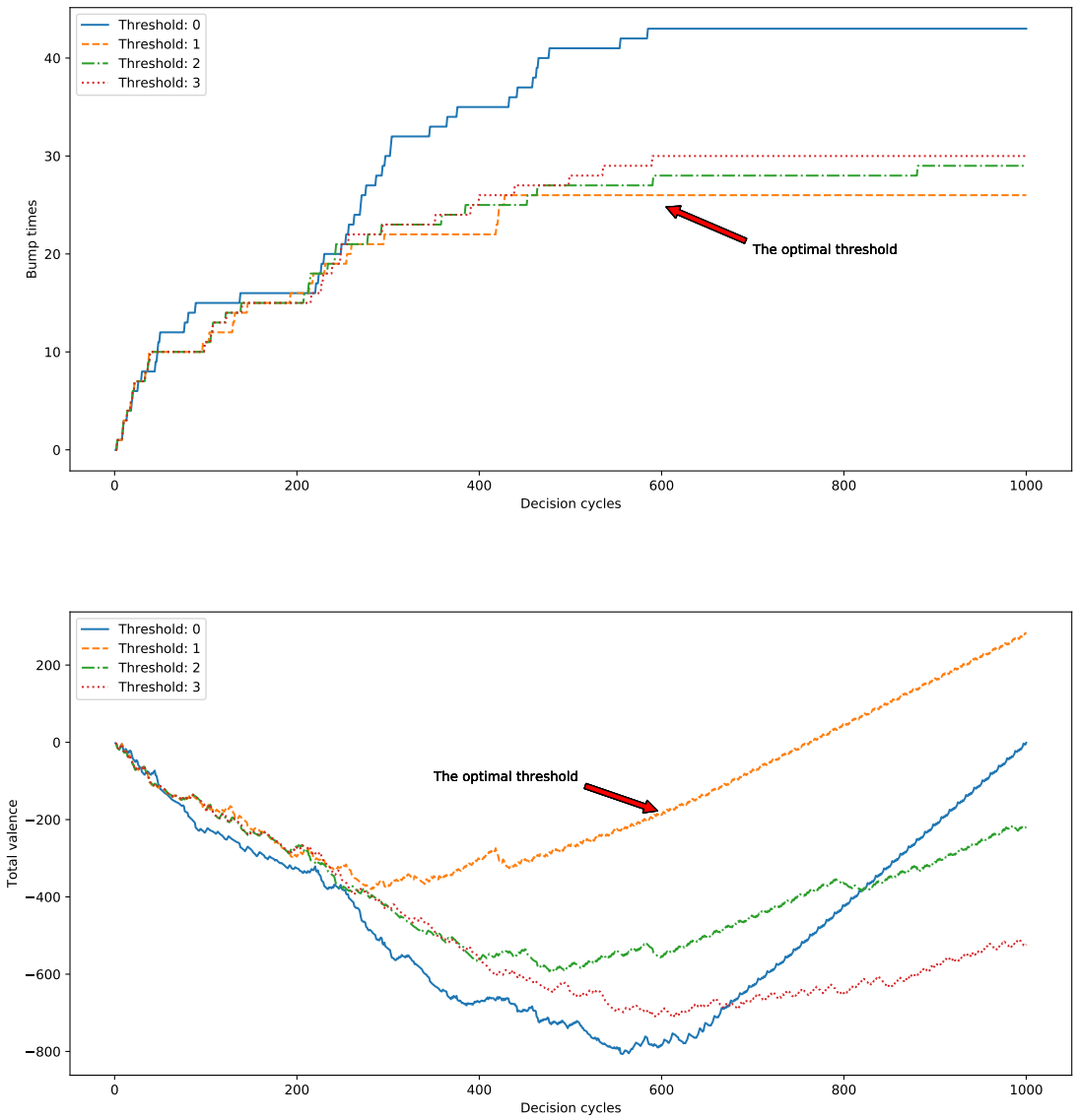


Figure 7.14: The results reported from the GAIT of interactions in different thresholds.

7.4.3 The growth of the episodic memory and the surprises exported from the GAIT

In the learning process of agent's cognitive development, the construction of composite interactions plays as a way to represent structured behaviors emergence from interactions, or in other words, learning regularities from interaction afforded by the environment. As interaction continues, the agent is building the perception of the environment and avoiding unfavorable interaction experience based on regularities it have learned, the newly learned composite interaction will gradually decrease until it disappears. Instead, most of the composite interaction will be gradually reinforced in subsequent interactions. This complies with agent's self-motivation, which successfully enacting sequences of interactions (autotelic motivation) and preferably enacting interactions that have predefined positive values (interactional motivation). Therefore, as the interaction progresses, the "surprises" in enacting intended interactions will decrease, the growth of the number of composite interactions should slow down, and the hierarchical structures will also become more stable. As shown in Figure 7.16 and Figure 7.15.

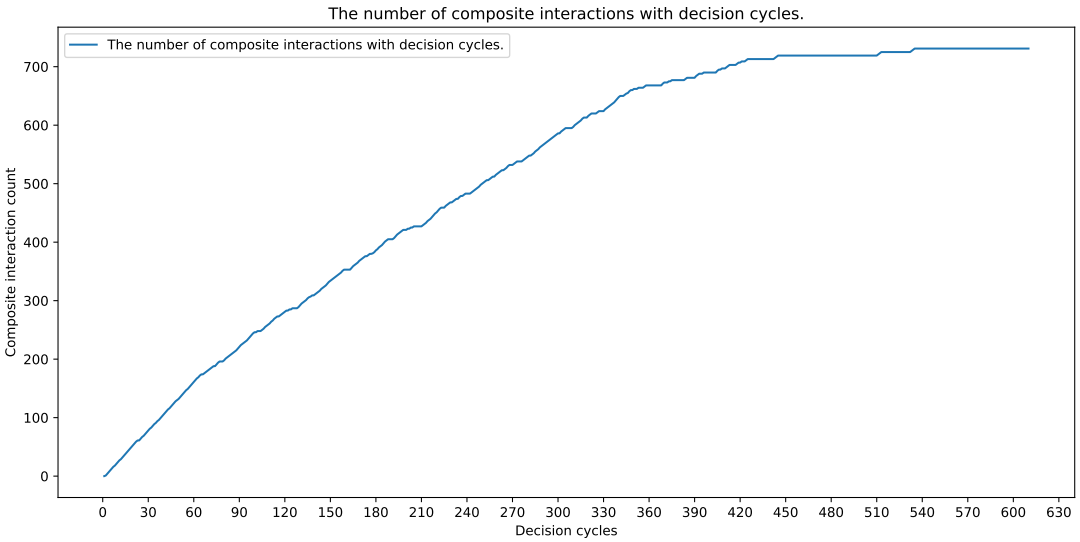


Figure 7.15: The growth of composite interactions with decision cycles in agent's interactions with the environment.

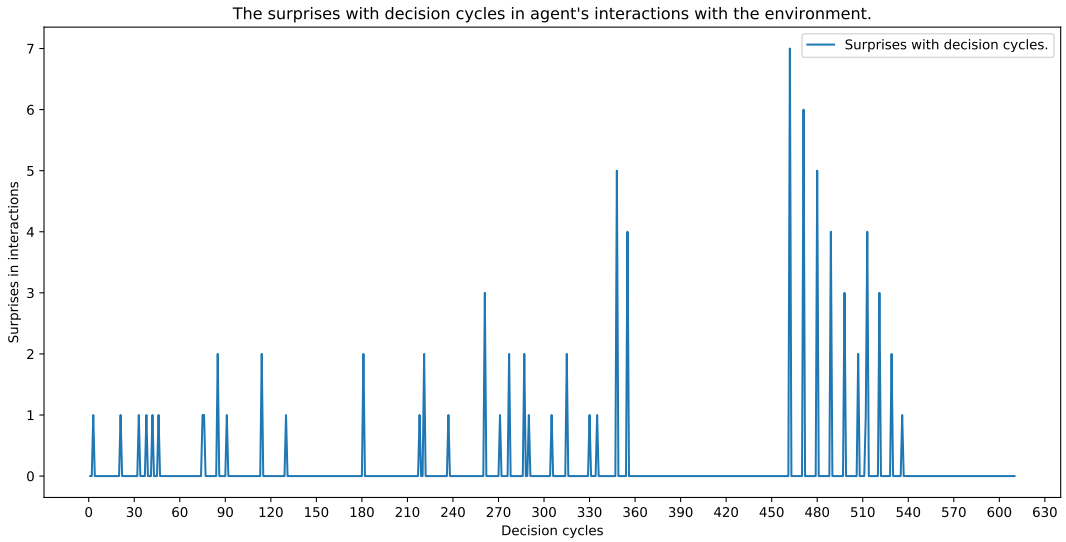


Figure 7.16: The surprises with decision cycles in agent's interactions with the environment.

In Figure 7.16, the surprise in agent's interaction decreases until it disappears. As interaction progresses, higher-level composite interaction will constructed and selected to enact, thus surprise in enacting this composite interaction grows stronger than before, but this "surprise" will decrease with the agent gets familiar with this intended interaction and the context. In Figure 7.15, the construction of composite grows rapid in the initial interaction process. As agent gets familiar with regularities patterns, this growth will slow down until goes stable. Thus the number of composite interaction remains constant.

7.4.4 The agent's performance in the changed environment

In the model of BEL-CA, the agent not only continuously constructs the knowledge of the environment but also acquires capabilities of self-adaptation and flexibility for generating proper behaviors in diverse situations. As shown in the interaction traces exported from the GAIT, from the 357th interaction, the agent could successfully interact with its environment and start avoiding unfavorable interactions (the collisions with the walls) using regularities that it has learned. In the 500th interaction, the environment has beed changed (as shown in Figure 7.17) for examining the agent's performance in the changed environment.

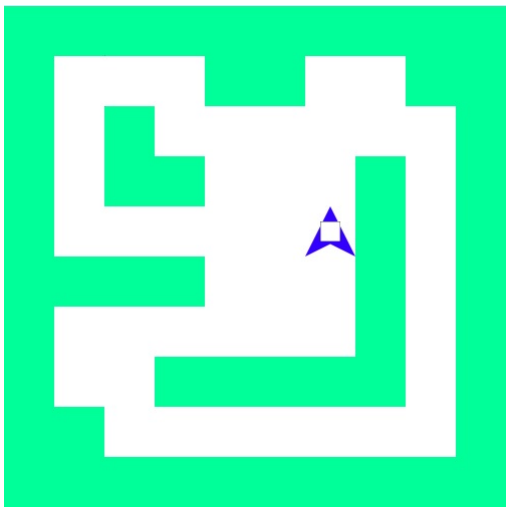


Figure 7.17: The changed environment.

With interaction traces exported from the GAIT, the agent reuses the regularities that it has learned in the previous environment and dynamically adapted new patterns from the interaction experience. At the beginning of interactions in the new environment, the proposed intended interactions are not satisfied with the context, thus there exists a period of oscillations in agent's performance, which is mainly manifested in the increase of bumping times and the oscillations of valences in the interaction (as shown in Figure 7.18). Nevertheless, with interaction continues, the agent gradually absorbs novel knowledge of the environment and successfully interact with its environment with generated behaviors, which represent that the agent has acquired capabilities of self-adaptation and flexibility. The bumping phenomena starts to disappear and valence is rising (as shown in Figure 7.18).

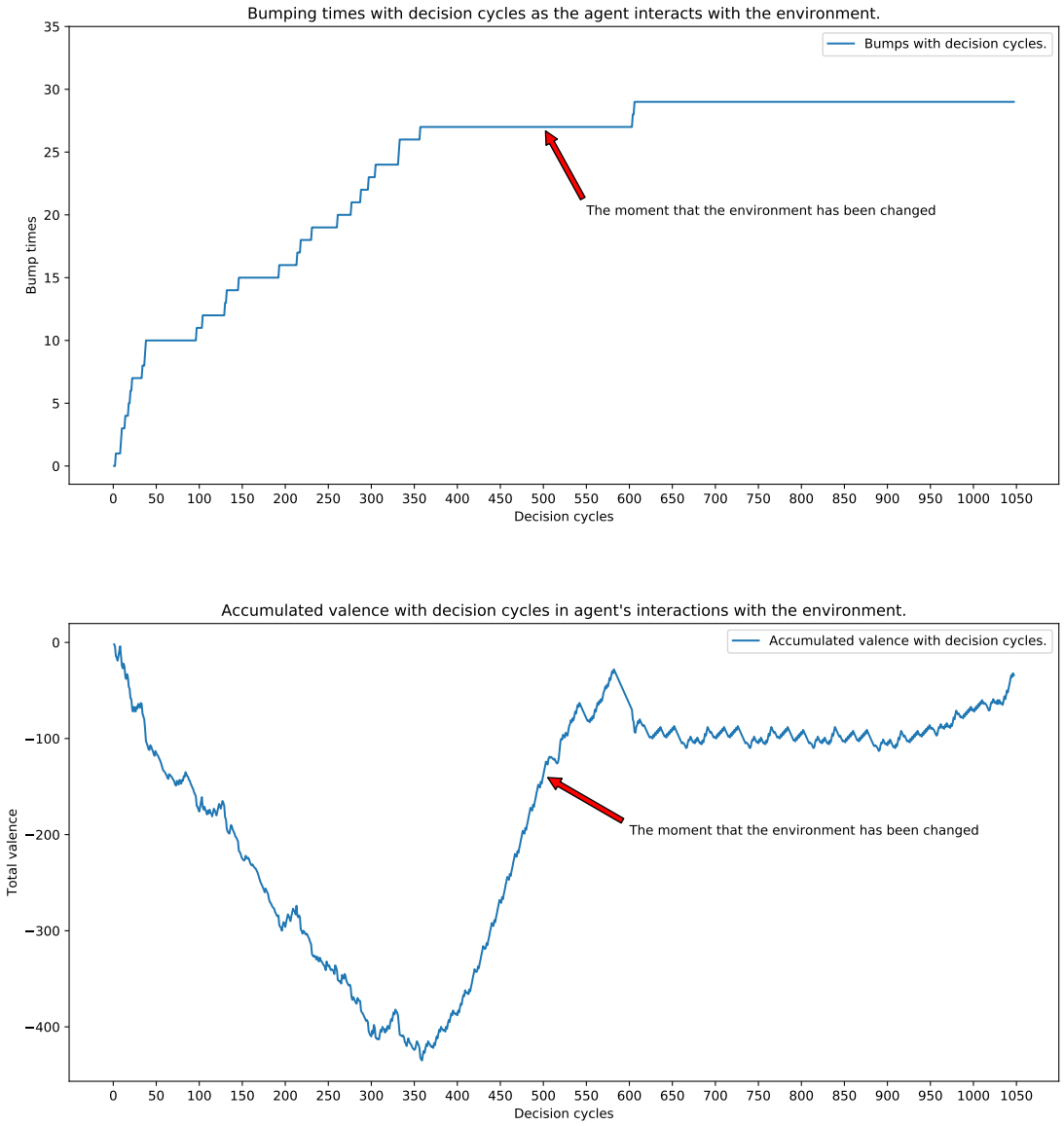


Figure 7.18: The results reported from the GAIT of agent's performance in the changed environment. Since the environment has been changed in the 500th interaction, the agent experiences a period of oscillations in the interaction with the environment. After then, the agent gradually absorbs novel knowledge of the environment and successfully interact with its environment with generated behaviors, thus the bumping phenomena starts to disappear and the valence is rising.

7.5 Simulations in autonomous robots

The previous section demonstrates CCA's ability of bottom-up hierarchical sequential learning in a simple simulation, which an autonomous agent is designed as a blue head arrow with randomly initialized direction in the environment of Small Loop Problem (SLP). As interaction continues, the agent gradually constructs the perception of the environment and acquire capabilities of self-adaptation and flexibility with diverse interaction scenarios.

However, for autonomous robots, the learning process is far more complex than that. For example, even in the realization of simple experiments, like moving or turnings, robot needs to control its own mechanical and power system effectively and accurately. Moreover, the enacted interactions are not available immediately but delayed for the reason that it needs to wait until the robot finishes all experiments in the enaction of an intended interaction sequentially. The simulations of BEL-CA for autonomous robots is interesting but also it's a big challenge. In this section, I describe the approaches of implementing this new simulation, for which demonstrating BEL-CA's performance in autonomous robots.

7.5.1 Robots and the environment

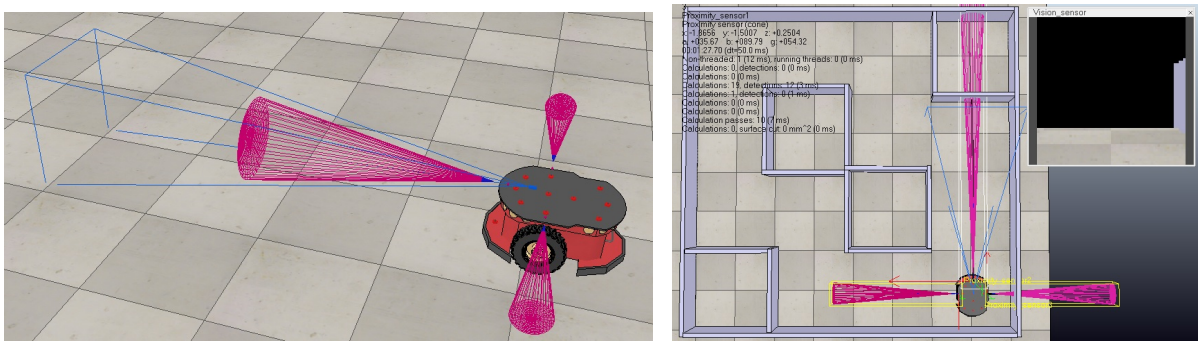
For the simulation of autonomous robots, the development of GAIT also follows a Client-Server architecture but in a different way. Our new simulation for autonomous robots is based on the platform of V-REP, and the robot in this simulation is a *Pioneer 3-DX*, a small lightweight two-wheel and two-motor differential drive robot. The V-REP is a cross-platform robot simulator which provides various robots and modules and allows to design desired robots and interaction environment according to different kinds of requirements. Meanwhile, the robot is equipped with sixteen ultrasonic sensors, three proximity sensors, a vision sensor of camera. In addition, the robot in this simulation as a client side and the server side of the algorithm BEL-CA is implemented by Python. The interaction traces are drawn by framework of PyQT5 and combined with Python's original GUI toolkits of Tkinter.

As shown in Figure 7.19(a), the robot has sixteen build-in ultrasonic sensors that are placed around the edge with different angles, and also it is equipped with three proximity sensors in different directions (forward, left and right), a vision sensor of camera on the head of the robot in the forward direction to capture the images of the environment in front of the robot. Particularly, the forward proximity sensor has a bigger detect range (Range: 3.5 m) and smaller angle (Radium: 0.005m, Angle: 5°) than the others two directions (Range: 1.5m, Radium: 0.005m, Angle: 10°), which allows it to detect farther and reduce the interference from the environment (typically, the walls). The proximity sensors not only enable the robot to obtain the *distance* of the object in the corresponding direction, but also get the object's normalized *surface normal vector* and the detected point coordinates.

In order to avoid physical collision between the robot and the wall, I set the robot to move forward only in conditions that the distance is greater than a certain threshold, otherwise it is not allowed to continue to move forward. At the same time, with the obtained *surface normal vector* of the object, the robot can continuously handle the controllers of two wheels on both sides to keep the *surface normal vector* vertical, which aligns the robot with the environment after each execution of the experiment.

Meanwhile, the environment is designed as the classic Small Loop Problem (SLP) (as shown in Figure 7.19(b)). In the initialization environment, the agent is positioned in the

lower right corner of the environment and oriented to the front. The walls around the robot are designed to be collidable, measurable,detectable and renderable, which allows the proximity sensors and vision sensors to function to obtain corresponding parameter data.



(a) The robot of Pioneer 3-DX.

(b) The classic environment of SLP for the Pioneer 3-DX.

Figure 7.19: The environment of SLP with a Pioneer 3-DX robot.

7.5.2 The implementations of experiments

The implementation of primitive interactions for the robot

The implementation of primitive interactions in robot plays an elementary and fundamental role in the simulation for autonomous robots. Learning process starts from these basic implementations. The first implementation of the robot is to move the robot forward. As mentioned before, the robot only moves forward in conditions that the distance between it and the wall in front of it is greater than a certain threshold, otherwise it is not allowed to continue to move. The calculation of the *distance* comes from the normalization of detected point coordinates from the proximity sensor.

$$\|distance\|_F = \sqrt{\sum_i p_i(x_i, y_i, z_i)^2} \tag{7.1}$$

An advantage of the robot Pioneer 3-DX is that its two wheels can be independently controlled, which can facilitate us to achieve precise control of the robot's forward and steering. To make the robot move forward, the only need is to set a certain speed and duration to let it reach a certain distance forward. For steering, it needs to set a duration, and set the same speed for the both two wheels but move in the opposite direction.

Alignment with the environment

Each time the robot executes an experiment, it needs to ensure that it is aligned with the environment, so that it is convenient for the robot to execute a series of experiments afterwards. As described above, the robot is equipped with three proximity sensors with three different directions (forward, left and right), which allows the robot to obtain distances from these directions and their normalized *surface normal vectors*. As shown in Figure 7.21, the robot is placed in the environment with an angle, the forward proximity sensor affords the coordinates of surface normal vector of the wall in front of the robot. The coordinate of the surface normal vector is a three-dimensional parameter in the form

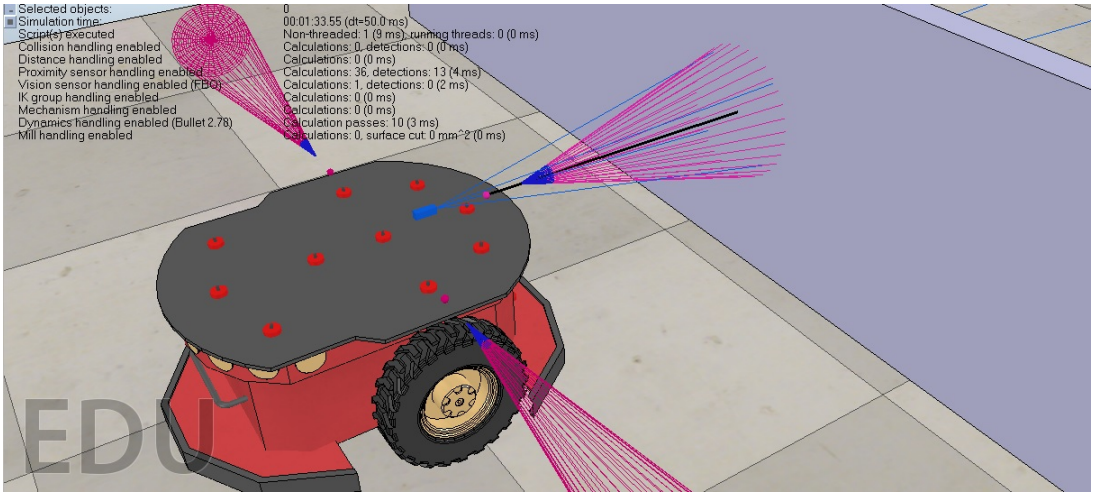


Figure 7.20: The detected points from the proximity sensor.

of [x,y,z] based on the coordinate system of the proximity sensor. As shown in Figure 7.22, the coordinate value of x,y,z respectively represent the direction of the coordinates along the red line, green line and blue line. When the direction of the robot is gradually perpendicular to the wall surface, the coordinate value of z will gradually tend to 1, and the coordinate value of x will gradually tend to 0. Based on this feature, the adjustment of the surface normal vector by controlling the movement of the two wheels Vector to realize the alignment of robot and environment.

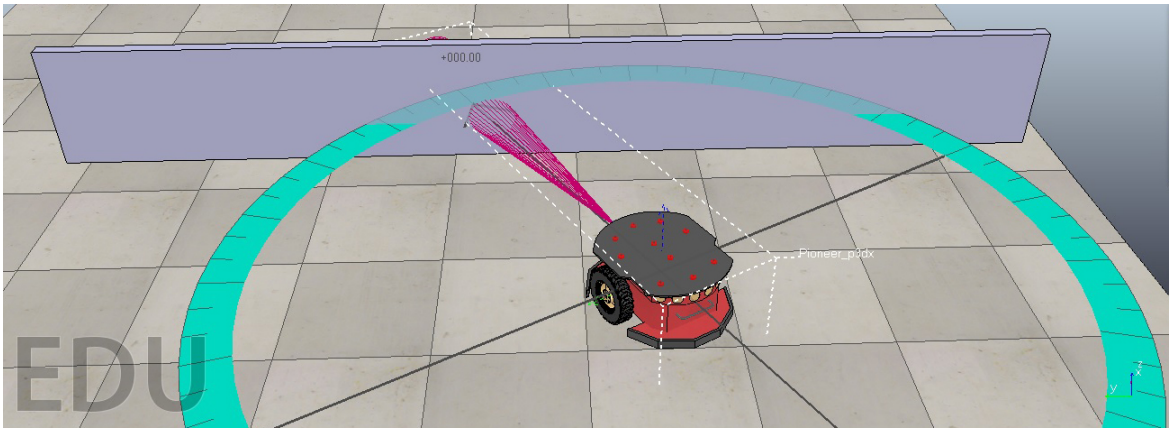


Figure 7.21: The alignment of of the robot in the environment.

7.5.3 Performance

After implementing the elementary experiments, the robot starts interacting with the surrounded environment and learning regularities of interaction afforded by the environment. Similarly, I deployed the toolkit of GAIT in this simulation to observe robot's learning process and the construction of hierarchical behaviors from its interaction traces.

7.6 Conclusion

In this chapter, I set up an experimental scenario and introduce an implementation of analyzing agent's interaction traces to demonstrate CCA's ability of bottom-up hierarchical sequential learning. The experimental scenario is designed based on the classic Small

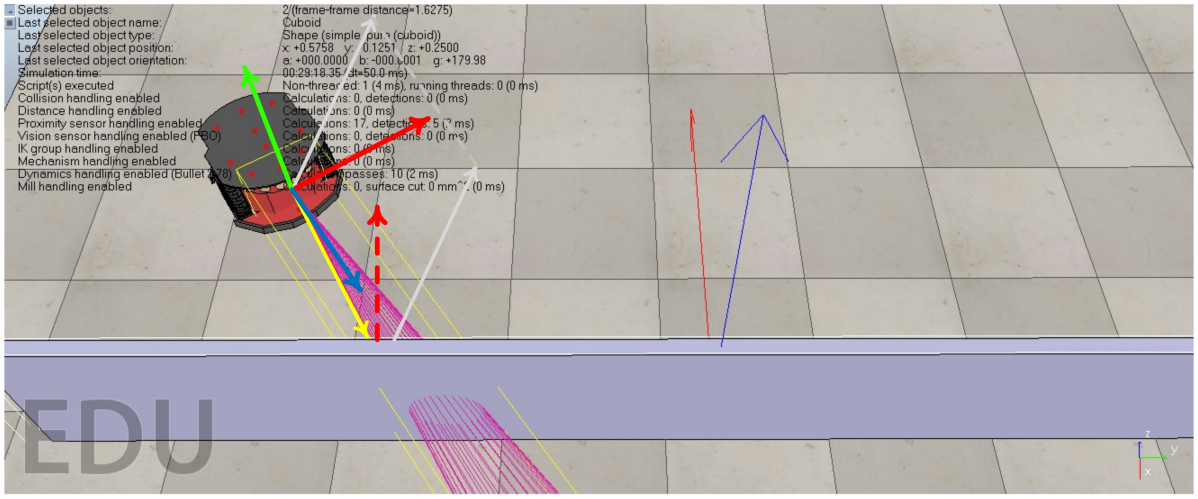


Figure 7.22: The coordinate system of proximity sensors and the description of its coordinate values.

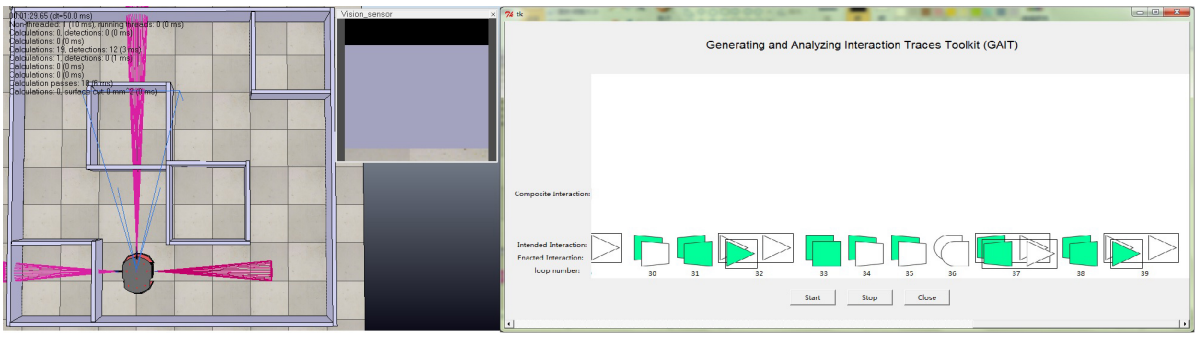


Figure 7.23: The combination of GAIT in simulations of robot on V-REP.

Loop Problem (SLP) [45, 175], which acts as a benchmark of implementing and demonstrating cognitive emergence for an autonomous agent. Meanwhile, I verify the agent's capabilities of self-adaptation and flexibility by modifying the environment to simulate interaction scenarios that it hasn't experienced before.

Chapter 8

Conclusion, open issues and perspectives

Contents

8.1 Conclusion

This dissertation introduces a computational model of Constructivist Cognitive Architecture (CCA) as the way towards simulating the learning mechanism of infants' early-stage cognitive development based on theories of enactive cognition, intrinsic motivation, and constructivist epistemology. Meanwhile, the CCA allows a self-developing agent to autonomously acquire the perception of the environment and obtain capabilities of self-adaption and flexibility to generate proper behaviors for tacking with diverse situations.

The BEL-CA, a bottom-up hierarchical sequential learning model for autonomously learning of hierarchical sequences of behaviors and obtain capabilities of self-adaptation and flexibility. Following with three processes of assimilation, accommodation and equilibrium in constructivsm, the agent autonomously organizes schemes it has learned from interactions into the hierarchical manner, which allows the agent to gradually understand the meaning of different experiments and simultaneously infer the structure of the environment based on the patterns in the stream of interaction traces.

Meanwhile, combined with BEL-CA and the implementation of toolkit GAIT allow us to report and explain the detailed learning process and the structured behaviors that the agent has learned on each decision making step. We report an experiment in which the agent learned to successfully interact with its environment and to avoid unfavorable interactions using regularities discovered through interaction.

We evaluated the agent's cognition emergence based on a classic Small Loop Problem (SLP) environment, which not involves a final goal for the agent to achieve, nor the limitations of interactions ending. The changeable environment is designed for simulating agent's performance in different levels of complex scenarios. With recorded enacted interaction traces from GAIT and hierarchically structured behaviors the agent has learned, we found that the agent could gradually exploit the hierarchical regularities afforded by the environment and learn to avoid unfavorable interactions using regularities that it has learned. Within 357 interactions, it could successfully interact with its environment and generate proper behaviors for different situations.

The experiments report that certain interactions become meaningful to the agent, with which the agent learns to use them to inform its future behavior. This result demonstrates that the agent learns to perform active perception, that is, the agent actively uses certain interactions as a form of perception to inform its knowledge of the current situation. Additionally, the agent addresses the autonomous ontology construction problem at a rudimentary level. It learns to actively distinguish between types of phenomena afforded by its environment and to cope with these phenomena by successfully enacting learned sequences of interactions.

8.2 Open issues

8.2.1 The growth of composite interaction

In the learning process of agent's cognitive development, the construction of composite interactions represents the emergence of structured behaviors from interactions, or in other words, learning regularities from interaction afforded by the environment. Since in each interaction, the agent always combines the previously enacted interaction and constructed composite interaction to form new composite interactions, doubts concern the growth of composite interactions as the interaction progresses.

At the beginning of the interaction between the agent and the environment, the growth of composite interaction is indeed very rapid. However, as interaction continues, the agent gradually acquires the perception of the environment and avoids unfavorable interaction experience based on regularities it have learned, the newly learned composite interaction will gradually decrease until it disappears (as shown in Figure 7.15). With the reason that most composite interaction will be gradually reinforced in subsequent interactions and few new interaction experience is generated. This complies with agent's self-motivation, which successfully enacting sequences of interactions (autotelic motivation) and preferably enacting interactions that have predefined positive values (interactional motivation). The same reason can be used to explain the "surprises" in enacting intended interactions will decrease as the interaction progresses.

Meanwhile, as shown in Figure 7.15, the growth of composite interactions slows down, and the hierarchical structures will also become more stable. As shown in Figure 7.16, we can find that the surprise in agent's interaction decreases until it disappears. As interaction progresses, higher-level composite interaction will constructed and selected to enact, thus surprise in enacting this composite interaction grows stronger than before, but this "surprise" will decrease with the agent gets familiar with this intended interaction and the context. In Figure 7.15, the construction of composite grows rapid in the initial interaction process. As agent gets familiar with regularities patterns, this growth will slow down until it disappears. Thus the number of composite interaction remains constant.

8.2.2 Differences between the valence and the reward

Under normal circumstances, it's generally regard the valence and the reward as same things. Due to an intuitive reason that the valence and the reward both serve the decision making for the agent to select an intention (an single action or a series of primitive interactions) for the next interaction. However, the actual situation is that there are essential differences between them. For solving the confusions from the difference between the valence in the CCA and the reward from reinforcement learning paradigm, we give an detail explanation from the following three aspects: their origin, their formalism and the way of use.

The origin of the valence and the reward

In order to implement agent's self-motivations, the tendency to successfully enact interactions and nature preferences to enact interactions have positive values, we introduce an innate value system to indicate all interactions are not equal to the agent. In the CCA, we associate each primitive interaction with an innate scalar *valence* v_t as a way to simulate agent's inborn behavioral preferences and qualify the agent's "feeling" of each interaction experience. Therefore, the agent's interaction does not use a reward function or a problem representation, it constitutes neither a reward maximization algorithm nor a problem-solving algorithm. Instead, the agent's preferences are defined independently of an preference to the environment's states.

However, in Reinforcement Learning (RL) paradigm, agent is designed to learn what to do - how to map situations to actions - so as to maximize a numerical reward signal. The RL agent is not told which actions to take, but instead must discover which actions yield the most valuable reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next series situations and, through that, the all accumulated subsequent rewards. These two characteristics - trial-and-error search and delayed reward-are the two most important distinguishing

features of reinforcement learning. In the RL, the agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state. Moreover, the agent also must have a goal or goals relating to the state of the environment.

The formalism of the valence and the reward

In the CCA, valence as an attribute of sensorimotor interaction remains constant during the agent interacts with the environment. For primitive interaction, its valence has been preset in the initial stage of agent interacts with the environment. For composite interaction, its valence is the sum of the valences of all primitive interactions in it. The valence calculation as shown in equation 8.1 below, we can find that the valence function $v(i_t)$ is defined independently of any state of the environment and the agent is motivated to select an interaction for successfully enacting it, rather than for the outcome of the interaction or achieving a specific goal.

$$v(i_t) = \begin{cases} valence(i_t), & i_t \in I \\ \sum_1^k valence(i_j^p), i_j^p \in i_t, 1 \leq j \leq k, & i_t \in C_t \end{cases} \tag{8.1}$$

In the RL, the purpose or goal of the agent is formalized in terms of a special signal, the reward, obtaining from the environment to the agent. At each time step, the reward is a simple scalar number, $r_t \in \Re$. Informally, the agent's goal is to maximize the total amount of reward it receives. The classic calculation of expected return as describes as equations below. The *value* of a state s under a policy π, denoted as $v_\pi(s)$, is the expected return when starting in s and following π thereafter. The definition of v_π formally by

$$v_\pi(s) = \mathbb{E}_t \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s \right], for\, all\, s \in S, \tag{8.2}$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected return value of a random variable given that agent follows policy π, and t is any time step. Similarly, the expected return value of taking an action a in state s under a policy π, denoted $q_\pi(s, a)$, formally by

$$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,|S_t = s, A_t = a \right]. \tag{8.3}$$

function $q_\pi(s, a)$ is called the *action-value function for policy* π. As shown in equations of 8.2 and 8.3, the expected return value depends on the state of the environment and the terminal state, which indicates the specific goals in the RL. In this case, the agent's motivation comes from the environment that is external to the agent itself, which is said to be the *extrinsic motivation.*

The usage of the valence and the reward

Both the usage of valence in CCA and reward in RL participates in the decision-making mechanism for selecting potential intended interactions or actions, but in different ways. In CCA, the intended interaction selection is based on the term of *proclivity* ($p_i = w(a_i) \times v(post(a_i))$), which comes from the activated composite interaction 's weight ($w(a_i), a_i \in A_t$) and the valence of its post-interaction ($v(post(a_i)), a_i \in A_t$). The anticipation with the biggest proclivity value has a high probability of being enacted. The intended interaction of the selected anticipation's experiment could be a primitive interaction which means

the agent only needs to perform a single experiment, or it could be a series of primitive interaction for the agent to perform a sequence of experiments.

However, in the case of RL, the decision-making mechanism mainly focuses on selection an action to obtain the maximum cumulative return value starting from that action. Once the action is selected, agent immediately executes this action and then selects the action for the next interaction in the same way. Noted that each time the agent makes a decision, only a single action will be performed by the agent, rather than a sequence of actions. This is another obvious difference with the enaction of composite interaction in CCA.

8.2.3 The allocation strategy for the valence of primitive interactions

In this dissertation, the term *valence* is presented as the agent's innate behavioral preference, which endows an agent with an intrinsic motivation that spurs it to enact interactions with positive valence rather than interactions with negative valence. Hence the way to qualify this preferences for the initialization of various primitive interactions' valence directly effect the agent's performance in knowledge construction of the environment. At the same time, it is also obviously that different valence allocation strategies will have different effects.

Generally, based on our human commonsense, we will assume that the experience of the agent successfully moving forward one step is considered as a satisfactory experience and should have positive hints. While for experience like "collision" should be given a negative hint, unless the agent enjoys this experience, such as rock climbing. For experiences such as turning the current direction and touching the wall, they play an indirect and different role in the agent's commitment to achieve this preference, they will be given with different negative valences. According to this valence allocation strategy, the agent will be motivated to find ways to experience interaction of "moving forward successfully" and avoid "collisions". Obviously, in the process of valence allocation, it is inevitable that we will have more or less influence on the agent's learning process.

In addition, the valence allocation involves a combinatorial optimization problem [167], which is to find a set of optimal valences among various combination possibilities. In our experiment, we need to allocate five valences (or parameters) for the following five different types of experiences: (a) moving forward success, (b) bumping with the wall, (c)turning the direction, (d) touching with empty, and (e) touching with a wall. These valences act as a set of parameters $P = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$ for the function of agent's performance $l_\theta = L(\theta_1; \theta_2; \theta_3; \theta_4; \theta_5)$ in interacting with the environment. Our goal is to find an optimal combination of $p_i = \{\theta_1^*, \theta_2^*, \theta_3^*, \theta_4^*, \theta_5^*\}$ to get the minimum of bumping times $\min L(\theta^*)$.

One of the simplest and most intuitive solution is the Brute-force Algorithm [162, 163, 164], which is used to find an optimal set of parameters by traversing all the possibilities of combinations. In the case of small parameter value range, this method can easily obtain the globally optimal solution. While the cost of this approach can be very high when the parameter range is extended. Assuming that the given interval length of each parameter is n, the algorithm complexity will reach the power of n in the worst case, it is apparently inefficient.

The most classic algorithm is the Monte Carlo Algorithm [165, 166]. The general idea is to randomly cast one point in the parameter combination space, and then search a small group of points in its nearby area to get the locally optimal solution. Using the same way to randomly cast another point, and then find another locally optimal solution. This procedure is repeated until the termination conditions are met. This method greatly

reduces the complexity of the algorithm, but it needs to ensure that each cast point is in different area, otherwise the obtained point is just the locally optimal solution that has been searched before.

The combinatorial optimization problem and the integer programming problem are beyond the scope of this dissertation, but the solution of these problems can greatly improve the agent's performance in sense-making and perception (re)construction of the environment.

8.3 Existing problems

However, in the description of the introduced cognitive architecture, we point out many questions that remain to be addressed in moving towards more sophisticated agents confronted which couplings that offer more complex sequential regularities of interaction. In this current version, we acknowledged that CCA relies upon too many hard-coded function, which should be ultimately removed to the agent with more flexibility to scale up in more complex environments. In particular, some of these functions should be autonomously constructed from agent's interaction experience, which leave us rooms for even more constitutive autonomy.

In terms of scalability, we should notice that the number of new schemas constructed at each round would not grow with the environment complexity but with the agent's complexity, which remains under the modeler's control. The time needed to explore the environment would however grow with the environment complexity. This raises the interesting question of the agent's "education", that is, designing "pedagogical" situations where the agent could more easily learn lower-level schemas on which higher-level schemas could anchor.

With reports from the GAIT, the agent needs to retrospect all previous learned composite interactions to retrieve the ones whose pre-interactions are matched with the current enacted interaction in each decision cycle. As interactions continue, the recorded enacted interaction traces progressively grow longer, hence the agent will spend a long time to activated all eligible composite interactions for anticipations, increases the burden on the selection mechanism as well.

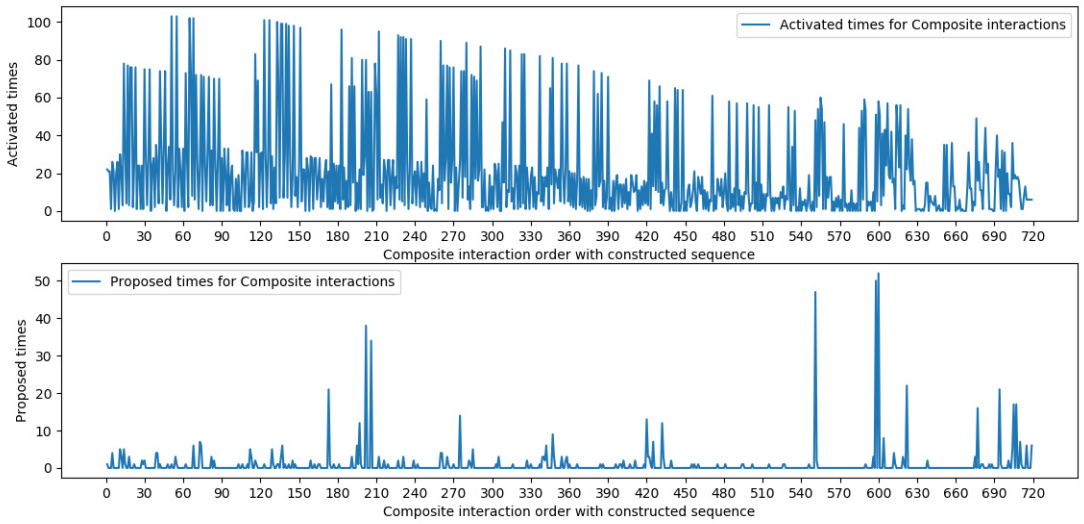


Figure 8.1: Utility rate of composite interaction.

In addition, the utility rate of composite interactions needs to be improved. As shown in Figure 8.1, the agent activates almost all composite interactions but only a few are proposed for enacting. Although the agent can be easily qualified for the work in the current setting scenario designed in this dissertation, the shortcoming will be easily revealed in conditions that the environment becomes more complex.

8.4 Future work and perspectives

Further work will be mainly focused on following aspects:

- Optimizing our model and upgrading the toolkit. For example, a experience-based predictive model could be used in the selection mechanism for better proposing anticipations for the agent to interact with the environment. With memorizing patterns that could improve the learning efficiency and eliminating composite interactions that probably will not use to simplify the activation and proposition processes in BEL-CA in the future.

- Implementing higher-level abstraction mechanisms in our agent and examining it in more complex environments. An abstraction of the interaction regularities endows the agent has capabilities to adapt to different interaction scenarios and generate behaviors flexibly.

- Acquiring the capability of exploration. The learned structure behaviors which allow the agent to generate proper behaviors in the environment that it has been familiar with. However, in changed environments, the agent persists in following the generated behaviors in each interaction prevents it to explore the unfamiliar parts in the environment. A dynamic exploring mechanism could spur the agent to acquire new regularities of interaction afforded by the environment.

- And also, in the enaction of composite interactions, we could flat composite interactions into sequences of primitive interactions without referring to its structure. In addition, we'd like to combine with this hierarchical sequential learning model to evaluate the performance of the agent in a multi-agent scenario, which provides more challenges and opportunities to improve the agent's learning ability in complex environment and dynamic situations.

Bibliography

[1] Nick Haber, Damian Mrowca, Li Fei-Fei, and Daniel LK Yamins. Emergence of structured behaviors from curiosity-based intrinsic motivation. *arXiv preprint arXiv:1802.07461*, 2018.

[2] Louise Goupil, Margaux Romand-Monnier, and Sid Kouider. Infants ask for help when they know they don't know. *Proceedings of the National Academy of Sciences*, 113(13):3492–3496, 2016.

[3] Katarina Begus, Teodora Gliga, and Victoria Southgate. Infants learn what they want to learn: Responding to infant pointing leads to superior learning. *PloS one*, 9(10):e108817, 2014.

[4] Alison Gopnik, Andrew N Meltzoff, and Patricia K Kuhl. *The scientist in the crib: Minds, brains, and how children learn.* William Morrow & Co, 1999.

[5] Robert L Fantz. Visual experience in infants: Decreased attention to familiar patterns relative to novel ones. *Science*, 146(3644):668–670, 1964.

[6] Evgenii Nikolaevich Sokolov. Perception and the conditioned reflex. 1963.

[7] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[8] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, volume 13, page 05, 2013.

[9] Olivier L Georgeon and Frank E Ritter. An intrinsically-motivated schema mechanism to model and simulate emergent cognition. *Cognitive Systems Research*, 15:73–92, 2012.

[10] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

[11] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[12] Maxime Guériau, Nicolás Cardozo, and Ivana Dusparic. Constructivist approach to state space adaptation in reinforcement learning. *Learning*, 4(S3):S2, 2019.

[13] Maxime Guériau, Frédéric Armetta, Salima Hassas, Romain Billot, and Nour-Eddin El Faouzi. A constructivist approach for a self-adaptive decision-making system: application to road traffic control. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 670–677. IEEE, 2016.

[14] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[15] Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. In *Advances in Neural Information Processing Systems*, pages 8388–8399, 2018.

[16] Pierre-Yves Oudeyer and Linda B Smith. How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, 8(2):492–502, 2016.

[17] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[18] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[19] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[20] Katherine E Twomey and Gert Westermann. Curiosity-based learning in infants: a neurocomputational approach. *Developmental science*, 21(4):e12629, 2018.

[21] Stefan Stojanov, Samarth Mishra, Ngoc Anh Thai, Nikhil Dhanda, Ahmad Humayun, Chen Yu, Linda B Smith, and James M Rehg. Incremental object learning from contiguous views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8777–8786, 2019.

[22] Karinna B Hurley and Lisa M Oakes. Experience and distribution of attention: Pet exposure and infants' scanning of animal images. *Journal of Cognition and Development*, 16(1):11–30, 2015.

[23] Alexander Riegler. The radical constructivist dynamics of cognition. *The mind, the body and the world: Psychology after cognitivism*, pages 91–115, 2007.

[24] Jianyong Xue, Olivier L Georgeon, and Mathieu Gillermin. Causality reconstruction by an autonomous agent. In *Biologically Inspired Cognitive Architectures Meeting*, pages 347–354. Springer, 2018.

[25] Harold Henry Chaput and Leslie B Cohen. A model of infant causal perception and its development. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society, ed. JD Moore & K. Stenning*, pages 182–87, 2001.

[26] Olivier L Georgeon, Frank E Ritter, and Steven R Haynes. Modeling bottom-up learning from activity in soar. In *18th Annual Conference on Behavior Representation in Modeling and Simulation (BRIMS), Sundance, Utah*, 2009.

[27] Andrei Marinescu, Ivana Dusparic, and Siobhán Clarke. Prediction-based multi-agent reinforcement learning in inherently non-stationary environments. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(2):1–23, 2017.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[29] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[30] Xiao Huang and John Weng. Novelty and reinforcement learning in the value system of developmental robots. 2002.

[31] Adrien Baranes and Pierre-Yves Oudeyer. Robust intrinsically motivated exploration and active learning. In *2009 IEEE 8th International Conference on Development and Learning*, pages 1–6. IEEE, 2009.

[32] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[33] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

[34] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

[35] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection science*, 15(4):151–190, 2003.

[36] Olivier L Georgeon, Mark A Cohen, and Amélie V Cordier. A model and simulation of early-stage vision as a developmental sensorimotor process. In *Artificial Intelligence Applications and Innovations*, pages 11–16. Springer, 2011.

[37] Chuang Gan, Xiaoyu Chen, Phillip Isola, Antonio Torralba, and Joshua B Tenenbaum. Noisy agents: Self-supervised exploration by predicting auditory events. *arXiv preprint arXiv:2007.13729*, 2020.

[38] Kuno Kim, Megumi Sano, Julian De Freitas, Nick Haber, and Daniel Yamins. Active world model learning with progress curiosity. *arXiv preprint arXiv:2007.07853*, 2020.

[39] Jacqueline Gottlieb, Pierre-Yves Oudeyer, Manuel Lopes, and Adrien Baranes. Information-seeking, curiosity, and attention: computational and neural mechanisms. *Trends in cognitive sciences*, 17(11):585–593, 2013.

[40] Goren Gordon. Infant-inspired intrinsically motivated curious robots. *Current Opinion in Behavioral Sciences*, 35:28–34, 2020.

[41] Michael John Lingelbach, Damian Mrowca, Nick Haber, Li Fei-Fei, and Daniel LK Yamins. Towards curiosity-driven learning of physi-cal dynamics.

[42] Rafik Hadfi. Investigating enactive learning for autonomous intelligent agents. *arXiv preprint arXiv:1810.04535*, 2018.

[43] Tom Froese and Tom Ziemke. Enactive artificial intelligence: Investigating the systemic organization of life and mind. *Artificial Intelligence*, 173(3-4):466–500, 2009.

[44] Olivier L Georgeon, James B Marshall, and Riccardo Manzotti. Eca: An enactivist cognitive architecture based on sensorimotor modeling. *Biologically Inspired Cognitive Architectures*, 6:46–57, 2013.

[45] Olivier L Georgeon, Christian Wolf, and Simon Gay. An enactive approach to autonomous agent and robot learning. In *2013 IEEE Third Joint International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, pages 1–6. IEEE, 2013.

[46] Ilya E Monosov. How outcome uncertainty mediates attention, learning, and decision-making. *Trends in Neurosciences*, 2020.

[47] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[48] Allison Bruce, Illah Nourbakhsh, and Reid Simmons. The role of expressiveness and attention in human-robot interaction. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 4138–4142. IEEE, 2002.

[49] Ahmed Hussain Qureshi, Yutaka Nakamura, Yuichiro Yoshikawa, and Hiroshi Ishiguro. Show, attend and interact: Perceivable human-robot social interaction through neural attention q-network. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1639–1645. IEEE, 2017.

[50] Jean Piaget. *The construction of reality in the child*, volume 82. Routledge, 2013.

[51] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[52] Fei Xu. Towards a rational constructivist theory of cognitive development. *Psychological review*, 126(6):841, 2019.

[53] Olivier L Georgeon, James B Marshall, and Simon Gay. Interactional motivation in artificial systems: Between extrinsic and intrinsic motivation. In *2012 IEEE international conference on development and learning and epigenetic robotics (ICDL)*, pages 1–2. IEEE, 2012.

[54] J Kevin O'Regan and Alva Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939, 2001.

[55] Etienne B Roesch, Matthew Spencer, Slawomir J Nasuto, Thomas Tanay, and J Mark Bishop. Exploration of the functional properties of interaction: Computer models and pointers for theory. *Constructivist Foundations*, 9(1), 2013.

[56] Bill N Schilit and Marvin M Theimer. Disseminating active map information to mobile hosts. *IEEE network*, 8(5):22–32, 1994.

[57] Peter J Brown, John D Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE personal communications*, 4(5):58–64, 1997.

[58] Anind K Dey. Context-aware computing: The cyberdesk project. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*, pages 51–54, 1998.

[59] Richard Hull, Philip Neaves, and James Bedford-Roberts. Towards situated computing. In *Wearable Computers, 1997. Digest of Papers., First International Symposium on*, pages 146–153. IEEE, 1997.

[60] AK Dey, GD Abowd, and A CyberDesk Wood. A framework for providing self-integrating context-aware services proceedings of the. In *International Conference on Intelligent User Interfaces (IUI'98)*, pages 47–54.

[61] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 92–99. IEEE, 1998.

[62] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[63] Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. Building expert system. 1983.

[64] John Durkin. 4 expert system. *The Handbook of Applied Expert Systems*, 4:15, 1997.

[65] Shu-Hsien Liao. Expert system methodologies and applications—a decade review from 1995 to 2004. *Expert systems with applications*, 28(1):93–103, 2005.

[66] Lewis P Lipsitt. Learning in infancy: cognitive development in babies. *The Journal of pediatrics*, 109(1):172–182, 1986.

[67] John B Watson. Psychology as the behaviorist views it. *Psychological review*, 101(2):248, 1994.

[68] Albert Globus and Arnold B Scheibel. The effect of visual deprivation on cortical neurons: a golgi study. *Experimental neurology*, 19(3):331–345, 1967.

[69] Shawn Schapiro and Katherine R Vukovich. Early experience effects upon cortical dendrites: a proposed model for development. *Science*, 167(3916):292–294, 1970.

[70] David H Hubel and Torsten N Wiesel. The period of susceptibility to the physiological effects of unilateral eye closure in kittens. *The Journal of physiology*, 206(2):419–436, 1970.

[71] William Huitt and John Hummel. Piaget's theory of cognitive development. *Educational psychology interactive*, 3(2):1–5, 2003.

[72] Frank Guerin. Constructivism in ai: Prospects, progress and challenges. In *AISB Convention*, pages 20–27, 2008.

[73] Robert J Vallerand. Toward a hierarchical model of intrinsic and extrinsic motivation. In *Advances in experimental social psychology*, volume 29, pages 271–360. Elsevier, 1997.

[74] Gary L Drescher. *Made-up minds: a constructivist approach to artificial intelligence*. MIT press, 1991.

[75] Paul R Cohen, Marc S Atkin, Tim Oates, and Carole R Beal. Neo: Learning conceptual knowledge by sensorimotor interaction with an environment. In *Proceedings of the first international conference on Autonomous agents*, pages 170–177, 1997.

[76] Harold Henry Chaput. Post-piagetian constructivism for grounded knowledge acquisition. In *Proceedings from the AAAI Spring Symposium on Grounded Knoweldge*, 2001.

[77] Leslie B Cohen, Geoffrey Amsel, Melissa A Redford, and Marianella Casasola. The development of infant causal perception. *Perceptual development: Visual, auditory, and speech perception in infancy*, pages 167–209, 1998.

[78] Leslie B Cohen. An information-processing approach to infant perception and cognition. *The development of sensory, motor, and cognitive capacities in early infancy*, pages 277–300, 1998.

[79] Leslie B Cohen, Harold H Chaput, and Cara H Cashon. A constructivist model of infant cognition. *Cognitive Development*, 17(3-4):1323–1343, 2002.

[80] Leslie B Cohen and Cara H Cashon. Infant object segregation implies information integration. *Journal of experimental child psychology*, 78(1):75–83, 2001.

[81] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 545–551. IEEE, 2017.

[82] Celeste Kidd, Steven T Piantadosi, and Richard N Aslin. The goldilocks effect: Human infants allocate attention to visual sequences that are neither too simple nor too complex. *PloS one*, 7(5):e36399, 2012.

[83] Ernst Von Glasersfeld. Thirty years constructivism. 2005.

[84] Olivier L Georgeon, Jonathan H Morgan, and Frank E Ritter. An algorithm for self-motivated hierarchical sequence learning. In *Proceedings of the International Conference on Cognitive Modeling. Philadelphia, PA. ICCM-164*, pages 73–78. Citeseer, 2010.

[85] Harold Henry Chaput. *The constructivist learning architecture: A model of cognitive development for robust autonomous robots*. PhD thesis, 2004.

[86] Aimee E Stahl and Lisa Feigenson. Observing the unexpected enhances infants' learning and exploration. *Science*, 348(6230):91–94, 2015.

[87] Doug Blank, Joshua M Lewis, and James B Marshall. The multiple roles of anticipation in developmental robotics. In *AAAI Fall Symposium Workshop Notes, From Reactive to Anticipatory Cognitive Embodied Systems*, 2005.

[88] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

[89] David W Aha. Feature weighting for lazy learning algorithms. In *Feature extraction, construction and selection*, pages 13–32. Springer, 1998.

[90] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[91] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[92] Simon L Gay, Olivier L Georgeon, and Christian Wolf. Autonomous object modeling based on affordances for spatial organization of behavior. In *4th International Conference on Development and Learning and on Epigenetic Robotics*, pages 87–92. IEEE, 2014.

[93] John E Laird and Clare Bates Congdon. The soar user's manual version 9.5. 0. *The Regents of the University of Michigan*, 2015.

[94] John R Anderson and Christian J Lebiere. *The atomic components of thought*. Psychology Press, 2014.

[95] David E Kieras and David E Meyer. The epic architecture: Principles of operation. *Unpublished manuscript from ftp://ftp. eecs. umich. edu/people/kieras/EPICarch. ps*, 1996.

[96] Joscha Bach. The micropsi agent architecture. In *Proceedings of ICCM-5, international conference on cognitive modeling, Bamberg, Germany*, pages 15–20. Citeseer, 2003.

[97] Ron Sun, Todd Peterson, and Edward Merrill. A hybrid architecture for situated learning of reactive sequential decision making. *Applied Intelligence*, 11(1):109–127, 1999.

[98] Pat Langley and Dongkyu Choi. Learning recursive control programs from problem solving. *Journal of Machine Learning Research*, 7(Mar):493–518, 2006.

[99] Chien Van Dang, Tin Trung Tran, Ki-Jong Gil, Yong-Bin Shin, Jae-Won Choi, Geon-Soo Park, and Jong-Wook Kim. Application of soar cognitive agent based on utilitarian ethics theory for home service robots. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 155–158. IEEE, 2017.

[100] Bryan Stearns, John Laird, and Mazin Assanie. Applying primitive elements theory for procedural transfer in soar. In *Proceedings of the 15th international conference on cognitive modeling*, 2017.

[101] Nicholas M DiFilippo. Framework for the automated disassembly of electronic waste using the soar cognitive architecture. 2016.

[102] John E Laird and Shiwali Mohan. A case study of knowledge integration across multiple memories in soar. *Biologically Inspired Cognitive Architectures*, 8:93–99, 2014.

[103] Nate Derbinsky and John E Laird. Competence-preserving retention of learned knowledge in soar's working and procedural memories. In *Proceedings of the 11th international conference on cognitive modeling*, pages 205–210, 2012.

[104] John Edwin Laird, Keegan R Kinkade, Shiwali Mohan, and Joseph Z Xu. Cognitive robotics using the soar cognitive architecture. In *CogRob@ AAAI*. Citeseer, 2012.

[105] Usef Faghihi and Stan Franklin. The lida model as a foundational architecture for agi. In *Theoretical Foundations of Artificial General Intelligence*, pages 103–121. Springer, 2012.

[106] John E Laird. *The Soar cognitive architecture*. MIT press, 2012.

[107] Teuvo Kohonen. *Self-organizing maps*, volume 30. Springer Science & Business Media, 2012.

[108] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[109] Luc Steels. The autotelic principle. In *Embodied artificial intelligence*, pages 231–242. Springer, 2004.

[110] Alisha Moreland-Capuia. The developing brain and trauma. In *Training for Change*, pages 1–31. Springer, 2019.

[111] J Piaget. The origins of intelligence in children (m. cook, trans.). new york, ny, us, 1952.

[112] Barry J Wadsworth. *Piaget's theory of cognitive and affective development: Foundations of constructivism*. Longman Publishing, 1996.

[113] Kathleen Stassen Berger. *The developing person through the life span*, volume 41. Worth Publishers New York, NY, 2014.

[114] S McLeod. Jean piaget cognitive theory simply psychology. simply psychology, 2012.

[115] S McLeod. Jean piaget— cognitive theory— simply psychology.[online] simplypsychology. org, 2009.

[116] Jack Block. Assimilation, accommodation, and the dynamics of personality development. *Child development*, pages 281–295, 1982.

[117] Ana A Sobel, Patricia A Resick, and Aline E Rabalais. The effect of cognitive processing therapy on cognitions: Impact statement coding. *Journal of Traumatic Stress: Official Publication of The International Society for Traumatic Stress Studies*, 22(3):205–211, 2009.

[118] Saul McLeod. Jean piaget's theory of cognitive development. *Simply psychology*, pages 1–9, 2018.

[119] Ernst Von Glasersfeld. An introduction to radical constructivism. *The invented reality*, 1740, 1984.

[120] Ernst von Glasersfeld. An introduction to radical constructivism. *AntiMatters*, 2(3):5–20, 2008.

[121] Susan L Hurley. *Consciousness in action*. Harvard University Press, 2002.

[122] Rodney A Brooks. New approaches to robotics. *Science*, 253(5025):1227–1232, 1991.

[123] Naoya Hirose. An ecological approach to embodiment and cognition. *Cognitive Systems Research*, 3(3):289–299, 2002.

[124] Murray Shanahan. *Embodiment and the inner life: Cognition and Consciousness in the Space of Possible Minds*. Oxford University Press, USA, 2010.

[125] Tom Ziemke. The construction of 'reality'in the robot: Constructivist perspectives on situated artificial intelligence and adaptive robotics. *Foundations of science*, 6(1-3):163–233, 2001.

[126] Michael L Anderson. Embodied cognition: A field guide. *Artificial intelligence*, 149(1):91–130, 2003.

[127] Owen Holland. The future of embodied artificial intelligence: Machine consciousness? In *Embodied artificial intelligence*, pages 37–53. Springer, 2004.

[128] Anthony Chemero and Michael T Turvey. Gibsonian affordances for roboticists. *Adaptive Behavior*, 15(4):473–480, 2007.

[129] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[130] J Kevin O'Regan. How to build a robot that is conscious and feels. *Minds and Machines*, 22(2):117–136, 2012.

[131] Chandana Paul. Morphological computation: A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54(8):619–630, 2006.

[132] Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[133] Inman Harvey, Ezequiel Di Paolo, Rachel Wood, Matt Quinn, and Elio Tuci. Evolutionary robotics: A new scientific tool for studying cognition. *Artificial life*, 11(1-2):79–98, 2005.

[134] Stefano Nolfi, Dario Floreano, and Director Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press, 2000.

[135] Luc Berthouze and Giorgio Metta. Epigenetic robotics: modelling cognitive development in robotic systems, 2005.

[136] Jordan Zlatev. The epigenesis of meaning in human beings, and possibly in robots. *Minds and machines*, 11(2):155–195, 2001.

[137] Jean Piaget. *The psychology of intelligence*. Routledge, 2003.

[138] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[139] George E Monahan. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16, 1982.

[140] Roland Benabou and Jean Tirole. Intrinsic and extrinsic motivation. *The review of economic studies*, 70(3):489–520, 2003.

[141] Filipo Studzinski Perotto, Jean-Christophe Buisson, and Luís Otávio Campos Alvares. Constructivist anticipatory learning mechanism (calm): Dealing with partially deterministic and partially observable environments. 2007.

[142] Michael Miller. Building minds with patterns. *ISBN-10: 0-692-54140-1*, 2016.

[143] Kristinn R Thórisson. Seed-programmed autonomous general learning. In *International Workshop on Self-Supervised Learning*, pages 32–61. PMLR, 2020.

[144] Brett Martensen. The perception–action hierarchy and its implementation using binons (binary neurons). *Procedia Computer Science*, 169:489–500, 2020.

[145] Olivier L Georgeon and James B Marshall. The small loop problem: A challenge for artificial emergent cognition. In *Biologically Inspired Cognitive Architectures 2012*, pages 137–144. Springer, 2013.

[146] Brandon Rohrer. Accelerating progress in artificial general intelligence: Choosing a benchmark for natural world interaction. *Journal of Artificial General Intelligence*, 2(1):1–28, 2010.

[147] Olivier L Georgeon. Little ai: Playing a constructivist robot. *SoftwareX*, 6:161–164, 2017.

[148] Olivier L Georgeon and Amélie Cordier. Inverting the interaction cycle to model embodied agents. In *BICA*, pages 243–248, 2014.

[149] Rolf Pfeifer and Christian Scheier. *From perception to action: The right direction?* IEEE, 1994.

[150] Olivier L Georgeon and Mathieu Guillermin. Mastering the laws of feedback contingencies is essential to constructivist artificial agents. 2018.

[151] Matthew E Roser, Jonathan A Fugelsang, Kevin N Dunbar, Paul M Corballis, and Michael S Gazzaniga. Dissociating processes supporting causal perception and causal inference in the brain. *Neuropsychology*, 19(5):591, 2005.

[152] Laura E Schulz and Elizabeth Baraff Bonawitz. Serious fun: preschoolers engage in more exploratory play when evidence is confounded. *Developmental psychology*, 43(4):1045, 2007.

[153] Suraj Nair, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Causal induction from visual observations for goal directed tasks. *arXiv preprint arXiv:1910.01751*, 2019.

[154] Bob Rehder. Categorization as causal reasoning. *Cognitive Science*, 27(5):709–748, 2003.

[155] Roberta Corrigan and Peggy Denton. Causal understanding as a developmental primitive. *Developmental review*, 16(2):162–202, 1996.

[156] Scott H Johnson-Frey. What's so special about human tool use? *Neuron*, 39(2):201–204, 2003.

[157] John McClure. Discounting causes of behavior: Are two reasons better than one? *Journal of Personality and Social Psychology*, 74(1):7, 1998.

[158] Albert Michotte. *The perception of causality*, volume 21. Routledge, 2017.

[159] Alan M Leslie. Spatiotemporal continuity and the perception of causality in infants. *Perception*, 13(3):287–305, 1984.

[160] Leslie B Cohen and Geoffrey Amsel. Precursors to infants' perception of the causality of a simple event. *Infant behavior and development*, 21(4):713–731, 1998.

[161] Olivier L Georgeon, Florian J Bernard, and Amélie Cordier. Constructing phenomenal knowledge in an unknown noumenal reality. *Procedia Computer Science*, 71:11–16, 2015.

[162] Jimmy Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 155–162, 2009.

[163] Michael R Fellows, Fedor V Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012.

[164] Daniel J Bernstein. Understanding brute force. In *Workshop Record of ECRYPT STVL Workshop on Symmetric Key Encryption, eSTREAM report*, volume 36, page 2005. Citeseer, 2005.

[165] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[166] Nasser R Sabar and Graham Kendall. Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences*, 314:225–239, 2015.

[167] Walter J Gutjahr. A converging aco algorithm for stochastic combinatorial optimization. In *International Symposium on Stochastic Algorithms*, pages 10–25. Springer, 2003.

[168] Alain Mille. From case-based reasoning to traces-based reasoning. *Annual Reviews in Control*, 30(2):223–232, 2006.

[169] Jeff Hawkins and Sandra Blakeslee. *On intelligence: How a new understanding of the brain will lead to the creation of truly intelligent machines*. Macmillan, 2007.

[170] William James. Pragmatism: A new name for old ways of thinking. *New York*, 1907.

[171] Karl R Popper. *Objective knowledge*, volume 360. Oxford University Press Oxford, 1972.

[172] Jean Piaget. L'épistémologie génétique. 1970.

[173] Sandra Clara Gadanho and John Hallam. *Exploring the role of emotions in autonomous robot learning*. Department of Artificial Intelligence, University of Edinburgh, 1998.

[174] Ron Sun. Motivational representations within a computational cognitive architecture. *Cognitive Computation*, 1(1):91–103, 2009.

[175] Olivier L Georgeon and James B Marshall. Demonstrating sensemaking emergence in artificial agents: A method and an example. *International Journal of Machine Consciousness*, 5(02):131–144, 2013.

Appendix A

Causality reconstruction

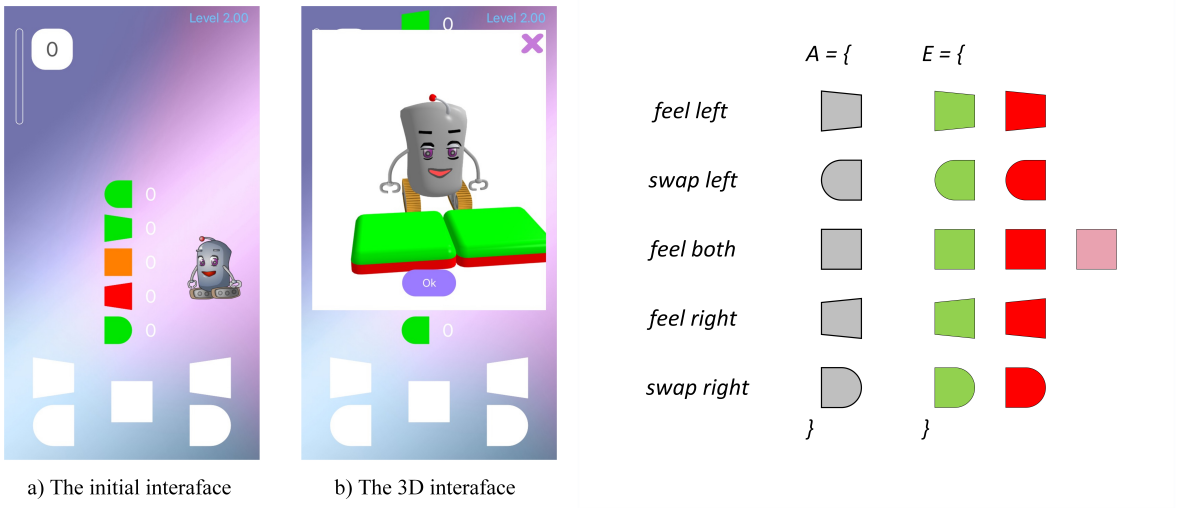
A.1 The interaction scenario

In the Little AI of Level2.00, the system initially provides five commands with same shape and same color. The effects of these five commands and the structure of the environment are unknown to the agent and the player as well [161], So we need the players to construct the knowledge gradually by their interactions with the agent and its environment. For the sake of demonstration, we intentionally place the reader in the same situation as the system: initially ignorant of the meaning of experiences.

The self-motivated agent has five innate actions and identified by different numbers which are $A = \{feelleft = 1, swapleft = 2, feelboth = 3, feelright = 4, swapright = 5\}$. As interaction starts, the agent selects an action amongst the set of actions A, then it simultaneously receives the feedback from the environment and automatically forms an experience combined with this action. In particular, action *feel left* and *feel right* touch the environment from two different sides and receive their corresponding feedback of the environment. There exits two different results: true (marked 1) or false (marked 0), action feel both touch the left and the right at the same time, it thus has three different result: left and right both are false(0,0), left and right only one are true(0,1 or 1,0), and left and right both are true(1,1). In addition, actions of *swap left* and *swap right* can change the environment. Therefore, the agent has eleven different kinds of experiences which are based on corresponding actions and the state of situated environment. When the agent experienced several experiences, then the environment will give the response result of the actions. In order to present these five actions more clearly, we use different icons and colors to identify five actions and eleven experiences respectively (as shown in Figure A.1(b)). With different situations the agent faced with and the changeable environment the agent situated in, we use three colors (red means false, green means true and pink only used for feel both means only one is true) combined with five actions to illustrate these eleven experiences.

A.2 The complete two-step regularities afforded by the environment.

In section 5.2.2 we presented 12 partial two-step regularities afforded by the environment, here we present the all fully two step regularities afforded by the environment (as shown in Figure A.2).



(a) Little AI interface.

(b) Five actions and eleven experiences.

Figure A.1: The Little AI interface and five actions with their eleven experiences.

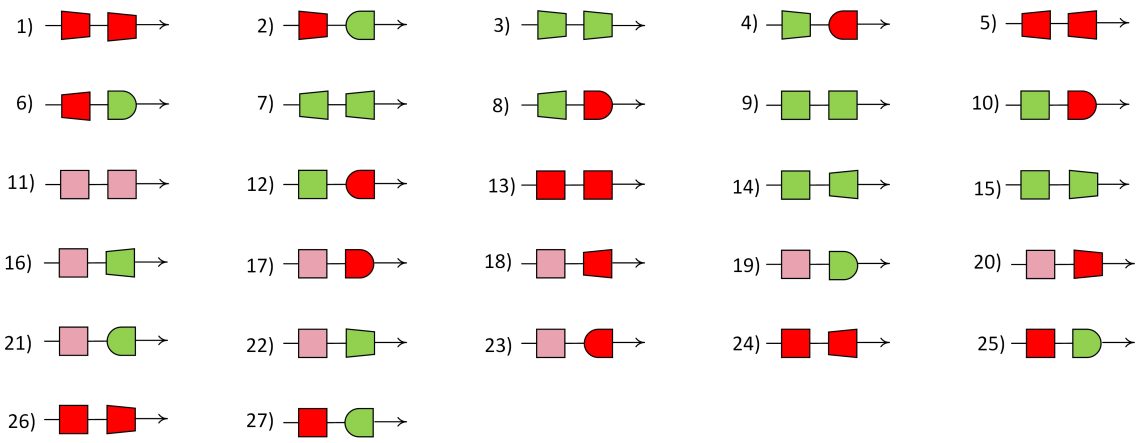
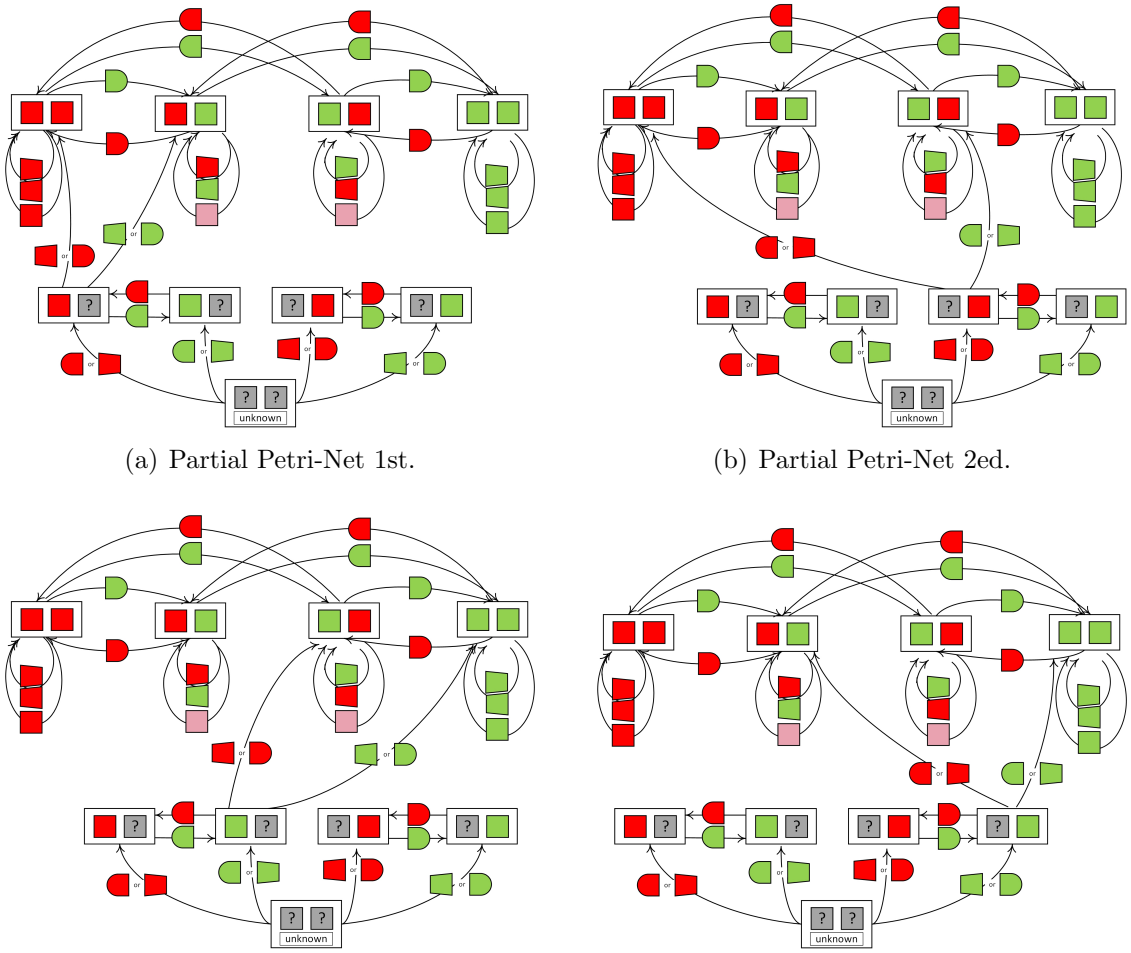


Figure A.2: The complete two-step regularities afforded by the environment.

A.3 The full structure of Petri-Net in Little_AI of Level2.00

As we introduced the partial structure of Petri-Net in the section 5.2.3, here we present the complete structure of the Petri-Net as shown in Figure A.3.



(a) Partial Petri-Net 1st.

(b) Partial Petri-Net 2ed.

(c) Partial Petri-Net 3rd.

(d) Partial Petri-Net 4th.

Figure A.3: The complete structure of the Petri-Net.

Appendix B

Interaction traces

In the interaction traces, a set of experiments $E = \{e0, e1, e2, e3, e4, e5\}$ which represents *moving forward,turn left, turn right, touch the front side, touch the left side, touch the right side* respectively. Combined with experiments and their possible feedback from interacting with the environment, a set of primitive interactions I is formed as shown in Figure B.1. The description of the interaction follows the sequence of its label, valence, weight, and its level.

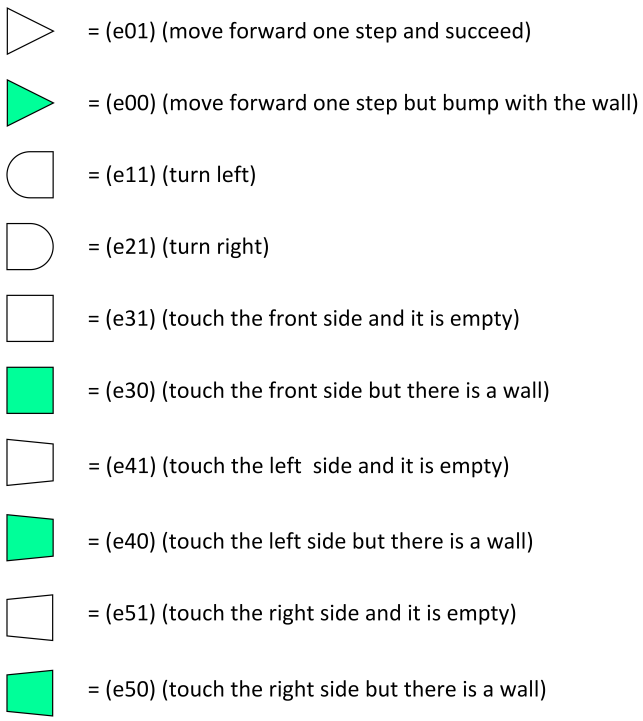


Figure B.1: The set of primitive interactions which are combined with experiments and their possible feedback from interaction with the environment.

```
   The valences are allocated as:
{moveSucess:5, moveFailture:-10, turn:-3, feelEmpty:-1, feelWall:-2}

   The interaction is starting as follows...
----------------------------------------
```

```
loop number is:1
intendedInteraction is:  e50,-2,0,0
enactedInteraction is:  e50,-2,0,0
learnCompositeInteraction()
The intended interaction is:  e50,-2,0,0
The enacted interaction is:  e50,-2,0,0
---------------------------------------
loop number is:2
The primitive intendedInteraction is:  e50,-2,0,0
The primitive enactedInteraction is:  e50,-2,0,0
learnCompositeInteraction()
LEARN:(e50e50),-4,1,1
---------------------------------------
loop number is:3
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e00,-10,0,0
learnCompositeInteraction()
LEARN:(e50e00),-12,1,1
LEARN:(e50(e50e00)),-14,1,2
LEARN:((e50e50)e00),-14,1,2
---------------------------------------
...
---------------------------------------
loop number is:8
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
learnCompositeInteraction()
REINFORCE:(e01e01),10,2,1
LEARN:(e01(e01e01)),15,1,2
LEARN:((e01e01)e01),15,1,2
---------------------------------------
loop number is:9
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e00,-10,0,0
learnCompositeInteraction()
LEARN:(e01e00),-5,1,1
LEARN:(e01(e01e00)),0,1,2
LEARN:((e01e01)e00),0,1,2
---------------------------------------
loop number is:10
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e00,-10,0,0
learnCompositeInteraction()
LEARN:(e00e00),-20,1,1
LEARN:(e01(e00e00)),-15,1,2
LEARN:((e01e00)e00),-15,1,2
---------------------------------------
...
---------------------------------------
loop number is:23
The best choice of the anticipation interaction is composite and its weight
is:1
```

```
The best choice of the anticipation interaction's weight less or equal
than threshold and proclivity is positive
The primitive intendedInteraction is:  e11,-3,0,0
The primitive enactedInteraction is:  e11,-3,0,0
learnCompositeInteraction()
REINFORCE:(e00e11),-13,2,1
LEARN:(e41(e00e11)),-14,1,2
LEARN:((e41e00)e11),-14,1,2
---------------------------------------
loop number is:24
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
learnCompositeInteraction()
REINFORCE:(e11e01),2,3,1
REINFORCE:(e00(e11e01)),-8,2,2
REINFORCE:((e00e11)e01),-8,2,2
---------------------------------------
loop number is:25
The primitive intendedInteraction is:  e30,-2,0,0
The primitive enactedInteraction is:  e30,-2,0,0
learnCompositeInteraction()
LEARN:(e01e30),3,1,1
LEARN:(e11(e01e30)),0,1,2
LEARN:((e11e01)e30),0,1,2
---------------------------------------
...
---------------------------------------
loop number is:38
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e11,-3,0,0
The primitive enactedInteraction is:  e11,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e00,-10,0,0
REINFORCE:(e11e00),-13,3,1
The intended composite interaction is:  (e11e01),2,5,1
The enacted composite interaction is:  (e11e00),-13,3,1
learnCompositeInteraction()
LEARN:(e00(e11e00)),-23,1,2
LEARN:(e11(e00(e11e00))),-26,1,3
LEARN:((e11e00)(e11e00)),-26,1,2
---------------------------------------
loop number is:39
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e50,-2,0,0
The primitive enactedInteraction is:  e50,-2,0,0
learnCompositeInteraction()
LEARN:((e11e00)e50),-15,1,2
LEARN:(e00((e11e00)e50)),-25,1,3
LEARN:((e00(e11e00))e50),-25,1,3
---------------------------------------
```

```
loop number is:40
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
learnCompositeInteraction()
LEARN:(e50e21),-5,1,1
LEARN:((e11e00)(e50e21)),-18,1,2
LEARN:(((e11e00)e50)e21),-18,1,3
----------------------------------------
...
----------------------------------------
loop number is:80
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e31e01),4,3,1
The enacted composite interaction is:  (e31e01),4,3,1
learnCompositeInteraction()
REINFORCE:(e11(e31e01)),1,3,2
LEARN:(e41(e11(e31e01))),0,1,3
LEARN:((e41e11)(e31e01)),0,1,2
----------------------------------------
loop number is:81
The best choice of the anticipation interaction is composite and its weight
is:1
The best choice of the anticipation interaction's weight less than threshold
or proclivity is negative
The primitive intendedInteraction is:  e30,-2,0,0
The primitive enactedInteraction is:  e31,-1,0,0
learnCompositeInteraction()
LEARN:((e31e01)e31),3,1,2
LEARN:(e11((e31e01)e31)),0,1,3
LEARN:((e11(e31e01))e31),0,1,3
----------------------------------------
loop number is:82
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
learnCompositeInteraction()
REINFORCE:(e31e01),4,4,1
LEARN:((e31e01)(e31e01)),8,1,2
LEARN:(((e31e01)e31)e01),8,1,3
----------------------------------------
...
----------------------------------------
loop number is:85
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
```

```
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e30,-2,0,0
The intended composite interaction is:  (e31e01),4,5,1
The enacted composite interaction is:  e30,-2,0,0
learnCompositeInteraction()
REINFORCE:(e01e30),3,3,1
LEARN:(e31(e01e30)),2,1,2
LEARN:((e31e01)e30),2,1,2
---------------------------------------
loop number is:86
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e50,-2,0,0
The primitive enactedInteraction is:  e50,-2,0,0
learnCompositeInteraction()
REINFORCE:(e30e50),-4,2,1
LEARN:(e01(e30e50)),1,1,2
LEARN:((e01e30)e50),1,1,2
---------------------------------------
loop number is:87
The best choice of the anticipation interaction is composite and its weight
is:1
The best choice of the anticipation interaction's weight less or equal
than threshold and proclivity is positive
The primitive intendedInteraction is:  e40,-2,0,0
The primitive enactedInteraction is:  e41,-1,0,0
learnCompositeInteraction()
REINFORCE:(e50e41),-3,2,1
LEARN:(e30(e50e41)),-5,1,2
LEARN:((e30e50)e41),-5,1,2
---------------------------------------
...
---------------------------------------
loop number is:106
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e40,-2,0,0
The primitive enactedInteraction is:  e40,-2,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e40e01),3,4,1
The enacted composite interaction is:  (e40e01),3,4,1
learnCompositeInteraction()
REINFORCE:(e21(e40e01)),0,3,2
LEARN:(e00(e21(e40e01))),-10,1,3
LEARN:((e00e21)(e40e01)),-10,1,2
---------------------------------------
loop number is:107
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e11,-3,0,0
The primitive enactedInteraction is:  e11,-3,0,0
learnCompositeInteraction()
LEARN:((e40e01)e11),0,1,2
```

```
LEARN:(e21((e40e01)e11)),-3,1,3
LEARN:((e21(e40e01))e11),-3,1,3
--------------------------------------
loop number is:108
The best choice of the anticipation interaction is composite and its weight
is:3
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e30,-2,0,0
The intended composite interaction is:  (e31e01),4,5,1
The enacted composite interaction is:  e30,-2,0,0
learnCompositeInteraction()
REINFORCE:(e11e30),-5,2,1
LEARN:((e40e01)(e11e30)),-2,1,2
LEARN:(((e40e01)e11)e30),-2,1,3
--------------------------------------
...
--------------------------------------
loop number is:110
The best choice of the anticipation interaction is composite and its weight
is:3
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e31e01),4,5,1
The enacted composite interaction is:  (e31e01),4,5,1
learnCompositeInteraction()
REINFORCE:(e11(e31e01)),1,4,2
LEARN:(e30(e11(e31e01))),-1,1,3
LEARN:((e30e11)(e31e01)),-1,1,2
--------------------------------------
loop number is:111
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e11,-3,0,0
The primitive enactedInteraction is:  e11,-3,0,0
learnCompositeInteraction()
LEARN:((e31e01)e11),1,1,2
LEARN:(e11((e31e01)e11)),-2,1,3
LEARN:((e11(e31e01))e11),-2,1,3
--------------------------------------
loop number is:112
The best choice of the anticipation interaction is composite and its weight
is:4
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e31e01),4,5,1
```

```
The enacted composite interaction is:  (e31e01),4,5,1
learnCompositeInteraction()
REINFORCE:(e11(e31e01)),1,5,2
LEARN:((e31e01)(e11(e31e01))),5,1,3
LEARN:(((e31e01)e11)(e31e01)),5,1,3
---------------------------------------
loop number is:113
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
learnCompositeInteraction()
LEARN:((e31e01)e21),1,1,2
LEARN:(e11((e31e01)e21)),-2,1,3
LEARN:((e11(e31e01))e21),-2,1,3
---------------------------------------
...
---------------------------------------
loop number is:153
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e11,-3,0,0
The primitive enactedInteraction is:  e11,-3,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e11(e31e01)),1,8,2
The enacted composite interaction is:  (e11(e31e01)),1,8,2
learnCompositeInteraction()
REINFORCE:(e41(e11(e31e01))),0,3,3
LEARN:(e41(e41(e11(e31e01)))),-1,1,4
LEARN:((e41e41)(e11(e31e01))),-1,1,3
---------------------------------------
loop number is:154
The best choice of the anticipation interaction is composite and its weight
is:9
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e31e01),4,6,1
The enacted composite interaction is:  (e31e01),4,6,1
learnCompositeInteraction()
REINFORCE:((e11(e31e01))(e31e01)),5,4,3
LEARN:(e41((e11(e31e01))(e31e01))),4,1,4
LEARN:((e41(e11(e31e01)))(e31e01)),4,1,4
---------------------------------------
loop number is:155
The best choice of the anticipation interaction is composite and its weight
is:8
```

```
The best choice of the anticipation interaction’s weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e30,-2,0,0
The intended composite interaction is:  (e31e01),4,6,1
The enacted composite interaction is:  e30,-2,0,0
learnCompositeInteraction()
REINFORCE:((e31e01)e30),2,6,2
LEARN:((e11(e31e01))((e31e01)e30)),3,1,3
LEARN:(((e11(e31e01))(e31e01))e30),3,1,4
---------------------------------------
...
---------------------------------------
loop number is:301
The best choice of the anticipation interaction is composite and its weight
is:4
The best choice of the anticipation interaction’s weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e21(e01(e31e01))),6,7,3
The enacted composite interaction is:  (e21(e01(e31e01))),6,7,3
learnCompositeInteraction()
REINFORCE:(e51(e21(e01(e31e01)))),5,4,4
REINFORCE:(e30(e51(e21(e01(e31e01))))),3,2,5
REINFORCE:((e30e51)(e21(e01(e31e01)))),3,2,4
---------------------------------------
loop number is:302
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
learnCompositeInteraction()
LEARN:((e21(e01(e31e01)))e21),3,1,4
LEARN:(e51((e21(e01(e31e01)))e21)),2,1,5
LEARN:((e51(e21(e01(e31e01))))e21),2,1,5
---------------------------------------
loop number is:303
The best choice of the anticipation interaction is composite and its weight
is:7
The best choice of the anticipation interaction’s weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e01(e31e01)),9,5,2
The enacted composite interaction is:  (e01(e31e01)),9,5,2
```

```
learnCompositeInteraction()
REINFORCE:(e21(e01(e31e01))),6,8,3
LEARN:((e21(e01(e31e01)))(e21(e01(e31e01)))),12,1,4
LEARN:(((e21(e01(e31e01)))e21)(e01(e31e01))),12,1,5
--------------------------------------
...
--------------------------------------
loop number is:313
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e51(e21(e01(e31e01)))),5,5,4
The enacted composite interaction is:  (e51(e21(e01(e31e01)))),5,5,4
learnCompositeInteraction()
REINFORCE:(e30(e51(e21(e01(e31e01))))),3,3,5
LEARN:((e31e01)(e30(e51(e21(e01(e31e01)))))),7,1,6
LEARN:(((e31e01)e30)(e51(e21(e01(e31e01))))),7,1,5
--------------------------------------
loop number is:314
The best choice of the anticipation interaction is composite and its weight
is:2
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e30,-2,0,0
The intended composite interaction is:  (e31e01),4,6,1
The enacted composite interaction is:  e30,-2,0,0
learnCompositeInteraction()
REINFORCE:((e51(e21(e01(e31e01))))e30),3,2,5
LEARN:(e30((e51(e21(e01(e31e01))))e30)),1,1,6
LEARN:((e30(e51(e21(e01(e31e01)))))e30),1,1,6
--------------------------------------
loop number is:315
The best choice of the anticipation interaction is composite and its weight
is:3
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
```

```
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e30,-2,0,0
REINFORCE:(e01e30),3,9,1
REINFORCE:(e21(e01e30)),0,5,2
REINFORCE:(e51(e21(e01e30))),-1,4,3
The intended composite interaction is:  (e51(e21(e01(e31e01)))),5,5,4
The enacted composite interaction is:  (e51(e21(e01e30))),-1,4,3
learnCompositeInteraction()
REINFORCE:(e30(e51(e21(e01e30)))),-3,2,4
LEARN:((e51(e21(e01(e31e01))))(e30(e51(e21(e01e30))))),2,1,5
LEARN:(((e51(e21(e01(e31e01))))e30)(e51(e21(e01e30)))),2,1,6
----------------------------------------
...
----------------------------------------
loop number is:319
The best choice of the anticipation interaction is composite and its weight
is:4
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e51(e21(e01(e31e01)))),5,5,4
The enacted composite interaction is:  (e51(e21(e01(e31e01)))),5,5,4
learnCompositeInteraction()
REINFORCE:(e30(e51(e21(e01(e31e01))))),3,4,5
REINFORCE:((e31e01)(e30(e51(e21(e01(e31e01)))))),7,2,6
REINFORCE:(((e31e01)e30)(e51(e21(e01(e31e01))))),7,2,5
----------------------------------------
loop number is:320
The best choice of the anticipation interaction is Primitive
The primitive intendedInteraction is:  e40,-2,0,0
The primitive enactedInteraction is:  e40,-2,0,0
learnCompositeInteraction()
LEARN:((e51(e21(e01(e31e01))))e40),3,1,5
LEARN:(e30((e51(e21(e01(e31e01))))e40)),1,1,6
LEARN:((e30(e51(e21(e01(e31e01)))))e40),1,1,6
----------------------------------------
loop number is:321
The best choice of the anticipation interaction is composite and its weight
is:1
The best choice of the anticipation interaction's weight less than threshold
or proclivity is negative
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
learnCompositeInteraction()
REINFORCE:(e40e51),-3,4,1
```

```
LEARN:((e51(e21(e01(e31e01))))(e40e51)),2,1,5
LEARN:(((e51(e21(e01(e31e01))))e40)e51),2,1,6
--------------------------------------
...
--------------------------------------
loop number is:401
The best choice of the anticipation interaction is composite and its weight
is:16
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e51(e21(e01(e31e01)))),5,7,4
The enacted composite interaction is:  (e51(e21(e01(e31e01)))),5,7,4
learnCompositeInteraction()
REINFORCE:(e30(e51(e21(e01(e31e01))))),3,15,5
REINFORCE:((e11(e31e01))(e30(e51(e21(e01(e31e01)))))),4,3,6
REINFORCE:(((e11(e31e01))e30)(e51(e21(e01(e31e01))))),4,3,5
--------------------------------------
loop number is:402
The best choice of the anticipation interaction is composite and its weight
is:17
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e30,-2,0,0
The intended composite interaction is:  (e31e01),4,6,1
The enacted composite interaction is:  e30,-2,0,0
learnCompositeInteraction()
REINFORCE:((e51(e21(e01(e31e01))))e30),3,9,5
REINFORCE:(e30((e51(e21(e01(e31e01))))e30)),1,7,6
REINFORCE:((e30(e51(e21(e01(e31e01))))))e30),1,7,6
--------------------------------------
loop number is:403
The best choice of the anticipation interaction is composite and its weight
is:17
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
```

```
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e51(e21(e01(e31e01)))),5,7,4
The enacted composite interaction is:  (e51(e21(e01(e31e01)))),5,7,4
learnCompositeInteraction()
REINFORCE:(e30(e51(e21(e01(e31e01))))),3,16,5
REINFORCE:((e51(e21(e01(e31e01))))(e30(e51(e21(e01(e31e01)))))),8,3,6
REINFORCE:(((e51(e21(e01(e31e01))))e30)(e51(e21(e01(e31e01))))),8,3,6
----------------------------------------
...
----------------------------------------
loop number is:416
The best choice of the anticipation interaction is composite and its weight
is:20
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e51(e21(e01(e31e01)))),5,9,4
The enacted composite interaction is:  (e51(e21(e01(e31e01)))),5,9,4
learnCompositeInteraction()
REINFORCE:(e30(e51(e21(e01(e31e01))))),3,18,5
REINFORCE:((e51(e21(e01(e31e01))))(e30(e51(e21(e01(e31e01)))))),8,4,6
REINFORCE:(((e51(e21(e01(e31e01))))e30)(e51(e21(e01(e31e01))))),8,4,6
----------------------------------------
loop number is:417
The best choice of the anticipation interaction is composite and its weight
is:21
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:  (e31e01),4,6,1
The enacted composite interaction is:  (e31e01),4,6,1
learnCompositeInteraction()
REINFORCE:((e51(e21(e01(e31e01))))(e31e01)),9,12,5
REINFORCE:(e30((e51(e21(e01(e31e01))))(e31e01))),7,7,6
REINFORCE:((e30(e51(e21(e01(e31e01))))))(e31e01)),7,7,6
----------------------------------------
...
----------------------------------------
loop number is:443
The best choice of the anticipation interaction is composite and its weight
is:2
```

```
The best choice of the anticipation interaction’s weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e11,-3,0,0
The primitive enactedInteraction is:  e11,-3,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:
((e11(e31e01))(e51(e21(e01(e31e01))))),6,3,5
The enacted composite interaction is:
((e11(e31e01))(e51(e21(e01(e31e01))))),6,3,5
learnCompositeInteraction()
REINFORCE:((e51(e21(e01e30)))((e11(e31e01))(e51(e21(e01(e31e01)))))),5,3,6
LEARN:(e30((e51(e21(e01e30)))((e11(e31e01))(e51(e21(e01(e31e01))))))),3,1,7
LEARN:((e30(e51(e21(e01e30))))((e11(e31e01))(e51(e21(e01(e31e01)))))),3,1,6
----------------------------------------
...
----------------------------------------
loop number is:554
The best choice of the anticipation interaction is composite and its weight
is:32
The best choice of the anticipation interaction’s weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:
((e31e01)(e51(e21(e01(e31e01))))),9,25,5
The enacted composite interaction is:
((e31e01)(e51(e21(e01(e31e01))))),9,25,5
learnCompositeInteraction()
REINFORCE:((e51(e21(e01(e31e01))))((e31e01)
```

```
(e51(e21(e01(e31e01))))),14,30,6
REINFORCE:(e30((e51(e21(e01(e31e01))))
((e31e01)(e51(e21(e01(e31e01)))))),12,4,7
REINFORCE:((e30(e51(e21(e01(e31e01)))))
((e31e01)(e51(e21(e01(e31e01))))),12,4,6
----------------------------------------
...
----------------------------------------
loop number is:610
The best choice of the anticipation interaction is composite and its weight
is:48
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:
((e31e01)(e51(e21(e01(e31e01))))),9,25,5
The enacted composite interaction is:
((e31e01)(e51(e21(e01(e31e01))))),9,25,5
learnCompositeInteraction()
REINFORCE:((e51(e21(e01(e31e01))))((e31e01)
(e51(e21(e01(e31e01)))))),14,38,6
REINFORCE:(e30((e51(e21(e01(e31e01))))((e31e01)
(e51(e21(e01(e31e01)))))),12,12,7
REINFORCE:((e30(e51(e21(e01(e31e01)))))((e31e01)
(e51(e21(e01(e31e01)))))),12,12,6
----------------------------------------
loop number is:611
The best choice of the anticipation interaction is composite and its weight
is:63
The best choice of the anticipation interaction's weight plus than threshold
and proclivity is positive
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The primitive intendedInteraction is:  e51,-1,0,0
The primitive enactedInteraction is:  e51,-1,0,0
The primitive intendedInteraction is:  e21,-3,0,0
The primitive enactedInteraction is:  e21,-3,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
```

```
The primitive intendedInteraction is:  e31,-1,0,0
The primitive enactedInteraction is:  e31,-1,0,0
The primitive intendedInteraction is:  e01,5,0,0
The primitive enactedInteraction is:  e01,5,0,0
The intended composite interaction is:
((e31e01)(e51(e21(e01(e31e01))))),9,25,5
The enacted composite interaction is:
((e31e01)(e51(e21(e01(e31e01))))),9,25,5
learnCompositeInteraction()
REINFORCE:(((e31e01)(e51(e21(e01(e31e01)))))
((e31e01)(e51(e21(e01(e31e01)))))),18,15,6
REINFORCE:((e51(e21(e01(e31e01))))(((e31e01)(e51(e21(e01(e31e01)))))
((e31e01)(e51(e21(e01(e31e01)))))))),23,12,7
REINFORCE:(((e51(e21(e01(e31e01))))((e31e01)(e51(e21(e01(e31e01))))))
((e31e01)(e51(e21(e01(e31e01)))))),23,12,7
```

Appendix C

Agent's performance in diverse environments

As we mentioned in the Section 7.4.1, with interaction traces exported from the GAIT, from the 357th interaction, the agent could successfully interact with its environment and start avoiding unfavorable interactions (the collisions with the walls) using regularities that it has learned. In the 500th interaction, we changed the environment in different ways (an example as shown in Figure C.1) and examine the agent's performance in the changed environment.

C.1 The first changed environment

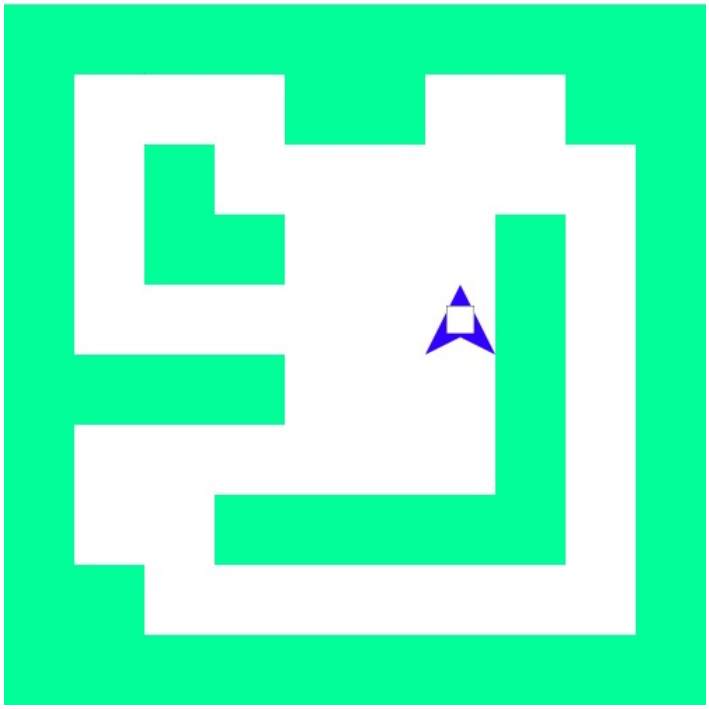


Figure C.1: The first changed environment.

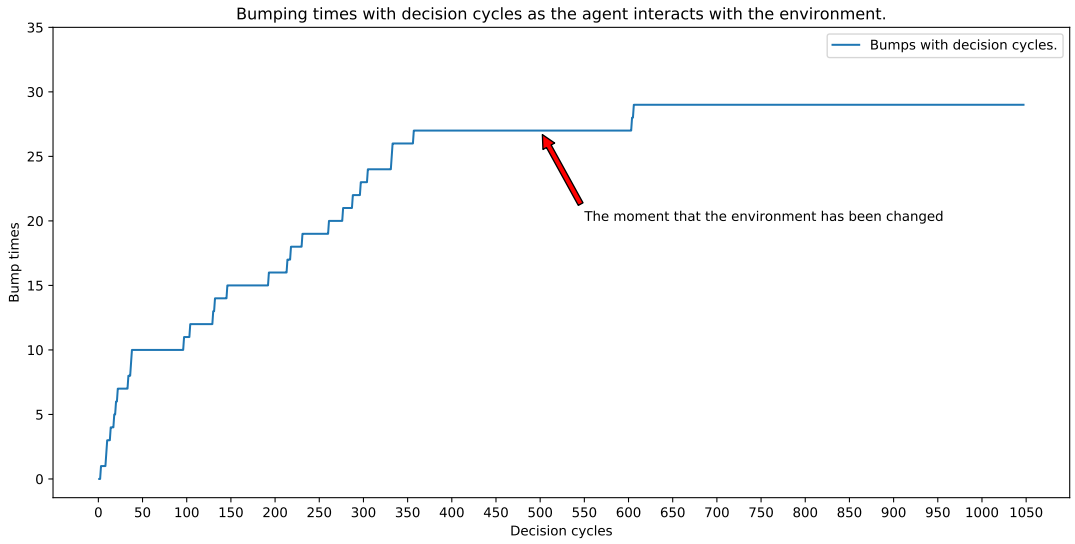


Figure C.2: The bump times with decision cycles in the first changed environment.

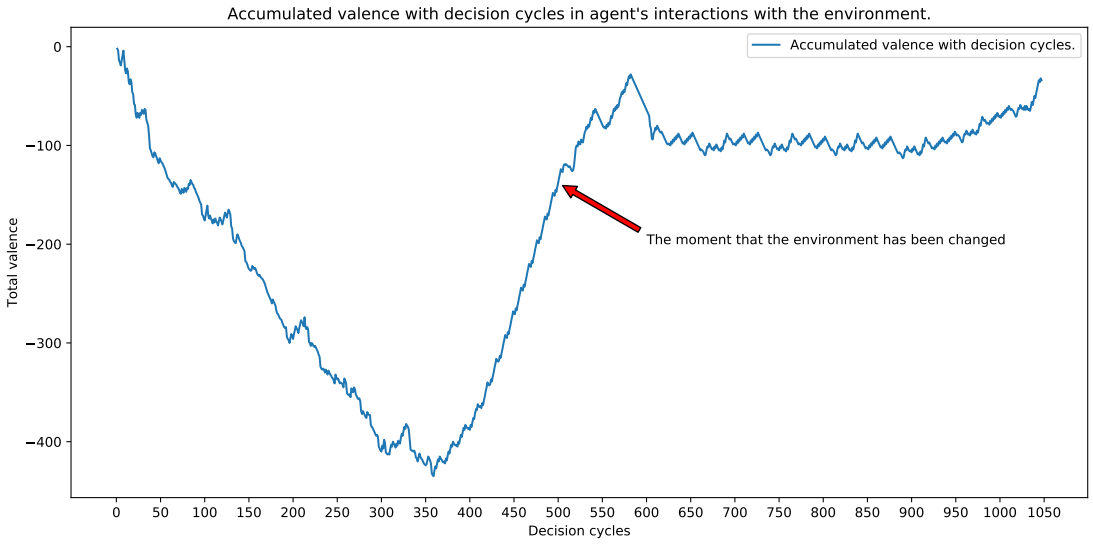


Figure C.3: The total valence with decision cycles in the first changed environment.

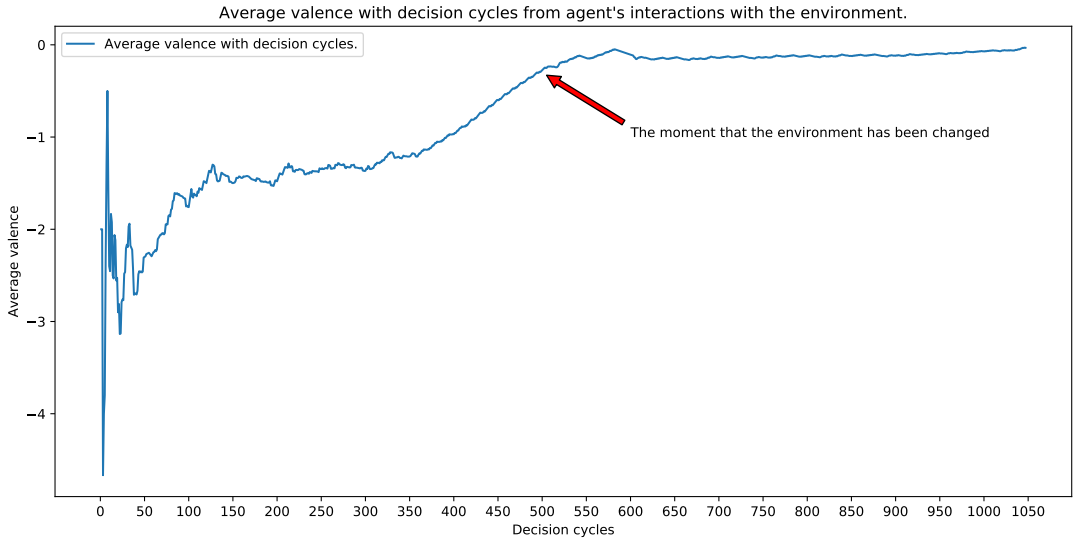


Figure C.4: The average valence with decision cycles in the first changed environment.

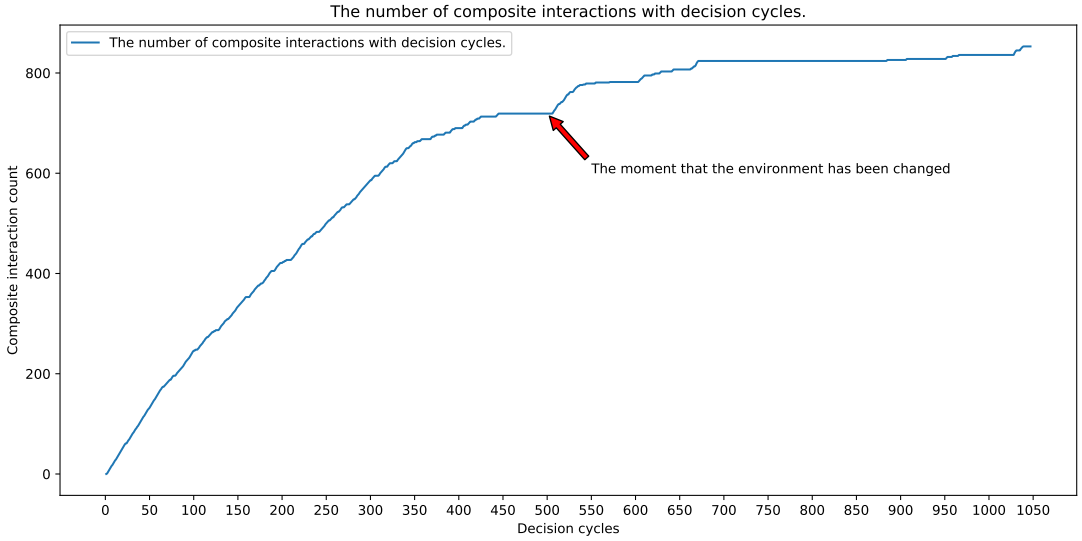


Figure C.5: The number of composite interactions with decision cycles in the first changed environment.

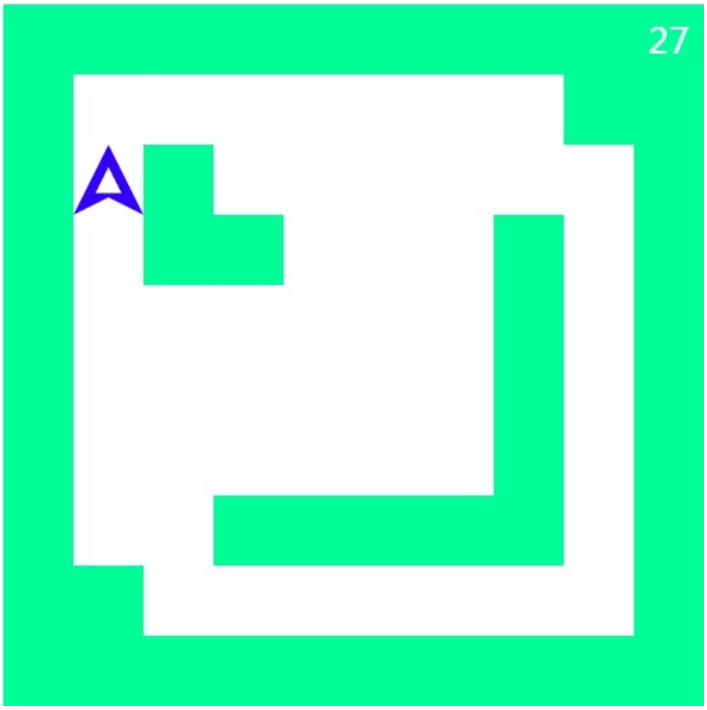


Figure C.6: The second changed environment.

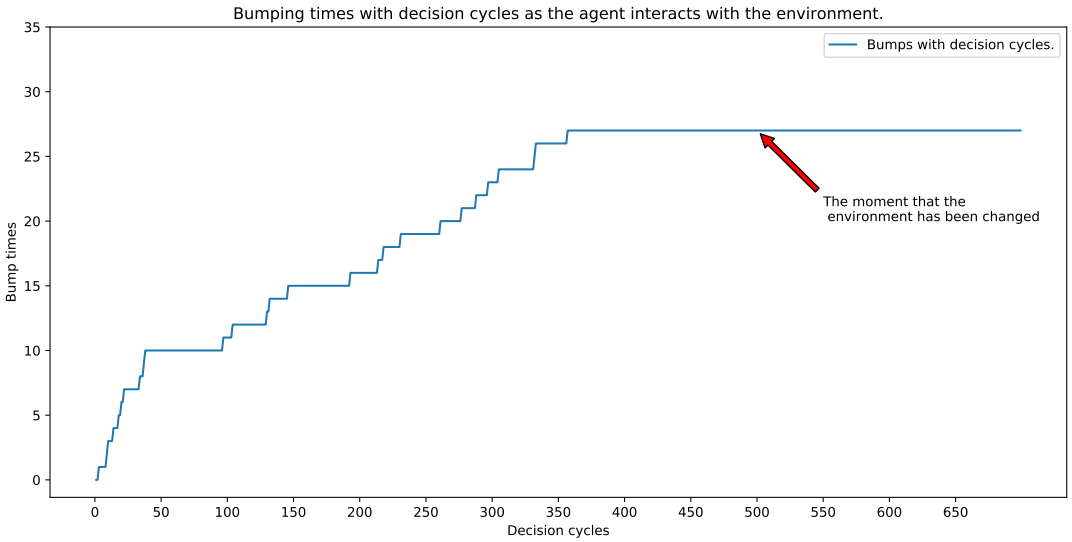


Figure C.7: The bump times with decision cycles in the second changed environment.

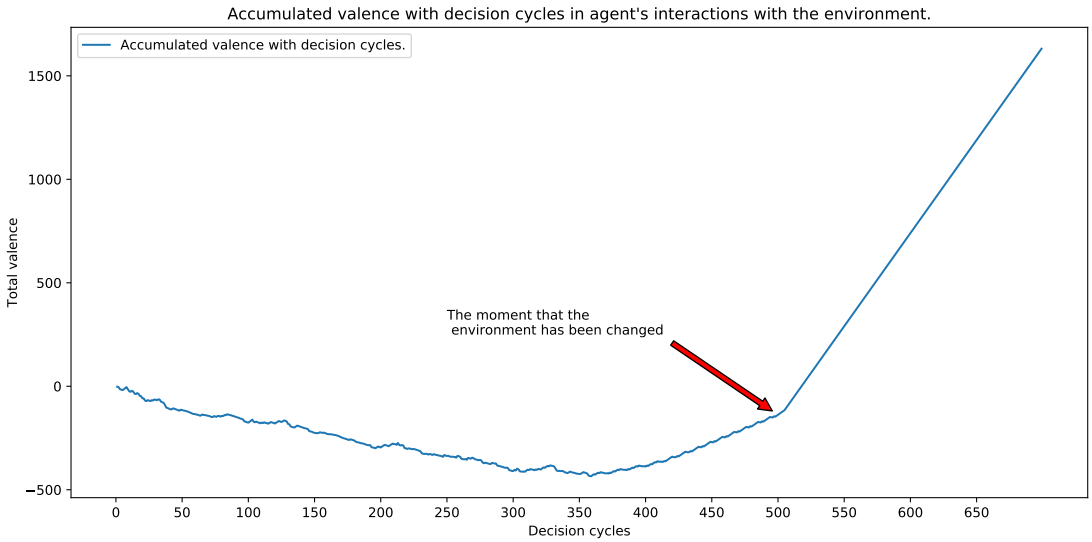


Figure C.8: The total valence with decision cycles in the second changed environment.

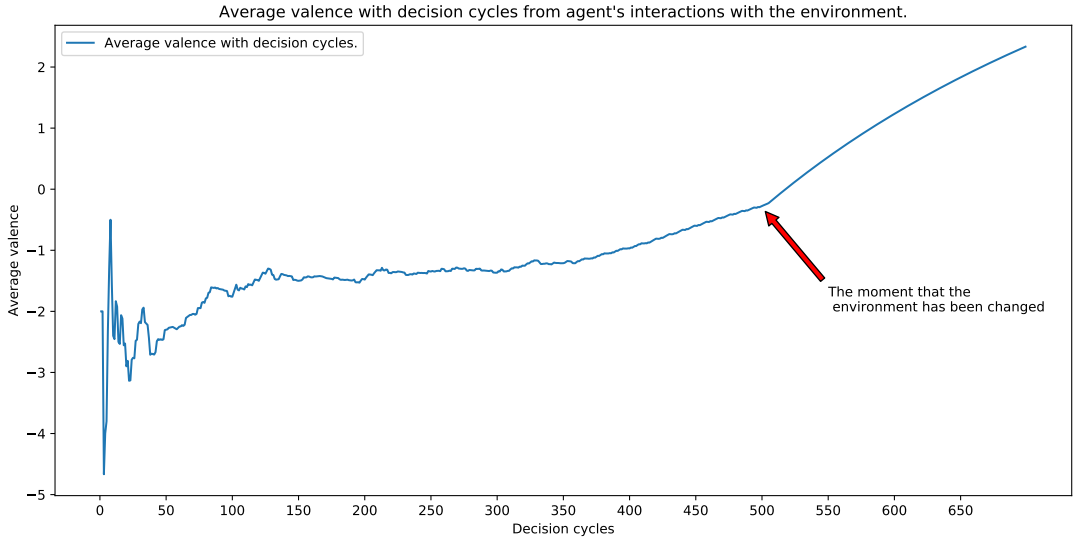


Figure C.9: The average valence with decision cycles in the second changed environment.

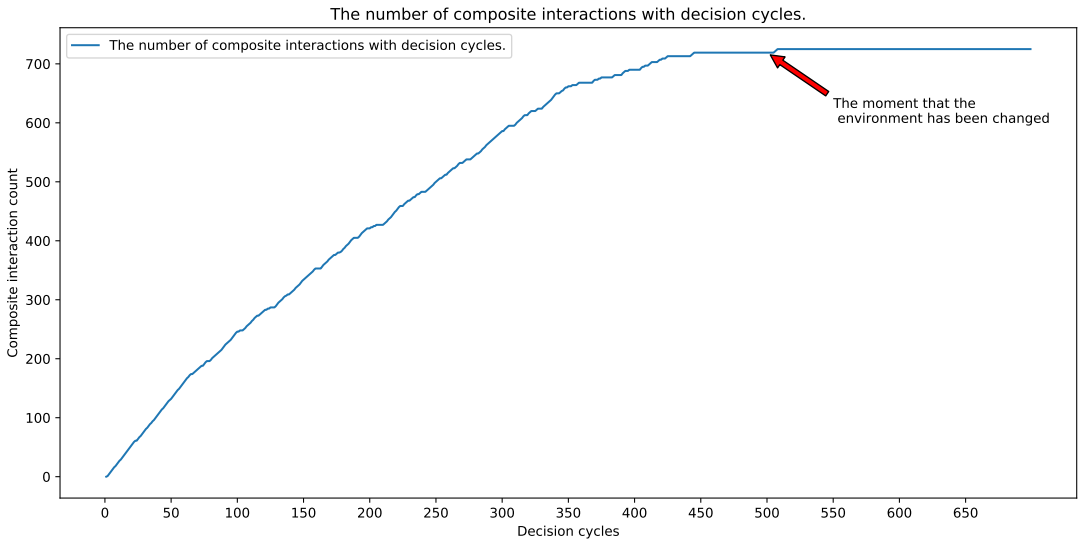


Figure C.10: The number of composite interactions with decision cycles in the second changed environment.

C.2 The second changed environment

C.3 The third changed environment

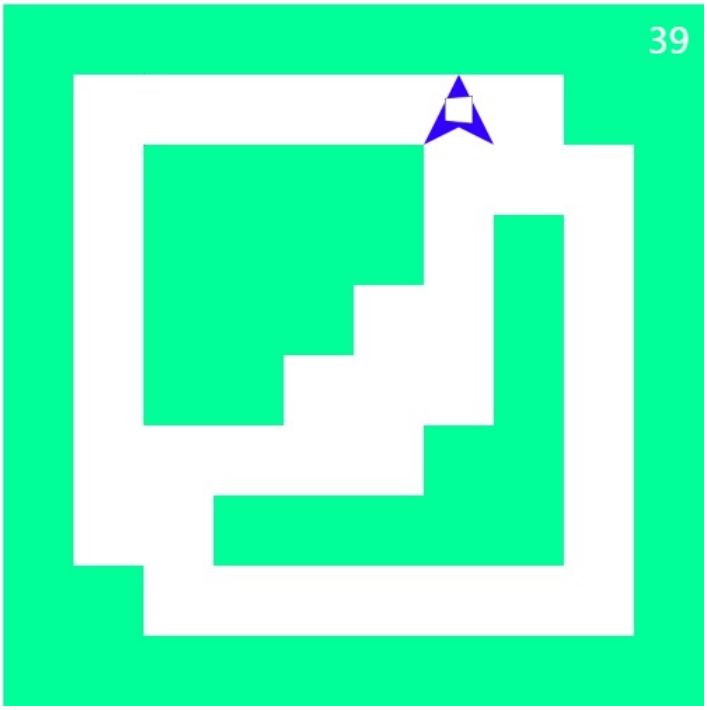


Figure C.11: The third changed environment.

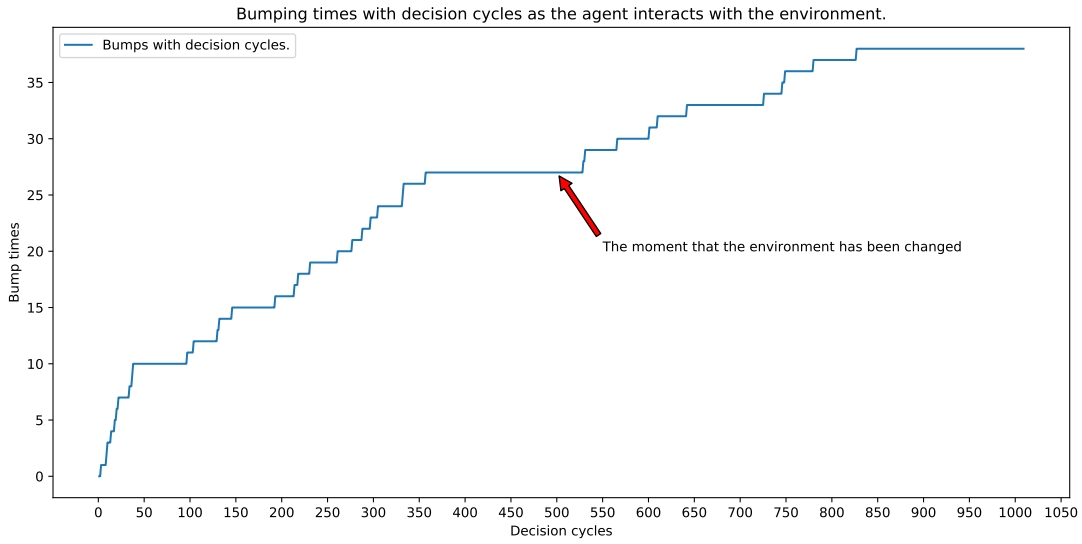


Figure C.12: The bump times with decision cycles in the third changed environment.

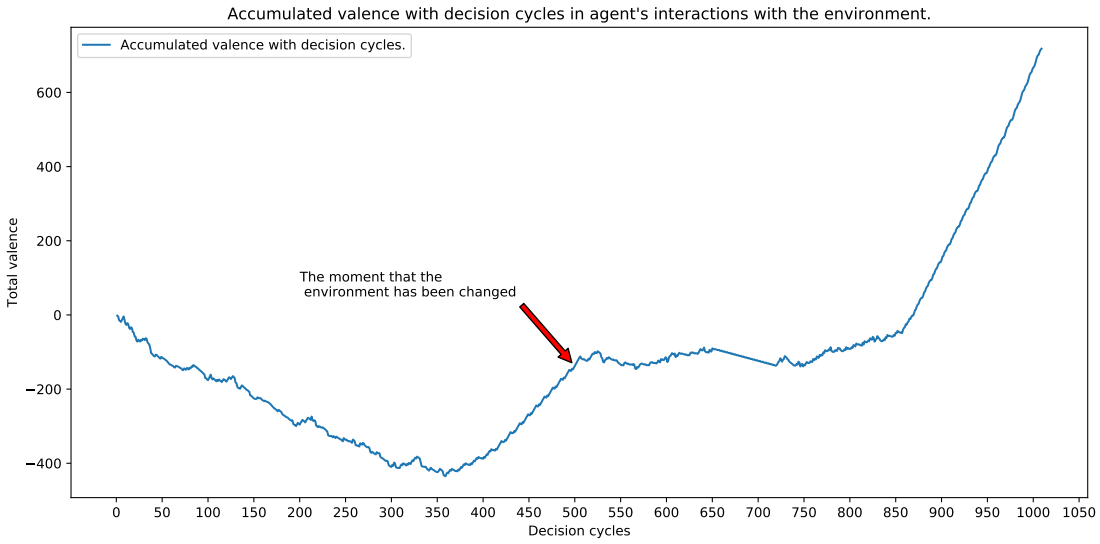


Figure C.13: The total valence with decision cycles in the third changed environment.

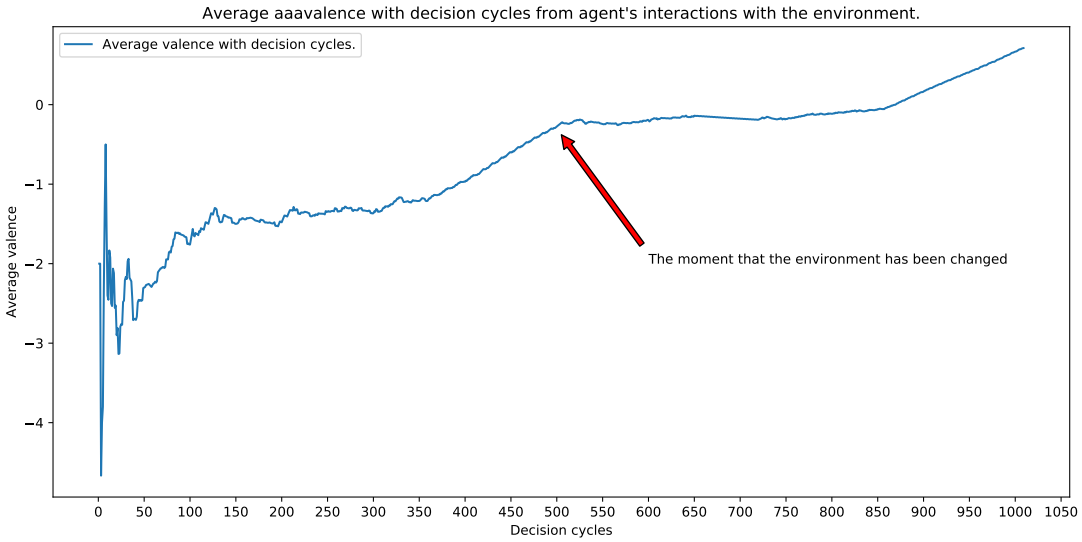


Figure C.14: The average valence with decision cycles in the third changed environment.

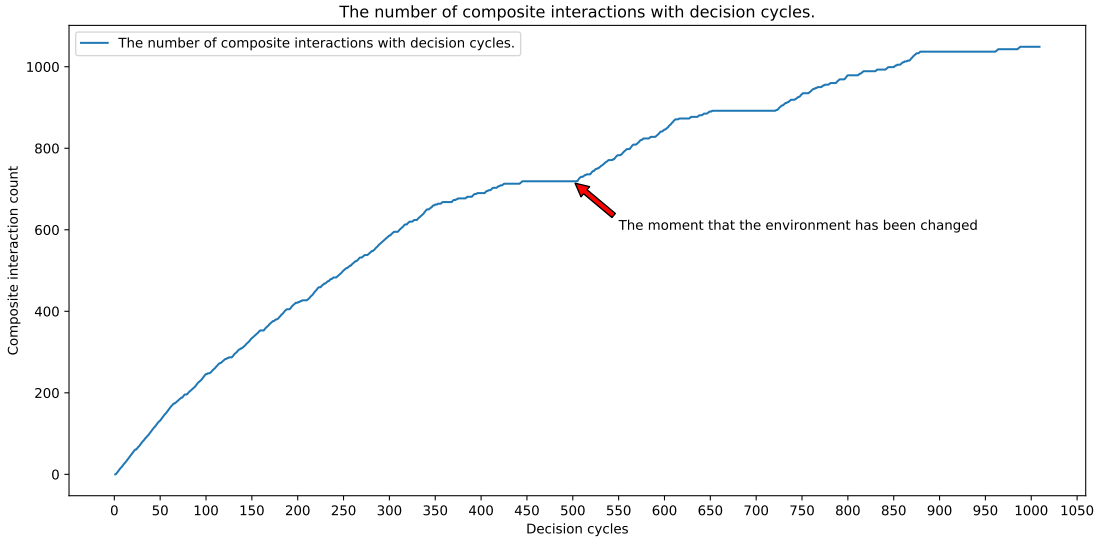


Figure C.15: The number of composite interactions with decision cycles in the third changed environment.