Learning with Play Behaviour in Artificial Agents

Suresh Kumar

Supervisors: **Dr Patricia Shaw**
Prof Qiang Shen

Aberystwyth University

This dissertation is submitted for the degree of
Doctor of Philosophy

June 2019

I would like to dedicate this thesis to my wife, Ambran Suresh, and daughter, Shriya Suresh.

Declaration

Word Count of thesis: DECLARATION	58073 (Excluding references, numbers and symbols)
This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.	
Signature:	Suresh Kumar
Date	13/06/2019

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where ***correction services** have been used, the extent and nature of the correction is clearly marked in a footnote(s). Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signature: Suresh Kumar Date: 13/06/2019

[*this refers to the extent to which the text has been corrected by others]

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signature: Suresh Kumar Date: 13/06/2019

Acknowledgements

This research is supported by the Aberystwyth University Doctoral Training Program, Faculty Development Program Sukkur IBA University Pakistan, my supervisors Dr Patricia Shaw and Prof Qiang Shen, my colleagues Prof Mark Lee, Dr Alexandros Giagkos, Dr Daniel Lewkowickz and Dr Raphäel Braud, and the UK Engineering and Physical Sciences Research Council (EPSRC), grant No. EP/M013510/1. I'm grateful of Dr Michael Sheldon, for the development of the PSchema tool, and Dr Frank Guerin and Dr Bernie Tiddeman for their feedback on this thesis.

Abstract

This thesis investigates the role of exploratory play in the development of the basic knowledge of actions and objects involved in play with artificial agents. We developed a learning system, *Dev-PSchema*, inspired from the sensorimotor stage and schema mechanism of Piaget's theory of cognitive development. The learning system enables the artificial agents to develop their knowledge, in the shape of schemas containing action and perceptions, based on their existing knowledge and play behaviour. We demonstrate the system embodied in two agents, a simulator and a real robot, developing their knowledge through exploratory sensorimotor experiences and extending for novel situations of the environment through schema generalisation. The schema generalisation mechanism enables the agents to extend their knowledge for novel situations and predict action outcomes. The agents begin learning with a set of basic actions, provided to interact with their environment and perform suitable actions selected through an action selection mechanism, the excitation calculator. This mechanism is modelled on the habituation paradigm, widely studied in developmental psychology. We demonstrate how the excitation mechanism can be tuned to demonstrate a range of behaviour preferences in the artificial agents, similar to the infants observed in the developmental psychology studies.

We then demonstrate the agents developing complex actions, labelled as schema chains, from their basic knowledge gained through the exploratory play. The agent demonstrates performing the schema chains as a singular action following a few repetitions. This skill is modelled on the *chain reflex* and *motor program* behaviours observed in humans while performing a sequence of actions. This capability enables the agent to develop complex skills that are used to achieve a state in the environment which is, otherwise, not possible to achieve with a single action. Furthermore, we demonstrate the capability to scaffold the learning of the agent through achieving tasks, increasing in complexity.

In conclusion, we demonstrate that the exploratory play helps the agents to develop their knowledge about actions and the objects involved. The developed knowledge is further used

to explore the environment, hence demonstrating open-ended learning.

Table of Contents

List of Figures

List of Algorithms

List of Equations

List of Tables

Chapter 1

Introduction to Modelling a Human approach to Learning

This thesis is concerned with the development of a mechanism for driving play like behaviour, inspired by developmental psychology. The mechanism will be implemented for artificial agents, both in simulation and on a real robot. The agents then demonstrated developing an understanding of objects through the process of playful exploration grounded in the field of epigenetic robotics through embodiment and developmental learning in a lab environment [166].

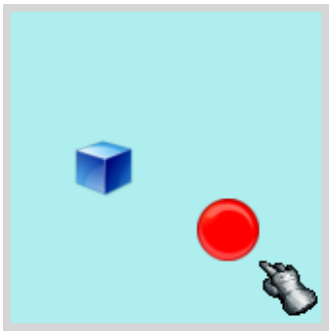
In this thesis we investigate a *schema* based learning approach containing underlying algorithms of action selection, action sequencing and developing contextual understanding for modelling play behaviour and investigate the behaviours of the artificial agents by varying different parameters of the system. The learning system used as a play generator in this thesis is inspired from Piaget's cognitive theory and play behaviour in early infancy. We investigate the development of object knowledge through play with an open-ended learning system driving exploratory behaviours. We also investigate the adaptation and reuse of the experience it has gained in novel environments.

The main focus in this thesis to which the schema mechanism is applied is that of building object understanding through play behaviour. Play evolves over time from practice to symbolic and then later to play with rules [144, 54]. Practice play appears in young infants (at the age of 1 month) where they enjoy repeating the actions which they have learnt previously. Infants' interest in practice play declines and gradually changes into symbolic play where they become interested in imitating contents of the actions rather than the actions [54]. In this thesis we are focused on practice play behaviour and cognitive development in early infancy, therefore, we mainly focused on independent play behaviour in infancy without any social support from parents or caregivers.

The learning system is able to record perceived sensory information from the environment along with actions that caused changes to the perceptions, in the form of schemas. Later, such schemas are used to interact with the environment and objects in it. Developing the accommodation process described by Piaget, the system evolves different schemas through continuous explorations and extends learning to novel situations. Furthermore, we demonstrate the capability of the system to simulate different individuals' behaviours, as observed from infants.

The learning system is integrated within artificial agents acting in two different environments. First is a virtual agent in a simulator and the second in the real world via a humanoid robot, iCub [122]. Figure 1.1 shows the artificial agents used in this thesis. Both agents are capable of perceiving the environment with their sensors and acting on the environment with their end effectors i.e. Hand.

We begin with the agents having knowledge about their motor capabilities and little or no knowledge about the environment. The agents are left to play in the environment containing different objects, learning "cause and effect" relationships between objects and the agent's motor actions.



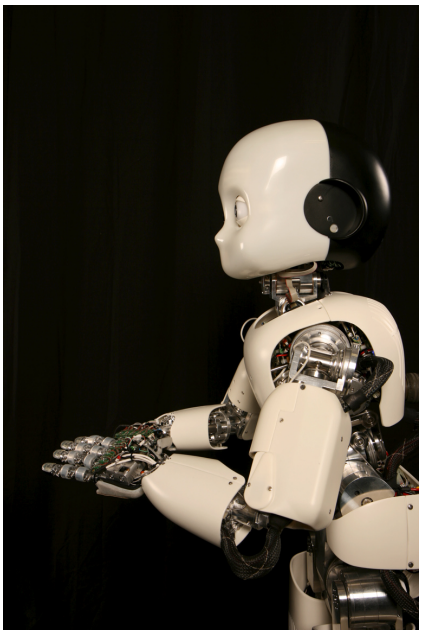


Fig. 1.1 The two artificial agents we used in this thesis. Left: A simple simulator with the capability to perceive and act. Right: iCub, a humanoid robot with vision and 53 degree of freedom.

Furthermore, we also want to emphasize the adaptability of the learning system for two completely different agents, from a simple simulator to a humanoid robot. This enables us to observe behaviours in different agents depending upon their perceptions of the environment using their sensors provided.

In the remainder of this chapter, we discuss motivations for developing robotic systems inspired by developmental psychology and outline the literature on play in early infancy and learning in order to identify the key characteristics of play behaviour in infants. Finally, we will outline the main contributions of this thesis and existing related publications by the author.

1.1 Motivations: Artificial Intelligence and Developmental Robotics

The ability to socialize and communicate through verbal and non-verbal languages, to solve problems and to reason, enables humans to be recognised as an intelligent species. Although there is not any single scientific definition for Intelligence, it can be considered as a collection of various attributes such as perception, reasoning, planning, adaptation and learning, autonomy, communication and creativity [74]. Understanding the human brain has been a topic of interest for different fields of studies, including medical science, psychology and artificial intelligence (AI). On the one hand medical science, including neuroscience, is focused on understanding the structure and functionality of the human brain. On the other hand, psychologists are trying to understand how human cognition develops and how we think. AI brings together both aspects, attempting to produce biologically plausible and biologically inspired models.

The brain is the most complex organ in human body and one of the slowest to develop [177]. When fully developed, the human brain is divided into lobes and each lobe has its own functions. For example, problem solving, creativity and short term memory are handled by the frontal lobe, while temporal lobes are responsible for memory and language, and the parietal lobe processes sensory information [80]. Simply, having a brain is not sufficient to be an intelligent species. It needs to go through a long process of development and learning. A complex structure of neuron connections is created in the human brain during the learning and development process.

Although enormous development can be seen in technology, it is safe to say that we are still not able to achieve all the characteristic in machines to be intelligent described by Honavar [74]. AI researchers model psychological and neuroscience studies on computers in an attempt to reproduce human-like behaviours in machines. Nilsson [133] relates AI with intelligent behaviours, such as learning, reasoning, communicating and perception. In the

broad sense, the goal of AI is to develop machines that can think and/or act as humans do. Despite this, a wide range of devices and machines, from cell phones through to fridges, are claimed to be intelligent or smart. However, for a machine to be considered as intelligent, it has to be automatic and autonomous. Such machines are commonly referred to as robots.

Capek [32] used the word "Robot" in his science fiction play, premiered in 1921, "Rosumovi universal robots" translated in English as Rossum's Universal Robots. In the play, humans acted as human-like machines. In 2010 a humanoid robot, Geminoid F, performed in a play in Tokyo, although still remotely operated from off stage [71]. Increasingly, real robots are being used in films, such as BB-8 in Star Wars: The Force Awakens. These two plays show different aspects of the development of Robotics and AI. In the former [32], an idea was presented to have a human-like machine, in shape and intelligence. More recently, the later play demonstrates the advancement in the field of robotics, particularly humanoid robotics. However, the robot in "Sayonara" [71] was operated by a human, therefore, it does not yet fit the definition for an "Intelligent Robot" as described above.

In general, robots are useful for working in environments where it is difficult or unsafe for humans to be. Moreover, robots do not tire from repetition of tasks and are typically very precise between repetitions. In science fiction, Asimov [11] defined three rules of robotics, where intelligent robots are working alongside humans. The first and most important one says "A robot may not injure a human being, or, through inaction, allow a human being to come to harm". While this rule is currently not applied, as it highlights that intelligent robots should be capable of perceiving, learning and reasoning about others, either human or robot, actions. They should be capable of predicting effects of actions/changes in environment and understanding what harm that might cause to humans. We, as humans, possess such capabilities. However, such capabilities are developed through years of experiences in the environment. We propose that, for a robot to have such capabilities as humans do, it must undergo a process of learning to gain experience as humans do.

Meeden and Blank [120] explain three traditional approaches for developing intelligent robots; i) Direct programming, ii) Supervised learning, iii) Evolutionary adaptation. Although none of these methods shows capability on a par with humans, they are motivations towards new methods, such as developmental learning. This idea seems to be very much supporting the idea proposed by Turing [188]; A child-like brain is easier to simulate than an adult-like brain and then proper learning allow it to develop to adulthood. This idea is the foundation of the field of "Developmental Robotics". Asada et al. [10] further propose Cognitive developmental robotics for humanoid robots.

In humans, changes in the central nervous system (CNS) due to the interaction with the environment are said to be the result of development, with innate behaviours or reflexes helping in this process [191]. As already discussed, replicating a child's brain should be simpler than an adult's. Turing [188] refers to infant's brain as a blank slate and assumes it can be easily written or programmed. Hence he proposes that in order to make a machine intelligent, it should begin with an empty brain and be allowed it to develop in the same way as humans do.

Developmental robotics also describes the importance of gaining experience through interaction with the environment, requiring embodiment and therefore also learning about itself. This approach is inspired by psychological studies of human development. Developmental robotics is a prominent interdisciplinary research field. Lungarella et al. [109] identifies developmental robotics as the intersection of two branches of science, robotics and developmental sciences such as developmental psychology. In a broad sense, this discipline supports the human-like development of robots, recognising that the learning must be embodied in a robotic platform, enabling it to interact with the world [28].

Vernon [191] reports that the human body and brain evolve together and therefore cannot be considered as separate. Embodiment has a great importance in the field of robotics and artificial intelligence. Making use of a robotic platform modelled on a humanoid form,

incorporating dexterity and perception enables more natural social interaction with humans and also provides similar affordances for interacting with the environment [27]. If robots are to help humans in any environment containing objects and instruments designed for humans, then robots should be embodied like humans. Asada et al. [9] also support the prominence of humanoid robots in cognitive developmental research. They believe, embodiment specifies the physical constraints and provides meaningful interactions in the environment.

In order to build a computational model which possesses the capability to develop over a period of time with experiences as humans do, detailed studies of human, particularly infant, development are of great importance. This means "Developmental psychology" plays an important role in the development of models based on human learning, especially for models of infant development. In section 1.2, we discuss the development and learning during infancy, followed by characteristics of play we aim to model as part of this contribution to the field.

1.2 Learning in Early Infancy

Vernon [192] describes development as the result of interactions between an agent and its environment, and agent's interactions with itself. Human development and learning are spread over a long period of time starting from within the mother's womb [64]. Born with different physical and cognitive constraints, infants during infancy develop physically and cognitively over the period of time. The gradual change in the physical and mental constraints at different ages help to shape the path for the subsequent important growth and reduce the complexity of learning [100]. This can also be considered as a reason for slow physical and cognitive development over a long period of time. This staged development in infants is seen as sequential and predictable [146, 60] and proximal to distal i.e., head to toe [177, 23].

There are different types of theories by psychologists on developmental learning. A long debate of *"Nature-Nurture"* or *"Nativism-Empiricism"* theories, continuous and discontinuous and active-passive learning, is still going on. Theories supporting nature or nativism consider development is mostly influenced by the nature of the genes or the organism itself and only a little by the environment. It has also been observed that prenatal development is mostly gene directed but the environment can have an effect as well, such as consumption of alcoholic drinks and malnutrition during pregnancy cause malformation and impairments [177]. The idea of core knowledge systems given by Spelke [179, 178] is one of the recent studies supporting the nature side of the Nature-Nurture debate. According to this theory, humans possess some innate systems of knowledge such as numbers, space and geometry. She believes that human cognition is based on such systems of knowledge which develops further to higher levels over the course of the development [178]. On the other side, researchers supporting the nurture or empiricism theories consider the environment as more influential than the organism itself in the learning and developmental process. Empiricism claims the learning and knowledge develops over a period of time based on the active experiences in the environment. Although infants show some capabilities of knowledge in early infancy, it has been found that infants show some limitations for different cognitive tasks at the different ages, for example, spatial orientation and frame of references [128], and the relationship between numerosity and displacement [58].

There is a long list of researchers supporting either side of the nature-nurture theories. Theories given by Piaget and Vygotsky are still considered as relevant and applied to date. Although both of them see the infant as an active participant in the development and learning process, they differ in their views about the involvement of the environment in this process. Vygotsky [197] supports social interaction, rather than individual, in the learning process, where infants need certain social support to achieve developmental milestones. He also considers the learning and developmental process as continuous and varying from culture to culture. Conversely, Piaget [147] places emphasis more on the agent. He considers the learning process as universal among all cultures and discontinuous, involving various

stages. Where Piaget seems to be supporting the interactional approach of learning. He emphasises more on the organism rather than the environment in the developmental process [52].

Piaget sees learning as a development process for schemas using assimilation and accommodation processes to achieve (mental) equilibrium [147]. He proposes a theory of hierarchical development of knowledge, from ego-centric learning to the theory of mind. According to Piaget, development begins with reflexive behaviours, such as those often seen as play in infants [147].

1.3 Characteristics of Play

Humans are curious by nature. They tend to explore the surrounding environment and tend to solve problems out of curiosity, even if there is no material reward available for that, except the internal satisfaction [108]. This curiosity can be seen in young children as well. They tend to explore the surrounding environment, wherever possible [147, 106]. This behaviour in infants and children is seen as play behaviour.

Play behaviour is recognised as a very important step in mammals for motor and cognitive development [150, 198, 139, 200]. Play is often seen as a natural activity of infants and children [153]. The widely accepted definition for *Play* is a behaviour that gives pleasure. However, there are some other activities which provide pleasure such as feeding, which are not considered as play [198]. Hughes [76] defines five important characteristics for describing an activity as play, it must be; intrinsically motivated, freely chosen, pleasurable, non-literal (contortion of reality i.e., symbolic) and actively engaging. Under such characteristics, play is a physical behaviour without any constraints, except physical, which provides the internal satisfaction and pleasure. As the child develops, both physically and mentally, so does the style and complexity of the play behaviour. Play behaviour begins with free exploratory activity, then later becomes symbolic or pretend play, gradually incorporating increasingly

complex rules in early childhood. Play typically generates knowledge about the components of the environment in which the player/agent interacts and this is not considered as a goal-directed activity. However, an activity without a goal could be considered meaningless and aimless. Lee [105] considers play as a *goal-finding* rather than *goal-following* activity. Thus play behaviour could be seen as a goal creating rather than goal following activity that helps in the learning process.

The question arises, what makes infants play? This is a very complicated question. Play behaviours begin with exploration, where infants try to explore their immediate environment [139]. According to the Piaget [147], new situations arising in the environment intrigues existing knowledge and this imbalance causes certain behaviour. In this process, infants not only learn the actions and their effects but also learn properties of objects and the environment [191, 195]. Researchers believe that play begins with exploration in the environment to answer questions such as "What can it do?" or "What does it do?". With experiences and development, the question changes to "What can I do with it?" [139]. The curiosity of exploring sensorimotor capabilities and action that can be performed on the objects makes an infant motivated to play and most sensorimotor capabilities can be considered as such exploratory play behaviours.

Human or animal behaviours are driven by a certain necessity or desire of reward or to avoid punishment. In early infancy, infants are seen with reflexive behaviours and motor babbling. Motor babbling, by chance, helps to discover something new such as touching the nearby objects. Through repetitions, such behaviours become more controlled and co-ordinated with the sensory feedback. Piaget [147] considers reflexes as hereditary organic reactions and contact with external objects transforms these reflexes into play behaviours. Simply, he considers reflexes are the response to certain stimulus but still there are arguments over this definition. Most reflexes, such as sucking, swallowing etc., seem to be for biological needs. However, there are certain behaviours without a fixed goal [162]. Berlyne [20] considers "'Drive", a variable depending upon some internal and external factors causing

such behaviours.

In psychology, motivations are seen as either intrinsic or extrinsic. Extrinsic motivations trigger some behaviours for external reward or any biological need. Arousal and drive can be replaced with motivations [80]. It has been observed in infants that sometimes their behaviours are not just for a fixed need or reward and can be instead observed acting just for pleasure. Such behaviours are described as being intrinsically motivated. Oudeyer et al. [135] distinguishes between the two and considers that the extrinsic motivation results in learning actions that are needed for the body to be in a state of equilibrium, such as an infant suckling to fulfil its body's need, and intrinsic motivations generate actions depending upon their own success and give pleasure. For example, an infant sucking their hand or fingers even though he/she was recently provided a meal. This act cannot be considered as the reaction of a need but intrinsically motivated and just for comfort.

Including infants, adults are also seen intrinsically motivated for certain activities, for example solving crossword puzzles. Thus Intrinsic motivations can be considered as a cause for learning and building knowledge about the objects, actions and the behaviours. Now the question arises what causes such intrinsic motivations in infants? According to Berlyne [20] an increased random activity and restlessness can be observed in infants when they face new, strange or surprising stimuli and refers to this condition as "Perceptual" curiosity. He believes that curiosity derives certain behaviours in order to get any response from the stimuli. This refers to the novelty and change in the environment that seem to be the cause for intrinsic motivation. Stojanov [183] also favours the curiosity as the ability to learn without explicit reward, while McCall [117] believes novelty, change and ambiguity can generate such motivations that cause exploratory behaviours.

We summarise that the intrinsic motivations are driven by curiosity that is caused by novelty, ambiguity or change in the environment. Intrinsic motivations are responsible for the exploratory behaviours and such exploratory behaviours are seen as important characteristics

of the play in infancy, that trigger certain actions such as throw, squeeze, bite and grasp etc., on objects and help to learn new skills and affordances. The strength of curiosity, hence exploration, depends upon the response from the stimuli [19]. Novel response or change in response will drive the exploration further, whereas no change will reduce the curiosity drive.

Intrinsic motivations have significant importance in developmental robotics for driving the open-ended learning of robots. Appropriate algorithms for intrinsic motivations in computational modelling can enhance the learning approach in robots [12]. If robots have to work in unconstrained environments, they must have skills to face changes in the environment. Such skills can be learnt via intrinsically motivated activities [14], as humans do. Thus, a learning system, modelled on infant development, should contain a behaviour generating system driven by intrinsic motivations for exploratory and play behaviour. Intrinsic motivations in robotics can also be modelled and triggered by novelty and change in the environment.

1.4 Developmental Stages and Schema Mechanism

In the 19^{th} century, new techniques of testing enabled researchers to investigate infant development more deeply. Different researchers investigated the possible involvement of the environment and brain in acquiring motor skills. Piaget was one of those researchers who saw this relation and considered knowledge building as an interactional approach [149]. He developed the first known "Theory of cognitive development". His theory criticized the views at the time of a maturational approach starting with preprogrammed innate motor skills. Instead, he observed that as the human body undergoes physical changes starting from the fetus going through to adulthood, there should be different programs for each of the physical stages. The key points of this theory are as follows:

- Reflexes play an important role in motor and cognitive development.
- The development process involves motor actions and perceptions, and is a closed loop system.

- Development is staged. Breaking down development into a set of coarse stages, which are further broken down.

Piaget considers that individuals build knowledge about actions or objects through interaction and exploration in early infancy. Piaget describes a schema as a unit of knowledge, which develops in context and function over a period of time. The development of schemas begin with simpler schemas learnt from reflexive behaviours that are then used in different contexts within the environment [69]. The complexity of the schemas increases through the assimilation and accommodation process [147]. Contents of the schema, for any concept, may differ from agent to agent, depending upon the exposure, but the basic architecture remains the same [1]. Piaget's theory proposes hierarchical development of knowledge in schemas.

Schema based knowledge is developed over the course of time and from simple to complex knowledge. According to Piaget's theory when a human faces any situation successfully using their existing knowledge, the learning process is referred to as assimilation. During the assimilation new information is added into existing schema but previous content of the schemas remain unchanged. In the case of an expected outcome(s), an imbalance or disequilibrium occurs causing the agent to undergo the process of accommodation. Disequilibrium causes either new knowledge to be gained in the form of adding a new schema or updates the existing knowledge according to the information obtained [148]. During the accommodation process, either the context referred to as preconditions (a state of the environment for which the schema is applicable), the results referred to as postconditions or both components will be changed. Drescher [45], however, argues that during accommodation old schemas are not changed but new schemas are generated by adding new information within the old one.

Piaget divided cognitive development in humans into the different stages, each covering set age ranges. According to Piaget's studies, the human cognitive learning process is divided into four main stages; Sensorimotor, Pre-Operational, Concrete Operational and Formal Operational. These stages are briefly described as follows;

- Sensorimotor Stage (Birth-2 years)

  At the sensorimotor stage, infants develop knowledge about objects and the environment through physical activities. The knowledge extends to the mental representation of objects and building object affordances. At this stage learning is egocentric and the main achievement is to develop object permanence. During this stage, exploratory play behaviour is seen in infants. The model in the thesis is based on this stage of the development, hence we will discuss this stage in details in Section 1.4.1.

- Pre-Operational Stage (2-7 years)

  The Learning at this stage is still ego-centric, however, young children develop the ability to represent objects symbolically. During this stage children are still not able to use logic and only able to focus on a single aspect of the environment. Non-literal, symbolic or pretend, play is developed during this stage. For example, children use different objects like a cell phone.

- Concrete Operational Stage (7-11 years)

  During this stage children start thinking logically. They are able to mentally process actions, however, this capability is limited to physical objects only. During this stage, children also learn to conservation of physical quantities even if appearance changes.

- Formal Operational Stage (11 years and over)

  During this stage, the ability to think about concepts and build relationships between the objects and events. Children at this stage are able to think and reason about the events and processes which they never actually experienced.

Piaget considers these stages as universal, in-term of sequence and irrespective of the culture in which the child is raised [119]. Applying this staged and discontinuous learning process he believes one cannot change stage until he/she has obtained expertise in the current stage, however, there are the examples where learning and knowledge in infants have been found overlapping within the stages [68]. Development in different sensory and motor capabilities is considered due to learning and maturation of different systems (e.g., vision, cortical) supported by certain primitive capabilities [179]. The limitations in the sensory

and motor capabilities at an early age help to shape learning about the environment and their own capabilities in a hierarchical process, reducing the ambiguity and noise [100]. Thus maturation in biological and cortical systems over a period of time results in staged development. As the learning model developed in this work is based on the sensorimotor stage, hence we discuss this stage in details.

1.4.1 Sensorimotor Stage of the Development

The Sensorimotor stage of Piaget's theory extends from birth to 2 years of life and is further divided into six sub-stages; Reflex acts (0-1 months), Primary circular reactions (1-4 months), Secondary circular reactions (4-8 months), Coordinating secondary schemes (8-12 months), Tertiary circular reaction (12-18 months) and Symbolic Thoughts (18-24 months). In the first sub-stage infants show reflexive responses to external stimuli such as the "Palmer grasp", infants closing hand when something touches their palm. In this stage, infants learn behaviours using primitive actions and learn effects with own body parts i.e., arms, legs and hands. Such behaviours are repeatedly used in the second sub-stage of the sensorimotor stage. Sucking hand/fingers and fixating surrounding objects repeatedly during wake-time are examples of primary circular reactions. Infants' interactions are extended to objects present in the environment at the third sub-stage, secondary circular reactions. Infants at this stage repeat the actions on the objects to obtain interesting effects, for example shaking a rattle to produce sound, which are discovered and learnt by chance. At this stage play behaviour is seen to serve the purpose of answering the question "What does it do?" as discussed in the section 1.3.

Infants at the fourth sub-stage are able to create immediate goals and develop an ability to form simple plans. At this sub-stage, coordinating secondary schemes, infants seem to build concepts about objects and develop immediate goals. For example, retrieving an object by displacing or avoiding the obstacle placed between the hand and the object. Thus planning and small action sequences can be seen at this stage. Infants also extend their knowledge, learnt schemas, to new situations in this stage and demonstrate an ability for generalisation.

Piaget believes that behaviours observed at this stage are more likely due to interest in the object developed following previous experiences with objects [147]. At this stage play behaviour is seen to serve the purpose for the question "What can I do with it?" as discussed in Section 1.3 using existing schemas. With such behaviours performed on new objects, the effect either meet expectations or contradict with the schema that was applied. If the schema expectations are met, the used schema may be adapted, if additional details to be added. In the case of repetitions of similar conditions in the environment and using similar schemas then generalisation can be initiated. Generalised schemas help to extend the knowledge to novel situations and objects, without needing to learn different schemas for each environment and object [13, 201, 67, 111]. For example, if an infant pushes a ball accidentally and makes it move away. After a few repetitions with different objects, this action schema can be generalised as "pushing an object will cause the object to move away". Infants often generalise very quickly, however this can lead to schemas that are over-generalised. Baldwin et al. [13] found that 9-16 months old infant extend their behaviours towards novel objects to obtain non-obvious property, e.g., sound, based on their shape similarity. Welder and Graham [201] also found that 16-21 months old infants generalised their behaviour over the shape of the objects. They also found that infants used common labels as a cue to generalise behaviour and expectation for the novel objects, even though they had a different shape. The over-generalised schemas go through an accommodation process and new generalised schemas are created with deductive reasoning, a top-down approach of reasoning beginning with very generalised concepts, and moving towards specific examples. Thus, if the schema expectations are not met, the learning process undergoes the accommodation process and creates a new schema.

At the fifth stage, infants show a higher level of intelligence, as compared to previous stages. Infants at this stage demonstrate patterns of behaviours by using sequences of schemas [147]. Also, at this sub-stage infants find solutions to new problems by adapting their existing knowledge. Practice play, from secondary circular reactions and coordinating secondary schemes, is extended into sequences of play behaviours, e.g., joining and dis-

joining interconnecting play blocks. At the sixth sub-stage, infants show imitating behaviours. Infants at this stage are now able to form mental representations of objects, enabling pretend and symbolic play which can seen at this stage. This demonstrates the development of "object permanence", which is the milestone of Piaget's first sensorimotor stage [119].

In conclusion, during the sensorimotor stage, infants build egocentric knowledge about the environment through play behaviour, driving interactions with objects in the environment. The style of play observed during this stage develops from free exploratory play at the start, then on to practice play and finally reaching pretend play at the end of the stage. Sensorimotor knowledge, i.e. schemas, generated at this stage is hierarchical and developed from the infants' previous experiences.

1.5 Research Question and Objectives

To mimic a cognitive system i.e., humans, a robot should also be capable of learning and building knowledge as humans do. Humans learning starts with exploration, as discussed in Section 1.4. To mimic this capability in robots, their learning system[1] should be set-up with a learning and knowledge building mechanism inspired from human development. Dev-PSchema, the learning tool used in this work enables robots of learning through exploratory play. The specific research objective we address in this thesis is focused on developing knowledge using experiences gained through exploratory Play.

> Develop Dev-PSchema as an open-ended learning system to generate exploratory play behaviour that enables the discovery of novel experiences to extend the knowledge in novel situations. Furthermore, we extend the system for the structural development of skills and knowledge using experiences gained through play behaviour.

This objective leads to develop a learning model with the following questions:

i. Can a schema based model offer open-ended learning through exploratory play and be able to incorporate new information without any predefined template?

[1] We will use the terms system, model and agent interchangeably.

ii. Can a schema system develop generalisation structures, as observed in infants, through play behaviours and find a functional relationship in the generalisation?

iii. Can a schema system simulate infants with different preferences for actions in a given situation of the environment as infants do?

iv. Can a schema based system develop action sequences using basic schemas, through exploratory play, and utilise the sequences as high-level actions, as observed in humans?

v. Can an external user help to scaffold and shape the schema knowledge developed through play?

Based on the objective and research questions, we have extended PSchema to developed PSchema (Dev-PSchema) to achieve the contributions, introduced below in Section 1.6. We then conducted six different experiments to evaluate the performance of Dev-PSchema for each of the contributions.

1.6 Contributions

The aim of this thesis is to investigate developmental learning in artificial agents with play behaviour using an intrinsically motivated open-ended learning algorithm inspired by developmental psychology. The model will be evaluated in an agent embodied in two platforms, embodied in different environments. Initially a simple agent in a Sandbox simulation and later as part of a more complex agent on an iCub humanoid robot. Experiments provide a demonstration of acquired adaptive skills and behaviours through interactions with the environment, driven by novelty and curiosity. From here on the term "agent" will be used for the general learner, simulator or iCub, equipped with the learning system, unless specified.

This study is mainly concerned with the sensorimotor stage of development, as defined by Piaget, as the aim of this research is to generate a mechanism for modelling infant learning

through play and knowledge representation. At this stage infants are egocentric and their intelligence is developed by building knowledge about themselves and their environment using sensorimotor experience. In short, motor or physical activity and its effect on the environment is perceived in order to build knowledge blocks [110]. Drescher [45] supports Piaget's theory expanding on the details to form "Schema mechanism". According to this knowledge in the brain is developed in the shape of a *Schema*, that consists of context(s), action(s) and results(s). The actions are like moving, sweeping, grasping, seeing etc., while the context and results are the information obtained from the senses before and after the action respectively. Thus these schemas contain knowledge about the situations, objects and actions. For any situation in the environment, either a new schema is generated or an existing schema is applied.

Through a process of schema generalisation, it is possible to reduce computation and memory costs, thereby increasing performance. Furthermore, higher level schemas are created building on experiences gained whilst interacting with the environment. These higher level schemas (chains) combine the effects of different actions represented by lower level schemas to form more complex actions. Over time and through play, these higher order schemas are no longer considered as chains of low level actions, and instead become atomic actions.

The learning system introduced in this thesis, Dev-PSchema, is an enhanced version of "PSchema", see Chapter 3 for details. This thesis presents the mechanisms proposed, and experiments performed to investigate the development of the different attributes of learning during play behaviour in the artificial agents. The key contributions achieved in this research are outlined below. Each contribution is then discussed in the detail in Chapters 3 to 7.

1.6.1 Open-ended Learning and Adaptability

This capability is inspired by learning in humans. Humans are open-ended learners and learn through experience, adapting their understanding according to observed sensory feedback. For example, newborn babies have poor vision which develops over a period of time [177]. As the vision develops, infants incorporate and use the visual information in their learning. Adults have been observed to learn how to navigate using artificial vision systems after years of blindness [44]. Whilst there are some recognised limitations relating to sensitive periods in development [33, 130], where possible, humans are able to incorporate and adapt, to varying degrees, to new sensory information [44, 24]. Similarly, infants also learn different actions over a period of time, e.g., grasping and manipulating, pointing, feeding, walking, climbing etc. Such actions are discovered and learnt through exploratory behaviours, enabling the discovery of more capabilities and learning [177]. In short, humans are able to incorporate and use motor commands and sensory information whenever it is possible.

Artificial learning systems modelled on developmental psychology are expected to demonstrate open-ended learning, similar to that of humans. The model should be able to incorporate sensory information, perceived through sensors, in learning like humans do. Furthermore, the model should be able to discover and learn behaviours through exploration, which in turn enable the model to explore more capabilities.

PSchema [172], on which Dev-PSchema is built, used a specific format for specifying a fixed set of sensory perceptions and actions to interface containing predefined properties with the artificial agents. As such the system was unable to add any new sensory information which was not predefined. Nor it could learn or construct new actions. Due to this, PSchema system needed significant adaptations to enable it to be interfaced with different robotic platforms.

A key contribution to Dev-PSchema is the ability to use an abstract form for sensory perception and actions, enabling open-ended learning, hence addresses the research question

(i) of this thesis given in Section 1.5. This feature of Dev-PSchema expands its capabilities beyond what other relevant learning models have demonstrated, see Table 2.1. Furthermore, Dev-PSchema is able to interface with any agent with little or no change in the system. As an example of this, we interfaced the Dev-PSchema with the "SandBox", a simple simulator, and a humanoid robot, "iCub". All the sensory information in the system is represented as "Observations" e.g., colour observations, shape observations etc. Dev-PSchema is developed with abstract observation, which enables any sensory information received, to be represented and considered. Using the abstract observations, Dev-PSchema is able to receive and process sensory information of any format from the sensory system of the agents or robotic platforms. Moreover, abstract actions enable the agent to learn new actions on the fly, rather than be limited to pre-defined actions. For example, if the system generates a higher level action by combining low level actions, that new action can be incorporated and used on the fly.

1.6.2 Generalising from Experiences

This contribution is modelled on infants' behaviours, where they have been found to generalise their behaviours over different properties of the objects [67, 86, 180, 16, 203, 187]. Infants developed generalised understanding about objects and events using their knowledge and experiences, having some similar characteristics. This enables infants to predict behaviour outcomes in similar environments and situations. Infants are often seen overgeneralising [13, 67], however, they are able to learn exceptions [91, 112].

This capability will enable artificial agents to learn from a few examples and extend the learning for novel situations and objects, thus reducing consumption in computation power and memory. The agent, equipped with the learning system, Dev-PSchema, is able to generalise specific aspects of sensory properties of the environment following multiple similar experiences. We start the artificial agent with little knowledge about itself and being free to explore using its capabilities i.e., available actions. It learns different action-object pairs along with the change in the sensory feedback before and after the action, recorded

as schemas. The agent is able to generalise its experiences as it plays in the environment and explores the novel situations or objects using generalised experiences. The generalised schemas, help to predict outcome in similar situations in the environment.

The key contribution here is for the agent to be able to partially generalise experiences, enabling the agent to learn exceptions in the generalised schemas. The agent is also able to find linear functional relationships between the different properties by identifying additive relationships between the numerical value of the properties in the sensory information. This capability enables the agent to predict the numerical values of the properties as an effect of an action. This contribution of Dev-PSchema addresses the research question (ii) of this thesis, see Section 1.5. Although different learning models demonstrate generalisation capability [6, 172, 141, 140], the functional generalisation has not been demonstrated in other related learning systems, see Table 2.1.

The research here has been presented in two papers, [95, 94], that have been published as the part of this study related to the generalisation mechanism. One paper focuses on the use of partial generalisation during play and the other on the functional generalisation.

1.6.3 Simulating Individual Variations

In developmental psychology experiments results often focus on the average of behaviours observed in infants [77, 57]. However, each individual shows variation in behaviour in part due to individual preferences. Developmental psychology inspired robotic learning system should also be able to demonstrate some such variations in the behaviours, due to individual preferences. This capability can be used to test hypotheses from developmental psychology.

Roboticists, generally, focus on developing a learning system with specific behaviour and knowledge. Which, in short, is try to reproduce a behaviour averaged from several individuals. Dev-PSchema is able to simulate different individual infants by tuning different excitation

parameters. It acts as an intrinsically motivated play generator with internal preferences. The excitation mechanism calculates the excitation of each action in its memory, based on its experiences, the perceived environment and its internal preference.

Behaviours of the agent used in this work are based on three parameters related to familiarity and similarity of the perceived environment, and previous experiences, see Chapter 5. Each of these parameters is weighted with user defined weights. Changes in the weights enable agents to simulate different individual behaviours and show different preferences for actions in a given environment. For example, providing higher weight to the similarity factor will enable the agent to interact more with the objects similar to that which have been interacted with previously, see Section 5.2.1. Similarly, less weight to familiarity, hence more to the novelty factor will encourage the agent to interact with less familiar objects or those which have not been interacted with for a while. Thus this feature enables the system to demonstrate the possibility of simulating different individuals, rather than an average individual, hence addresses the research question (iii) of this work, see Section 1.5. This capability in Dev-PSchema goes beyond the capabilities demonstrated by other related learning models, discussed in Chapter 2, as they tend to model an average behaviour.

1.6.4 Forming Higher Level Actions

Most actions performed by humans on a day to day basis can be defined by high-level actions and objectives. These actions/objectives are typically achieved by a series of low level motor actions or a sequence of actions, referred to as primitive actions. For example, drinking water is a high-level objective, which can be achieved by a sequence of actions such as; reaching for the glass, grasping it, filling it with water, opening of mouth while transporting the glass to the mouth, and adjusting the angle of the glass in the mouth to enable comfortable drinking. In this example, a sequence of lower level or primitive actions are executed to achieve the overall objective. These primitive actions are continuous and often inseparable from each other whilst predominately maintaining the sequential ordering, with some occasional overlap

between actions. Simple action planning is observed in infants as young as 8-12 months old, as discussed in the Section 1.4. Thus infants are able to plan higher level gaols at an early age. For example, reaching for an object and displacing or avoiding any obstacle in the path.

Action sequences, in developmental psychology, are seen either as reflexive chains or motor programs [160, 98]. During a reflexive chain, a pause between actions, in action sequences, occurs to obtain feedback from the environment before proceeding to the next action in the sequence. After a few repetitions successful action sequences become motor programs, where they are executed without any pause between actions for sensory feedback.

Robots may also need to achieve objectives involving a sequence of primitive actions in a given environment. To achieve such an objective, the robot will need to form a plan by identifying the necessary steps before execution. The objective may be defined by an instructor, for example a human, or set by the robot itself, based on observations and experiences. An added contribution to Dev-PSchema is an extension to its chaining mechanism, enabling it to create a sequence of schemas containing related sensory information. This mechanism enables the agent to build higher level, complex, actions by sequencing low level actions and learning hierarchical structures of schemas. Although some other learning models have demonstrated the capability to develop high-level actions (see Table 2.1) through action sequencing. Dev-PSchema develops high-level actions as reflexive chains and motor programs, attempting to model the development of complex behaviours in humans [98], where sequences of primitive actions are transformed into a single high-level action, see Chapter 6 for details. This contribution serves the research question (iv) of this thesis, see Section 1.5.

Furthermore, this can be seen as opportunity for an external agent to shape the agent's knowledge through providing an objective state to achieve. For example, "Reach" and "Grasp" actions can be combined to form the higher level-level action "Reach and Grasp" through staged development. This capability will help to address the research question (v) of

this thesis, given in Section 1.5. The chaining system also serves the purpose of achieving a higher-level target sensory state[2] by performing a sequence of actions in a given environment to bring about that state. Multiple actions can be combined and executed in sequence to achieve a target in the environment which is, otherwise, not possible to achieve with a single action.

Following all the updates in PSchema, Dev-PSchema acts as a play generator for an artificial agent acting as a simulated infant. During continuous play, the agent demonstrates the capability of exploration, utilisation and exploitation, of existing knowledge. The agent explores the surrounding environment and finds the object-action relationships by playing in the environment. Furthermore, the agent re-uses the experiences while utilising existing knowledge on perceived objects. To demonstrate the exploration and exploitation capabilities of the agent we performed experiments on a simple simulator *SandBox* and a real robot iCub. Both agents demonstrated the capability of exploration and exploitation during the experiments in a given environment.

1.7 Thesis Structure

An overview of each of the following chapters in this thesis is given below:

Chapter 2: Developmental and Cognitive Robotics

In Chapter 2 we present literature review on developmental psychology inspired learning models for artificial agents. We begin with artificial neural networks (ANN) and brain simulations inspired from neuroscience. Later we introduce developmental psychology inspired learning models and finally narrow down our literature review schema based learning systems. The main focus of this chapter will be learning models inspired from Piaget's cognitive theory

[2]We will use the terms sensory state, environmental state and the state interchangeably throughout this thesis.

and its sensorimotor stage.

Chapter 3: Play Generator: Dev-PSchema

In Chapter 3 we introduce the learning model used in this work. We discuss different aspect of developmental learning in infants and related modelling used in the system in parallel. We begin with a little introduction of PSchema, a learning system on which the system used in this work is built upon. Then we discuss components of this learning framework in details including schema representation and creation, excitation mechanism, chaining mechanism and generalisation mechanism.

Chapter 4: Generalising from Experiences

In Chapter 4 we demonstrate the generalisation capability of the system with experimental results. We performed two experiments, each for partial and functional generalisation. In the first experiment we demonstrate that the agent partially generalises its experiences as it interacts with the environment. In this experiment objects are introduced in the environment and the agent is free to interact with them using primitive actions. The agent generalises its experiences using the outcomes of the interactions. The second experiment is to demonstrate the functional generalisation in agent's experiences. The agent interacts with the environment using visual activity and generalises its experiences.

This chapter is related to the contributions discussed in the Section 1.6.2.

Chapter 5: Simulated Infants and Play Behaviour

In Chapter 5 we demonstrate the capability of the system as simulating individual infants using two experiments with a simulator, SandBox. The first experiment demonstrates the variations in preference for object interactions by tuning different parameters of the excitation

mechanism in Dev-PSchema.

The second experiment demonstrates the variations in preference for actions to play in a given environment by tuning the parameters of the excitation system. These experiments demonstrate play behaviour in simulated infants. This chapter is related to the contribution discussed in the Section 1.6.3.

Chapter 6: Forming Higher Level Actions

In Chapter 6 we present two experiments to demonstrate learning of high-level actions through schema chaining/actions sequencing. We used a simulator and real robot to demonstrate the chaining capability of the system. In the first experiment we let the agent play and demonstrates the capability of the system to explore the environment using intrinsic motivations. The agent discovers the higher level actions thorough play and utilises to achieve the objective set by the agent itself.

The second experiment demonstrates the learning of motor programs, as introduced in Section 1.6.4. The agent learns an interesting effect with a sequence of actions and tries to achieve the effect repetitively. With repetitions, the sequence becomes a motor program, which can be executed without any pause for sensory feedback. We test the motor program in three test conditions, to demonstrate the successful use of a motor program and adaptability in case of failure.

Chapter 7: Shaping Learning

In Chapter 7 we demonstrate scaffolding and shaping of the knowledge in Dev-PSchema. We conducted two experiments to demonstrate these capabilities of the system. In the first experiment, we let the agent play in its environment to gain specific skills with the objects through exploration. The complexity of the skills increases throughout the experiment. Once

the agent gains the relevant experience, which takes more than one action to achieve, we ask the agent to achieve the learnt experience with another object. The agent responds with all possible solutions for the given problem and executes the user selected solution.

In the second experiment we let the agent explore and learn associations of an action with an objects as tool use. Moving the object to a defined location causes a novel object, a toy, to appear. Once the agent learns this association, the agent is provided an objective to bring the toy back in the environment through the learnt association. The agent responds with the possible solutions for this problem.

Chapter 8: Conclusion and Future Developments

In Chapter 8 we evaluate the contributions made in the thesis based on the experimental results and the literature from developmental psychology. We discuss about the adaptability within the system and further development of the learning system used in this work.

1.8 Publications

Parts of this thesis have been published in the following conferences and journal.

- Kumar, S., Shaw, P., Lewkowicz, D., Giagkos, A., Shen, Q., Lee, M. (2016, September). Developing object understanding through schema generalisation. In Developmental and Learning and Epigenetic Robotics (ICDL-EpiRob), 2016 Joint IEEE International Conference on (pp. 33-38). IEEE.

- Kumar, S., Shaw, P., Lewkowicz, D., Giagkos, A., Lee, M., Shen, Q. 2016. Generalising Predictable Object Movements Through Experience Using Schemas. In E. Tuci., A. Giagkos., M. Wilson., J. Hallam. (eds) From Animals to Animats 14: 14th International Conference on Simulation of Adaptive Behaviour, SAB 2016, Aberystwyth,

UK, August 23-26, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9825 14th International Conference on Simulation of Adaptive Behaviour. Springer Nature pp. 329-339.

- Kumar S, Shaw P, Giagkos A, Braud R, Lee MH, Shen Q. Developing hierarchical schemas and building schema chains through practice play behaviour. Frontiers in Neurorobotics. 2018;12:33.

Chapter 2

Developmental and Cognitive Robotics

If an unsupervised robot needs to act in a natural environment, it needs to learn and build knowledge about the environment it is working in. This knowledge should not be restricted to the agent's behaviours only but also needs to understand the effect of its actions on the objects. There should be a mechanism that processes information to learn and build knowledge and make it useful systemically. There are different theories regarding the learning and information processing in humans. Similarly, different methods have been introduced and used by various researchers in the field of robotics for learning in artificial agents. We begin our discussion with neuroscience and learning systems inspired from it. Later we discuss different learning methods for robots followed by a detailed discussion on developmental robotics and *schema* based learning systems. Finally, we provide a summary of characteristics for a learning model inspired from developmental psychology.

2.1 Neuroscience & Modelling

Initial approaches to make machines intelligent involves replicating human-like information processing in machines. The very fundamental part of the human brain is neurons, which performs information processing. Artificial Neural Networks (ANN), in artificial brains, are part of the efforts to replicate human brain processing. It is assumed that ANN, a computational

system, will replicate the characteristics possessed by the biological neurons. Learning in ANN is adapted by changing network architecture and weights of the network connections. Moreover, in ANN learning is based on a predefined set of examples of the specific task that need to be learned. In ANNs learning is referred to as training. ANNs are trained with a set of examples before application. There are three main learning paradigms in ANN; supervised or reinforced, unsupervised and hybrid. In supervised learning, an ANN is trained with input values and error between the expected outcome and the obtained outcome. Weights of the network are adjusted according to the error during the training process and the process is continued until the error disappears or decreases to a threshold level. In unsupervised learning process ANNs are trained without predefined results. This method is suitable for clustering data using statistical properties. In hybrid learning, which is a combination of supervised and unsupervised approaches, some of the output labels are provided by the user. Deep learning algorithms, ANN with many layers and specialized computational units, are able to extract features from the inputs autonomously rather than requiring them to be provided by the user. Thus deep learning algorithms are able to learn high-level concepts (classes) from simpler ones (features) [65].

There are certain limitations in the ANNs and ANN based Deep learning, some of them are listed below:

- Power, data and time

  For an ANN to be precise and provide accurate solutions it needs to be trained with a large set of examples [15]. Thus an ANN can take a large number of examples to learn. To train with a large number of examples, ANNs need more time and more computational power as well to be trained. Even greater computational power and significantly more time are required for training deep learning algorithms [205].

- Context sensitivity

  ANNs, including deep learning networks, lack the capability to expand their knowledge, where learning in one context or situation cannot be utilised properly in a different

context/situation. Deep learning networks, in particular, are over-sensitive to training context for planning, acting and reasoning [205]. Deep networks misidentify the objects if they are in a different context than they were observed during the training.

- Black-box model

  ANNs are black box models, hence it is not possible to analyse the relationship between inputs and outputs. Although, in some cases, an ANN may provide perfect solutions for a problem, it is not possible to represent the relationship between inputs and outputs. In some cases it may be required to check the relationships between input and output, in which case it is not possible to make use of ANNs. Thus, ANNs are still unable to develop internal models for the physical world in the same humans do, so are therefore unable to reason and develop logic about the world [104, 97].

- Structural methods

  ANNs have no structured methods to find optimum network configuration and parameters [15, 169]. For the same problem, two or more ANNs with different structures can be used with different weights and each ANN may lead to different approaches to reach the solution for a given problem.

- Separate learning

  ANN, mostly, learns each problem separately and a single network can solve one problem only. There are some recent approaches introduced in the literature that help to solve multiple objective/problems using a single network. To develop human-like intelligence, an artificial intelligent system should be able to learn new concepts through utilising existing knowledge, rather than re-learn from scratch each time [97].

Even though ANNs have certain limitations, they are still suitable for prediction and pattern recognition problems [206, 78]. However, our focus here is on learning models related to general artificial intelligence rather than task specific knowledge, whereas ANNs are more suitable for task specific learning. Building on some of the underlying principles of ANNs, there are various projects aimed at developing simulations of neural networks on

the scale of the human brain. These are biologically inspired artificial brain architectures, some of which are starting to produce interesting results in the research field. Below is the summary of some of the major projects:

- *Blue Brain* project is a biologically inspired research project to simulate the neural processing of the human brain [40], to study the emergence of intelligence [114]. The project officially started in July 2005 and was aimed at achieving the goal by 2018. This project, as assumed, aims to be able to simulate the processing of any part of the human brain if specific information is provided. The main focus of the project is to develop simulation of the cerebral cortex, a part of the human brain responsible for a person's ability to remember, think, reflect, empathize, communicate, adapt to new situations, and plan for the future [40]. *Blue Gene*, a System on a Chip (SoC) is one of the outcomes of the project. The SoC can simulate 10,000 neurons that make up a neo-cortical column in the human brain. Although, the brain model of the project may help to understand the functioning and diseases of the human brain, and help to build supercomputers, the development of an intelligent agent with human-like learning is a completely different task, irrelevant to the project. Furthermore, *Blue Gene* is more power consuming system as compared to other similar architectures [40].

- *Human Brain Project* (HBP) is a project similar to Blue Brain project was launched by the European Commission's Future and Emerging Technologies (FET) scheme in 2013. The main aim of this project was to simulate human brain and develop brain inspired computing [115]. The researchers believe that the project will help to understand brain functions, from a single neurons to a whole brain, and help to understand and develop drugs for different brain diseases. *SpiNNaker* chips are the result of such computer architecture [56]. It is a multi-layer (core) parallel SoC, where each core is designed to simulate a network of up to 1,000 simple spiking neurons, each with 1,000 synaptic inputs [21]. SpiNNaker used predefined lookup table functions loaded into each core's

local memory. Also, the network dynamics are specified by the user [124]. Despite these limitations, that cause the chip to be limited in its functions, it can still be used for various applications including robot control, vision processing and modelling of biological circuits [55].

- IBM, with a group of academic researchers, also developed a brain inspired computer chip, TrueNorth, a very dense, energy-efficient platform capable of supporting a range of cognitive applications [55]. The chip is a result of Neuromorphic Adaptive Plastic Scalable Electronics (SyNAPSE) project launched in 2008 by Defence Advanced Research Projects Agency (DARPA). The chip contains one million programmable neurons, and 256 million programmable connections (synapse) between the neurons [42, 7]. The chip is programmed by specifying the behaviours of the neuron in the architecture and the connectivity between them [7]. TrueNorth has been demonstrated in a real-time object recognition application running at very low power [121]. Although TrueNorth offers the capability to simulate a large number of neurons with very small power consumption, it is limited in real time neural plasticity [55].

- *Neurogrid*, a project by researchers at Stanford, is another approach to simulate neural activity [17]. Neurogrid is a piece of computer hardware that simulates human brain activity. It contains millions of neurons and billions of connections between them. The research project claimed to simulate multiple cortical areas in real time [88]. Although the system offers extremely low power neural network simulation in real time, the architecture only allows the simulation of a limited number of cortical areas, rather than the entire brain and does not offer real time plasticity of neurons in their functions [55, 17].

Projects discussed above represent large scale brain simulation and are of great importance in the research field of artificial intelligence. Advancements in the neuroscience are

helping to understand how different parts of the human brain work and nanotechnology has enabled researchers to develop microchips that contain a large number of processing units in a single chip. Even though, these projects produced some interesting results, they still lacked human-like learning [63]. These brain simulators can begin with any arbitrary age, rather than beginning with no intelligence as humans do. Although, these chips and computers contain billions of transistors and can simulate millions of neurons, they are still a long way from the human brain which contains billions of neurons. As of yet, it is not possible to develop to this scale with manageable size and power consumption.

Apart from the limitations in size and power consumption, brain simulations demonstrate abstract intelligence rather than embodied architectures to enable behaviours as humans do. Embodiment is emphasised in artificial intelligence in order to develop human-like intelligence [28, 26, 143, 8]. Furthermore, the use of humanoid robots is encouraged in learning paradigms for developing intelligent robots [10, 27, 164]. The humanoid shape in robots helps to provide a similar level of constraints in the environment as humans experience.

2.2 Programming Machines and Robots

To develop an intelligent robot beginning with no or little intelligence, it should contain a programmable memory. This memory stores solutions for different problems and possible outcomes of the behaviours in the environment. Traditionally, there are three main programming methods for a robot's behaviours and developing an intelligent robot; direct programming, evolutionary learning and supervised learning [120].

2.2.1 Direct Programming

In direct programming the robot simply executes the pre-programmed behaviours and commands. Human(s) implement the solutions, in a program, for a given problem that a robot

need to solve and equip the robot [120]. Any learning in the program, if required, is performed off-line, before the robot is equipped with the program [126]. There are several examples of direct programming in robotics applications [75, 152]. Huang et al. [75] demonstrated biped walking in a robot through direct programming the walking cycle, and foot and hip trajectories. Similarly, Pounds et al. [152] demonstrated flight of a quad-rotor. The authors demonstrated the stable flight of the machine with a payload of 1 kg through designing the control mechanism to stabilise the altitude, pitch and roll. In both examples here, the environment dynamics has been assumed through selecting and modelling suitable parameters of the environment. A directly programmed robot demonstrates good performance in an environment for which it is designed, however it is brittle to changes in the environment and situations.

2.2.2 Supervised Learning

In supervised learning, robots are trained to respond with using a set of training data, provided as sensory input. The robot is trained with a limited set of problem representations and the associated output is provided for each set of the input training set. With this approach, robots are able to solve similar problems to those they are trained on and demonstrate some level of generalisation [113]. However, robots can only be trained on a small set of tasks, hence will not be able to learn general-purpose strategies for a variety of problems. Apart from that, the robots only learn during training, hence do not demonstrate continuous learning. Several supervised learning techniques including shaping, local reinforcement signals, imitation etc., have been used in artificial intelligence and robotics problems [82], such as [151, 99]. Pinto et al. [151] demonstrated learning a visual representation of objects through interactions with deep learning network. The robot was trained by interacting with the objects through grasp, push and poke actions. The experiments performed provided interesting results in recognising objects similar to those experienced previously. Although the network shows interesting classification ability, 72%, it was trained with a large number of examples, over 130K. Similarly, Laud and DeJong [99] demonstrated a flight control network for a quad-

copter through supervisory learning. The network was trained with 512 initial and 1024 branching trajectories with approximately a million steps per iteration.

In both examples discussed above, the learning systems are trained with a large number of problem examples before the operation. During the operations, the learning system just acts as a control system, without further learning while solving a problem similar to those it was trained with.

2.2.3 Evolutionary Learning

Following an evolutionary approach, performance of a population of randomly generated robotic controllers is evaluated on a fitness function, developed by humans. The highest performing controllers are allowed to reproduce with another to create a next generation of robotic controllers. This process is repeated until a better robotic controller is obtained. Due to the use of abstract fitness functions, evolutionary algorithms can provide task independent learning as compared to traditional supervised learning approaches. However, the level of generality is very limited. Furthermore, evolutionary algorithms require a large number of trials, like ANNs, to find a set of controllers that fit best with the fitness function. Selecting a suitable evolutionary algorithm and its fitness function for a robotic application also limits the capability of the evolutionary approach [123]. A few examples of evolutionary learning are given in [189, 36, 184].

Chernova and Veloso [36] demonstrated autonomously optimizing fast forward gaits in quadruped robots through evolutionary algorithms. The approach resulted in a 20% increase in the speed of their robot's walking motion as compared to previous walking motion. Similarly Sugihara and Smith [184] demonstrated path and trajectory planning in a simulated mobile robot through Genetic Algorithms (GAs). The robot is tasked to find the optimum path to the goal position, avoiding solid obstacles that block the motion and hazardous obstacles that cause an increase in path length. The robot performed the task with an optimum path,

however, the accuracy in finding the optimum path decreases with the number of obstacles in the environment. In both examples here, the learning systems were trained over a large number of iterations to achieve the generation with the highest fitness function.

Although the direct programming approach is suitable and can show higher performance for task oriented problems, this approach lacks the key aspect of human-like learning, *Open-ended* and *Developmental*. These capabilities enable humans to learn throughout their life. Thus this approach is excluded when modelling human-like learning in robots. The remaining two approaches demonstrate certain level of task independence and generality, however these approaches can only learn a few small steps from the starting point [120]. For example in supervisory learning knowledge produced in one set of example may be used in other similar situations. However, transferring knowledge from one task to another is not achievable [90]. Similarly, evolutionary algorithms show a level of generality in performing task in different situations, however they can still only perform a limited number of tasks, depending upon the fitness function [127]. This motivated researchers to explore in different directions and resulted in another community in the field of artificial cognition, *Developmental Robotics*.

2.3 Introduction to Developmental Robotics

Under developmental robotics, robots develop their knowledge in an independent developmental process. In a developmental system, the aim is to build a robot to continuously develops its knowledge, on top of its existing knowledge, by putting itself into novel situations. Developmental robotics is an interdisciplinary field, combines developmental psychology and robotics, emphasising on developing learning in robots inspired by human development starting from little or no intelligence, like infants. The systems modelled on human-like learning and reasoning are often described as *Cognitive systems*.

In cognitive systems, learning is based on the interaction with the environment. Humans, as a cognitive system, learn by interaction with in the surroundings [192, 147]. To build a human-like intelligence in a machine, it is needed to develop, learn and adapt like humans do. Although, the idea of such a research field as cognitive developmental robotics was proposed some time back [188], it only formally emerged in 2000 [31]. Developmental or cognitive robotics is a branch of robotics in which an agent learns its capabilities and their effects on the environment rather than having them hard-wired within it. Thus learning is neither task-oriented not domain specific, unlike ANNs. Learning in developmental robotics is seen as constructive, increasing the knowledge by building on existing understanding.

2.3.1 Developmental Learning Paradigms

As discussed, the aim of a developmental system is that the agent extends its knowledge by building on the existing knowledge. The learning is open-ended, independent of other agents and task independent. An agent, with an effective developmental system discovers new behaviours and knowledge, which in turn help to explore further thereby extending its learning. There are two main learning paradigms used in developmental systems; Goal-directed and Autonomous.

2.3.1.1 Goal-Directed Learning

In goal-directed developmental learning, a series of increasingly complex goals are set by the human developer to make agent learn a specific behaviour and task. The agent is initially also provided with a limited set of behaviours and knowledge to learn a specific task. The agent may be provided with same task specific capabilities and the environment and tasks structured to support learning, however the agent still learns by itself making decisions, without further human interference. Should the agent need to learn new different skills in the future, a similar process is followed, guiding the agent through a series of tasks to build on

its existing knowledge.

A Human operator or instructor, as an external agent, helps the robot to achieve and learn a goal in the environment. This *help* is provided either by giving feedback to the agent at the end of each trial during the task learning (supervisory learning) or by directing or demonstrating the task (social learning). Different learning strategies can be used for implementing goal-directed learning including social learning (i.e., social interaction, imitation etc.) [141, 125] and supervisory learning [204, 83], depending upon the structure of the learning model.

2.3.1.2 Autonomous Development-Open-Ended Learning

In the autonomous learning paradigm, an agent is equipped with a system aiming to enable independent learning before being placed in an environment to act and learn. The starting point may be seen as the "birth" stage of the developmental learning robot [202]. The environmental may be altered by humans to help the agent learn a specific task, however, the learning path is chosen by the agent itself.

Autonomous learning aims to allow for open-ended learning, where behaviours, learning objectives and learning strategies are set by the agent itself, rather than the external user. Depending upon the learning model, different learning methods can be implemented for autonomous developmental learning such as intrinsically motivated exploratory play [173, 46, 6, 136] and randomly generated behaviours [165, 48]. In an intrinsically motived learning system, the agent decides the behaviour based on the perceived sensory information and internal model of the system. Whereas in randomly generated behaviours, the agent performs the actions at random and learns from obtained results.

2.3.2 Intrinsically Motivated Learning Systems

Many researchers are working in the field of developmental robotics using different simulated and physical robotic platforms such as iCub, NAO, ASIMO, COG, CASPER etc. Several learning systems have proposed to learn and adapt from active and passive experiences, either by exploration, direction or demonstration, including Planning Domain Definition Language (PDDL) framework [141], Object-Action-Complexes (OACs) [204, 92], Schema networks [83], PSchema [172, 173], Developmental Engagement-Reflection(Dev-ER) [6, 5], Intelligent Adaptive Curiosity (IAC) [136, 84, 134], Constructivist Anticipatory Learning Mechanism (CALM) [140], Constructive Learning Architecture Schema Mechanism (CLASM) [35] and Intrinsic Curiosity Module (ICM) [137]. This subset in developmental learning systems based on schema-like representation for knowledge containing perceptual state before an action, the action and either perceptual state after the action or changes in the perception as in [172, 141, 204, 83, 6, 136, 140, 70]. However, ICM uses a deep learning network to learn the feature module from the training set and learn predictions [137]. The term schema is referred to as a unit of knowledge, introduced by Piaget in cognitive developmental theory [147]. According to this theory infants learn through active experiences during play triggered by intrinsic motivations. Novelty in the environment is one of the key factors responsible for intrinsic motivations, as discussed in Chapter 1. Repetitive environments/outcomes/objects make infants habituated, hence they get less interested in further interaction. The habituation paradigm is widely used in developmental psychology to investigate infants' interests in the environment and understanding about relationships between different characteristics in the perceived environment. The habituation paradigm is further discussed in Chapter 5.

As our main goal, in this thesis, is developing a learning system that builds knowledge through intrinsically motivated active exploratory play, we keep our focus on schema and schema-like leaning systems which develop through play behaviours. In Chapter 1 we discussed characteristics of play and learning, including intrinsic motivations, open-ended and independent behaviours, generalisations, predictions and developing complex skills. Here

we discuss various intrinsically motivated learning systems by considering the characteristics of the systems in turn.

2.3.2.1 Knowledge Representation

Learning models use different mechanisms to represent knowledge containing sensory perceptions and actions. Oudeyer et al. [136] introduced an intrinsically motivated learning model called "Intelligent Adaptive Curiosity" (IAC). The system represents the knowledge in *experts*, with a structure similar to the schemas in [172, 6, 140]. However, schemas can be either generalised or concrete instances, rather than just concrete experts [136, 84, 134]. An intermediary system in IAC transforms the low-level sensory information into the high-level binary visual and proprioceptive perceptions i.e., object presence, oscillation and biting. Whereas PSchema and Dev-ER models use high-level sensory perceptions having predefined attributes with different values rather than binary as in IAC. The visual perceptions in PSchema and Dev-ER are developed through experiences separately. Furthermore, As compared to PSchema and Dev-ER which use high-level action representation e.g. reach, look up etc., IAC is capable of using intermediary level motor commands e.g. pan and tilt control, bash strength and angle etc. Through explorations, the robot develops its knowledge which is further used to explore the environment. The exploration, action selection, mechanism is further discussed in 2.3.2.2.

PSchema, Dev-ER, CALM and CLASM, begin development through basic schemas, action Self Organising Modules (SMO) for CLASM, containing either preconditions [6] or postconditions [172] or just an action [140, 35]. Whereas the IAC model begins development with motor commands, rather than the expert templates. Experts in IAC are represented in a tree structure combining several instances in a single expert. A new instance of an expert is developed if it failed to predict its action effects. An expert is divided into two when it reaches a threshold of its instances, i.e., 250 [136]. Whereas in PSchema, Dev-ER and CALM, each experience instance, schema, is recorded separately in the memory. A new schema is added into

the memory if any of the existing schemas in the memory failed to match the state, on which an action is performed, with schema preconditions or postconditions with the obtained results.

Some other learning models, including [141, 125, 83, 92, 137, 35], have demonstrated learning with either schemas or schema-like knowledge representation. These learning systems have been demonstrated to solve problems using the knowledge developed through active and passive experiences. However these learning models initially develop knowledge through either supervised experiences [141, 92] or performing random actions [125, 83, 35], unlike learning models in [172, 6, 136, 140] which develop knowledge through intrinsically motivated exploratory actions. Thus limiting the learning resources for the model. The developed knowledge is used further to solve a problem following a demonstration through imitation [141, 125, 92] or planning actions to maximise the reward for the goal state which is explicitly defined in the system [83, 137].

The learning models in [141, 125, 83, 92] develop the knowledge through repetitive experiences triggered by the experimenters. The models develop knowledge through a large number of examples and represent the learning, containing a state, actions and its effects on the state, including Markov Decision Process (MDP) based schema networks [83], Bayesian Networks (BN) [125], Episodic-Like memory (ELM) [141] or any of the reinforced learning methods and BN [204, 92].

2.3.2.2 Action Selection

Action selection selects a suitable action to act in its environment from a pool of actions. The mechanism is an important attribute of learning models inspired from developmental psychology. Intrinsic motivations are seen a basic driver for exploratory behaviours in infants, as discussed in Chapter 1. Performing the selected action in the environment may cause changes that will lead the learning model to develop new knowledge in its representation.

The learning models, PSchema, Dev-ER and IAC, use internal function(s) as motivation for action selection in their models, hence can be referred to as intrinsically motived systems. Drescher's proposed schema mechanism contains a schema, hence action, selection algorithm based on intrinsic motivations, rather than any goal status. The action selection mechanism is based on the goal pursuit and exploration. The goal pursuit is based on the predictions of the schema and enable the model to achieve higher goal, whereas the exploration factor helps to explore the environment [46]. Sheldon's PSchema model [172], an implementation of Drescher's schema mechanism, contains action selection mechanism based on the schema excitation calculation. The excitation based on the schema predictions through postconditions and the schema statistics to trigger exploration in a given environment. Thus the model demonstrates a balanced action selection between exploration and predictive behaviours.

Similar to PSchema, in Dev-ER intrinsic motivations are simulated through an internal attention mechanism [6]. However, the mechanism uses a list of variables for the attention mechanism rather than just a single excitation calculation as in PSchema. The mechanism uses variables representing pleasure and displeasure, emotional states of interest, surprise and boredom and cognitive curiosity. The pleasure emotion is simulated through visual preferences. The agent's pleasure shifts from level 1 to 2 when a luminous object is moved from peripheral to centre of its vision. It gets displeasure when the object of interest is lost. Similarly, emotional states of the agent, represented through three boolean variables; interest, surprise and boredom, which depend upon the object of interest and its presence it the visual field. The *interest* state for an object is set *True* if the agent tends to focus on the object. Similarly, surprise and boredom states are set *True* if the object of interest is recovered visually or stays in the vision for longer respectively. The surprise state causes the agent to develop new schemas with changing its state from displeasure to surprise between preconditions and postconditions. Thus the new schema predicts that the action in the schema will bring the object of interest into the visual field. However, if the action fails, the curiosity state triggers adaptation in the executed schema.

As compared to PSchema and Dev-ER models which use action selection balanced between predictive behaviours and explorations, IAC model tends to perform more exploratory behaviours. In IAC model [136], initially, actions are selected randomly to develop some representations through observing the action effects. It uses Learning Process (LP) as a cue for the action selection. LP is calculated through commutative errors in predictions for each action. Large error represents lack on ability by the expert to predict the outcome, hence leading the agent to explore with the actions that produce unpredictable results. Thus the action selection mechanism of the model depends upon the unpredictability of the environment, hence it is more likely to explore the environment through selecting less predictable actions.

In CLASM model [35], the agent selects the actions randomly to interact with its environment. Whereas, CALM model [140] action selection is based on the internal state of the agent, hence can be referred to as an intrinsically motivated model. Whereas, in ICM model action selection is based on the predictions of the model. The more unpredictable the environment is, the higher curiosity hence intrinsic motivation are.

In Dev-PSchema action selection is depends upon the perceived environment and previous experiences of the agent. The action selection is tunable through changing the weights for excitation parameters. This enables the Dev-PSchema to demonstrate different action selection in the same perceived environment. None of the above discussed studies demonstrate this capability.

2.3.2.3 Abstraction

A learning model inspired from developmental psychology should be able to process new perceptions as sensory and motor capabilities are developed. Thus the learning should be able to develop knowledge through processing new perceptions. To do such, the model should be using abstract representation of actions perceptions to develop the knowledge.

The learning models in [172, 6, 136, 140, 35] use a list of attributes for perceived sensory information as a perceptual representation. Thus to incorporate any new sensory perception, the models either should be provided with perceptual definitions, as seen in [172], or use existing attribute representations [6, 136, 140, 35, 137]. Similar to the sensory perceptions, the above discussed models use predefined actions with fixed attributes. Thus the models will need a new set of definitions for actions to move from one agent to another with different or additional motor capabilities. Whereas Dev-PSchema, in contrast to other models discussed above, uses the abstract format for sensory information and actions, enabling the agent to incorporate new sensory information and actions without any prior definitions.

2.3.2.4 Open-ended Learning

Learning models inspired from developmental psychology should demonstrate an incremental learning. The model should expand its knowledge through new experiences. The learning models, in [172, 6, 136, 140, 35], have demonstrated incremental, hence open-ended, knowledge development through experiences beginning from a basic knowledge set. PSchema system has demonstrated an ability to learn schemas through active experiences starting from a set of bootstrap schemas. The bootstrap schemas represent a reflex-like behaviour in infants, triggered through different stimuli. Bootstrap schemas only contain postconditions, end effect, of the action without any preconditions. Similarly, Dev-ER model [6] demonstrated learning new schemas through experiences using basic schemas. The basic schemas containing context (preconditions) and an action, in contrast to the PSchema system where basic schemas contain action and postconditions. The context contains generalised attributes of objects, which can be replaced with currently observed objects. The context also contains the internal state of pleasure and displeasure for different attributes and actions in the basic schemas. Thus, unlike the PSchema system [172] which contains basic schemas with postconditions only, Dev-ER schemas contain generalised preconditions.

IAC model has also shown the capability for learning developmentally from experiences [136, 84, 134]. However, unlike PSchema [172] and Dev-ER [6], it begins learning with motor actions rather than the basic schemas in the memory. The IAC algorithm lets the agent develop its understanding by performing motor actions and observing their effects. That is why IAC is able to demonstrate development of sensorimotor knowledge with low-level motor commands [84]. Whereas PSchema and Dev-ER models require underlying low-level systems that translate messages between the model and an agent.

Although learning models, including in [141, 83, 92, 137], demonstrate developmental progress in learning, they tend to rely on external reinforcement. This limits the open-ended development of the models.

2.3.2.5 Generalisation

Generalisation helps learning systems to extend knowledge gained from a few experiences to another experience. Generalisation also helps to predict outcomes of different actions in a given environment, through developing general concepts with experiences.

PSchema model [172] is capable of generalising its experiences through inductive reasoning and builds generalised schemas. In inductive reasoning a generalised conclusion is built from a set of specific instances. PSchema uses a combination of two different schemas with identical actions to develop a generalised schema. The generalised schema is used to predict its action effects by instantiating generalised attributes with those from the current perceived environment. The agent may develop different levels of generalised schemas, from single to all generalised attributes in pre/post conditions through experiences. Generalisation in PSchema is further discussed in Chapter 4. On the other hand, Dev-ER [6] and CALM [140] use deductive reasoning to create generalisation. In deductive reasoning, generalisation move from a very generalised concept to a specific instance. Using a basic (generalised) schema the agent build a new generalised schema with generalised attributes

that are irrelevant to the current experience. Another level generalised schema is developed if the agent obtains a similar effect with different objects. Thus developing very generalised schemas to concrete schemas through experiences. This is the opposite of PSchema and Dev-PSchema where development goes from concrete schemas to the very generalised schemas (from un-generalised to partial, and partial to complete generalised schemas). In addition to this, Dev-PSchema is also able to find functional relationship in schema properties.

ICM model has been demonstrated with some level of generalisation, however, this was made possible through tuning the pre-trained network [137]. Similarly, IAC learning model is not capable of generalising its experiences, hence limiting its performance in the particular situations [136, 84, 134]. CLASM model [35], also, has not been demonstrated with generalisation capability. This limits ICM, IAC and CLASM models to use existing knowledge in novel situations to predict action effects.

2.3.2.6 Predictions

The learning models, in [172, 6, 136], are capable of predicting the action effects. PSchema [172] uses schema postconditions for predictions. If a schema predictions matches the currently perceived state, the schema postconditions are considered to be the results of the schema action. The agent can predict the outcome with either concrete schemas having preconditions matching to the perceived state or instantiating pre and postconditions in the generalised schemas. Similarly, Dev-ER system [6] is also able to predict schema action effects through instantiating the schemas with perceived states having similar attributes. In both models, predictions are high-level and without mathematical function. For example, both models are able to predict that an object can be put into centre of the vision if the robot turns its head in the direction of the object e.g., left, (*gaze_x* 0.1, *gaze_y* -3.2).

IAC model is only able to predict in situations which have been seen previously [136, 84, 134]. It is only able to predict expert action effects having the same context as the perceived

state. Similarly, CLASM model [35] is only able to predict action outcome in previously observed contexts. Hence, they are unable to predict the action effects for any situations which previously have not been experienced, whereas PSchema, CALM and Dev-ER models are able to predict action effect in novel situations using generalised schemas [172, 140, 6].

2.3.2.7 Complex Actions

Infants develop complex skills through experiences and combining different low-level actions (see Chapter 1 for details). Learning models inspired from infants' learning are expected to demonstrate such developments. Although, PSchema [172] is capable of planning a sequence of actions from a given state to a distant state as a chain. The planning is only performed when an external agent sets a target state. Thus the chaining capability in PSchema cannot be considered as development of complex actions that are re-used in future experiences. Dev-ER [6] and IAC [136] are unable to develop complex actions.

The learning models, PSchema, IAC, Dev-ER, CALM and CLASM are unable to develop complex action through exploratory play, [172, 136, 6, 140, 35]. This limits these learning models to develop complex actions hence complex skills developed from basic actions. Thus, for solving the tasks that require more than one action, the learning models will go through re-planning and executing each action independently. Dev-PSchema is able to develop high-level actions through schema chains. This capability in Dev-PSchema demonstrates more faithful modelling of human behaviours, in which an action sequence is considered as a single high-level action after few successful repetitions [160, 199, 118, 79].

In conclusion, the learning models discussed above have been identified with limitations in abstraction [172, 6, 136, 140, 137, 35], open-ended learning [141, 83, 92, 137], generalisation [136, 35, 137] and complex actions [172, 6, 136, 140, 35]. However, from these learning models and learning characteristics in early infancy discussed in Chapter 1 a learning model

inspired from developmental psychology can be inferred that should include the following characteristics:

1. **Primitive actions**

   The model should be provided with an initial set of behaviours for interaction with the environment and learning [109]. This is similar to having reflexive behaviours in newborns. Thus the system will begin with some basic behaviours on which to develop. Such behaviours may be learnt separately or mature with learning.

2. **Developmental Learning**

   The model should demonstrate open-ended learning like humans do [31]. The model should be able to continue learning, even beyond the achievement of tasks specified. This capability will enable the system to learn from experiences, wherever possible.

3. **Independent Learning**

   The model should be able to learn independently. Learning may begin with exploratory play behaviour, as seen in infants [147]. According to Piaget initial knowledge in infants begins with their own experiences. Thus a learning model reflecting infant development should be able to learn independently from their own actions and behaviours. It may also possess the capability to learn from imitation and other social learning, however exploratory behaviour is a key component [108, 106, 117, 18].

4. **Abstraction**

   The model should be able to accept and associate new sensory information with its behaviours. This ability will enable the system to accept the new information with no previous experiences and definitions. This capability will also enable the system to perform in different environments and context giving different sensory information, without making any changes to the model itself.

5. **Intrinsic motivations**

   The model should contain a mechanism for attention and action selection through intrinsic motivation. This mechanism will be responsible for triggering behaviours. As discussed in Chapter 1, exploratory play in infants is triggered through directing their attention towards novelty, ambiguity or change in the environment. Thus the model should be able to direct attention towards novel objects/situations and also demonstrate habituation as observed in infants [177, 14, 86].

6. **Generalisation**

   The model should be able to learn from a small number of experiences and use those experiences to extend its knowledge. The model should also be able to adapt if existing knowledge does not correlate with the experienced outcome. Infants have been found to generalise their experiences very quickly and associate certain behaviours with certain objects or environments [201, 67, 86]. Failing to re-acquire such associations should cause the system to re-evaluate the generalisation.

7. **Predictions**

   The model should have the ability to predict the outcome of its behaviours. Visual anticipation has been observed at a very early age in infants [167, 2, 196]. They seem to predict the results of their or other agents behaviours before execution. This capability will enable the agent to predict the outcome of actions before actions are actually performed.

8. **Complex Behaviours**

   The model should be able to learn complex behaviours from primitive or reflexive behaviours. Although learning complex behaviours is not seen in early infancy, infants aged 14-16 months have been observed to learn and perform complex behaviours containing more than one action [29]. Most studies on infants learning complex actions

are based on learning through imitation but the results show that infants are able to form complex actions in other context too. The complex actions may represent means to achieve a distant goal which is not possible to achieve directly with a single action. The learning system modelled on infants' behaviours should be able to develop complex actions through combining common sequences of actions together. Such higher-level complex actions through repetitions can be considered as a "single" action. These complex actions help to develop distant (objective) states in their environment that are not possible to achieve through one action. The distant objective state may be defined by the model itself through its internal mechanism or an external agent.

9. **Different Behaviours**

   The model should be able to show different behaviours. In developmental psychology experimental results are shown as an average of behaviours, however each individual shows difference in the behaviours [13, 77, 57]. The model should therefore be able to demonstrate different behaviours in a similar environment, as infants do. This capability will enable the model to simulate different individuals with different behaviours.

The learning systems discussed above, modelled on developmental psychology, have shown significant results. However, they lack some of the characteristics listed above. Some models need large data sets to train [166, 83], hence do not show independent learning, lack in the capability to plan actions [6], do not contain exploratory behaviour or independent learning [141, 83, 70, 92] and are not able generalise their experiences [141, 6, 92, 96], hence limiting the performance within novel environments. Moreover, all the models discussed above tend to model an average individual hence will demonstrate a generic behaviour in a given environment. In Table 2.1 we provide a comparison of different learning models, including Dev-PSchema developed as part of this work, based on the points concluded above. It should be noted that point 1 has been excluded as any artificial learning is always provided

with initial actions or behaviours to perform in an environment.

In Chapter 3 we introduce the learning model developed here, Dev-PSchema.

Model	Open-ended learning	Independent learning	Abstraction	Intrinsically motivated	Adaptation & Generalisation			Prediction	Developing complex actions	Different behaviours
					P	C	F			
Dev-ER[6]	✓	✓	×	✓	✓	✓	×	✓	×	×
IAC [136]	–	×	×	✓	×	×	×	✓	×	×
BNs [125]	-	✓	×	×	×	×	×	✓	×	×
OACs [92]	–	–	×	–	×	×	×	✓	✓	×
PDDL [141]	–	×	×	×	✓	✓	×	✓	×	×
CLASM [35]	✓	×	×	–	×	×	×	✓	×	×
CALM [140]	✓	✓	×	✓	✓	✓	×	✓	×	×
ICM [137]	×	×	×	✓	–	✓	×	✓	✓	×
MDP Schema [83]	–	–	✓	×	–	✓	×	✓	×	×
PSchema [172]	–	✓	×	✓	✓	✓	×	✓	×	×
Dev-PSchema	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2.1 Comparison of learning systems with capabilities.

✓ = present, × = not present,

– = partially present/not applicable/not enough information available/depending upon the application or implementation

P = Partial generalisation, C = Complete generalisation F = Functional generalisation

Chapter 3

Play Generator: Dev-PSchema

The aim of this work is to demonstrate an independent and intrinsically motived learning system that is able to learn from sensorimotor experiences using exploratory play behaviour. This system will also demonstrate the capability of performing sequential actions as behaviour planning, generalising experiences and applying knowledge to novel situations. Furthermore the system is able to demonstrate different behaviours in a similar environment by tuning different parameters of the system.

This work introduces a play generator, Dev-PSchema that drives exploration, when interfaced with the agent, for open-ended learning through sensorimotor experiences. This system is modelled on the sensorimotor stage of Piaget's cognitive theory. At this stage, as discussed in Section 1.4.1, infants learn from sensorimotor experiences and learning is ego-centric. Dev-PSchema, also, demonstrates the egocentric learning through active sensorimotor experiences. According to Piaget's theory learning is recorded in the shape of schemas, connecting behaviours and sensory information. Similarly, Dev-PSchema records learning in the shape of schemas, containing preconditions (sensory information before action), action and the postconditions (sensory information after action). Thus Dev-PSchema incorporates the schema like learning mechanism building through experiences. In this Chapter, we discuss the working mechanism and basic technical components of the systems where mechanisms have been inspired by those seen in developmental psychology. The

contributions of the thesis are discussed in Chapters 4, 5, 6 and 7 along with the technical mechanisms taking inspirations from mechanisms observed in infants by psychologists.

Dev-PSchema needs to be interfaced with an agent to demonstrate learning through embodiment and behaviours. Here we demonstrate interfacing it with an iCub humanoid robotic platform and a *Sandbox* simulator. We extend Sheldon's work [172], where a demonstration for effective learning and play behaviour is shown to be applied in two different robotic platforms, an Adept robotic arm with egocentric camera and iCub, with the help of PSchema tool. We have enhanced the PSchema tool by significantly modifying and adding new underlying mechanisms. The new version defined here is named as Dev-PSchema (Developed PSchema). It makes use of high-level sensory information and actions, thus requires an intermediary system to translate and transfers messages to an agent's sensors and motors.

The intermediary system is responsible for preparing high-level perceptions from raw sensory information. It also translates high-level actions into motor commands and transfers them to the agent to execute. For simplicity, we refer to this intermediary system between Dev-PSchema and an agent as the sensorimotor control (SMC). The SMC used for the iCub is divided into several subsystems including reflexive motor command control, vision and tactile sensory information processing system. The SMC is used as a tool here so not described in depth, for more details on the system please see [62, 107, 102, 101, 103]. The *Sandbox* simulator has its own intermediary controller, similar to the SMC for the iCub, for motor commands and processing sensory information. Figure 3.1 shows a high-level connection diagram between the internal systems of the agent and Dev-PSchema.

The SMC prepares perceived sensory information for Dev-PSchema as symbolic representation and transfers high-level actions into low level motor command and joint positions. The interface controller, as shown in Figure 3.1, manages connections between Dev-PSchema and the SMC. Dev-PSchema is able to process different sensory information as received through the SMC. However, spatial parameter labels, e.g., x or y, gaze_x or gaze_y etc., are

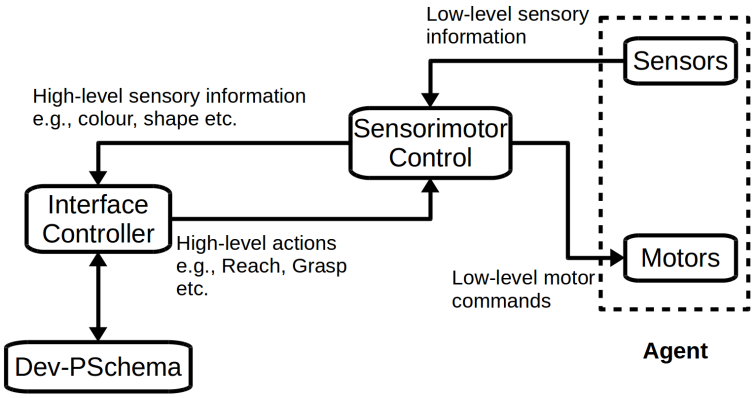


Fig. 3.1 An illustration of the interface between Dev-PSchema and the sensors and motors of an agent.

pre-set. This enables the system to separate spatial properties from the others, in a given sensory information. In Dev-PSchema spatial properties (location) are referred to as *Coordinates*.

Dev-PSchema works in three different modes; Bootstrap, Play and Problem solving. Montesano et al. [125] describe three developmental stages of learning for robots; sensorimotor coordination, world interactions and step towards social learning. In sensorimotor coordination the robot learns basic motor skills and visual perceptions. In Dev-PSchema, part of this stage is modelled as the bootstrapping mode. In the bootstrapping mode Dev-PSchema builds a model of basic actions. Visual perceptions and eye-motor coordination are learnt through a separate process by SMC, described in [62, 107, 102, 101, 103, 34]. A detailed architecture of the low-level system is shown in Appendix A. For simplicity, we consider visual perception and motor coordination as learnt capabilities that are available for use by Dev-PSchema.

The play mode represents the world interactions stage in developmental learning stages for robotics [125]. In the world interactions stage, defined by Montesano et al. [125], the robot is expected to learn about the changes in perception as a result of actions, build object affordances, and develop prediction and planning skills. In play mode the agent interacts with the environment and learns the associations between sensory information and performed actions. Furthermore, Dev-PSchema is capable of planning action sequences to achieve

higher-level target states which are not possible with a single action. This work here focuses on independent learning hence the imitation part of the developmental learning stage is not considered. In the following sections we outline all the underlying mechanisms of Dev-PSchema and, where appropriate, how they changed from those in PSchema, starting with the modes of operation.

3.1 Modes of Operation

Dev-PSchema can be operated in three different modes, activated through the interface controller shown in Figure 3.1. The modes of operations are modelled on infants' capabilities at the sensorimotor stage of cognitive theory.

3.1.1 Bootstrapping

Bootstrapping is used to build bootstrap schemas, a set of basic schema with primitive actions to interact in the environment. Bootstrap schemas are built from motor babbling, in the absence of object interactions, and getting feedback from the environment[1]. During the bootstrap process the agent performs all the available actions and observes the feedback to create bootstrap schemas. During these behaviours the agent does not interact with objects in the environment. Figure 3.2 shows a flowchart of bootstrap process in the system.

The bootstrap process is modelled on the motor babbling behaviour observed in newborns and infants at an early age. Motor babbling is seen as either reflexive or exploratory behaviours triggered by distant stimuli [37]. The bootstrap process begins with the creation of an empty schema from a schema template and the specified action is added into it. After creating an action schema, the action is executed in the environment. The executed schema is updated with the observed changes in the environment as postconditions. The agent will be able to use the bootstrap schema to interact with the environment and

[1] A set perceptions of the environment at any time is called *World State* (WS).

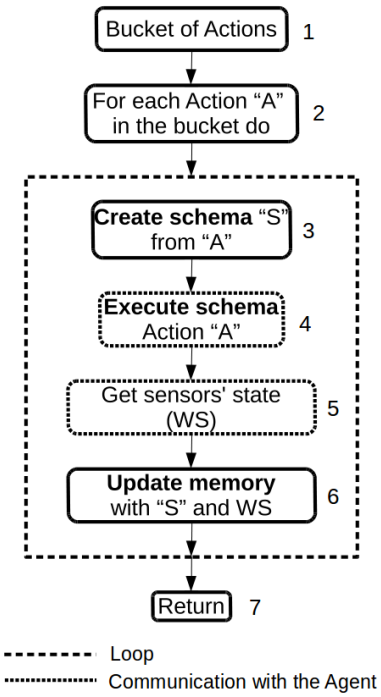


Fig. 3.2 Flow diagram for bootstrapping mode.

build up its knowledge to the next level, from basic motor skills towards object affordance. Figure 3.3 shows an example of a bootstrap schema generated after performing a grasp action.

Preconditions	Action	Postconditions
	Grasp	<Observation=proprioceptive, hand=3.0, grip=100, x=2.0, y=3.0> <Observation = touch, average = 50>

Fig. 3.3 An example of a Bootstrap schema.

The bootstrap schema, in Figure 3.3, contains action and postconditions only, as the effect were observed when a random action was performed from the bucket of actions in the absence of objects in the environment. The bootstrap schema is considered to be the result of the motor action performed at a random location and observing the proprioceptive feedback e.g., hand position, grip etc.

Algorithm 1 shows the procedure of creating a schema from a given action, used in bootstrapping process.

Algorithm 1 Creating a schema from an action

```
1: procedure create_schema_from_action (Action A)
2:     for Each schema S in memory do
3:         if Schema action is same as A then
4:             return S
5:         end if
6:     end for
7:     S = new_empty_schema
8:     Set A as Schema S action
9:     Return S
10: end procedure
```

Algorithm 1 represents a mechanism for creating a schema with an action only, without any sensory information. The mechanism initially searches its long-term memory[2] if any of the existing schemas contains the same action. If it is unable to find any schema containing the given action, it creates an empty schema template and adds the provided action as the schema action. This mechanism is used in several different algorithms, discussed later in this Chapter. This mechanism is similar to the one used in PSchema, the key difference between the mechanism used for representation of actions and perceptions is abstract representations. Dev-PSchema uses the abstract representations whereas the PSchema utilises pre-defined templates. The difference between the two representations is discussed further in Section 3.2. The newly created schema is added into the long-term memory, which is responsible for recording all the schemas and chains including their statistics. The short-term memory of Dev-PSchema keeps a record of the currently perceived world state and it is updated after every action performed.

3.1.2 Play Mode

Play mode starts with the agent interacting with object through actions in bootstrap schemas. The agent uses changes caused by the schema action to create higher level schemas from

[2]The term memory is used to refer to long-term memory here, unless specified.

the bootstrap schemas, wherever possible. The process of creating the next level schemas by obtaining changes in the environment using existing schemas corresponds to the accommodation and assimilation in cognitive theory of development. Whereas, this mode of operation is similar to the active time in infants where they interact with their environment using basic motor skills, hence named as *Play* mode. The excitation calculator, an internal mechanism responsible for calculating excitation of each action in the long term memory, executes the most excited schema. The newly created schemas also participate in schema selection mechanism through the excitation calculator as they are included in the memory. The excitation calculator is further discussed in Chapter 5. Figure 3.4 shows a flowchart of operation in the play mode.

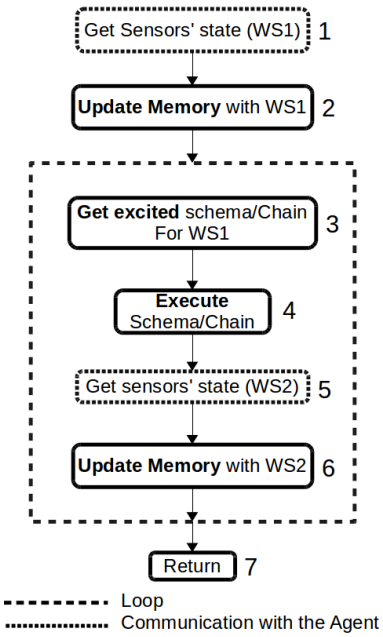


Fig. 3.4 Flow diagram for the Play mode.

During the play, the system works in a loop between performing an action or a sequence of actions and getting the feedback from the environment. The agent selects an excited schema or sequence of schema actions, a chain, through the excitation mechanism to execute in the environment. The schema execution is followed by updating the short-term memory. During the update, new schemas are created wherever possible. The new schema may be

created by adding new information into an existing schema (assimilation) or by creating a completely new schema that is different to the existing one (accommodation). The agent builds up the knowledge by learning new schemas using and re-using existing schemas in different situations. Thus demonstrating a hierarchical development of knowledge, as seen in infants [147]. Figure 3.5 demonstrates how a new schema is created from a basic bootstrap schema.

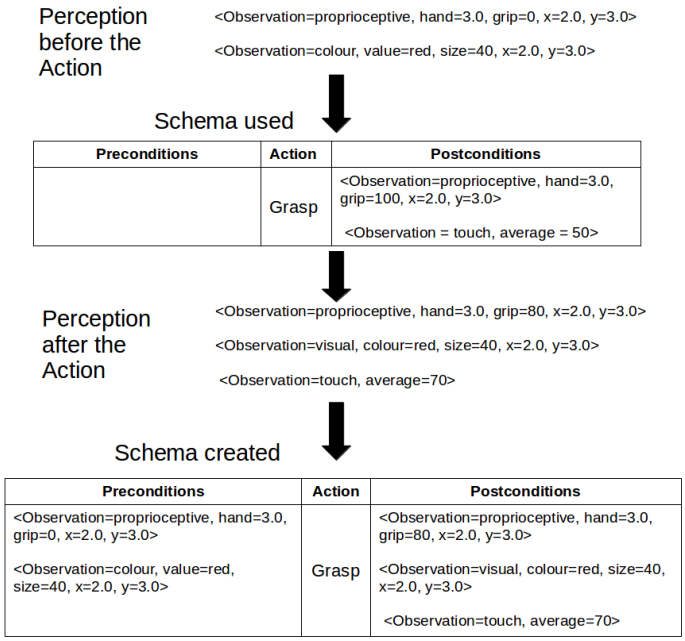


Fig. 3.5 An example of creating a higher level “Grasp” schema from a bootstrap schema, sensory states before and after the action.

Figure 3.5 shows that the agent receives the bootstrap grasp schema as the most excited schema for a given sensory state. The execution of the grasp action causes a change in the sensory state. As the postconditions of the bootstrap schema did not include the new change i.e., touch observation and grip=80 in proprioceptive observation, hence the outcome is unexpected. The unexpected outcome caused the system to create a new schema with the changes observed and add it to the memory. The new schema is compared to the existing schemas in the memory during this process to avoid duplicate schemas and to identify opportunities for

generalisation (See Chapter 4 for more details on generalisation).

3.1.3 Problem Solving

In problem solving mode, Dev-PSchema solves a user defined problem by making use of what it has learnt while playing. In this mode, the user enters a problem as a sensory state to be achieved. Dev-PSchema finds all the possible solutions to achieve the target state starting from the current state. Figure 3.6 shows a flow chart of the problem solving mode in Dev-PSchema.

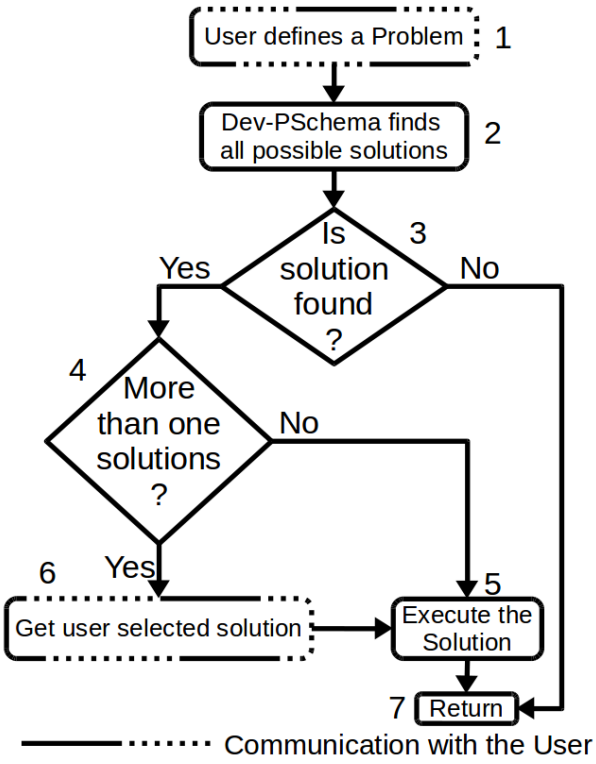


Fig. 3.6 Flow chart of problem solving mode in Dev-PSchema.

This mode is part of modelling the capability of developing high-level actions through action sequencing observed in infants at the fourth stage of cognitive developmental theory. Infants at this stage have been observed planning in order to solve problems that requires

more than one action. This capability enables Dev-PSchema to achieve an objective state, that require more than one action, through the process of action sequencing (see Chapter 6 and 7 for action sequencing in Dev-PSchema).

3.2 Sensory State - Observations and World State

Dev-PSchema is a high-level learning system, dealing with high-level sensory information and actions. Thus sensory information is prepared for Dev-PSchema by a lower level system. Sensory information from the agent's sensors is prepared for Dev-PSchema in the form of perceptions containing the sensor's type and the underlying properties. There are two categories of sensors used in this work, proprioceptive and physical. Proprioceptive sensors provide sensory information about the agent itself, such as the position of the manipulator or hand and the size of the grip. Physical sensors provide sensory information about the state of the environment, from the agent's perspective, including the visual perceptions. Visual sensory information may be preprocessed into several features, such as colour, shape etc. In Dev-PSchema each feature type is treated as individual perception type so a separate observation is created for each. All the sensory information is recorded as a set of observations in Dev-PSchema. A set of observations at a particular instance are referred to as a world state (WS) in the system.

As an example, one of the platforms used in the experiments is the iCub humanoid robot. The robot is equipped with two cameras as eyes. Using a resolution of 320×240 the camera images are processed to detect colour patches, bright patches, edges and motion [171]. The Sensorimotor Control (SMC) system developed by Shaw et al. [170] as part of IM-CLeVeR project is used to provide interface layer between the iCub and Dev-PSchema. A series of sensorimotor mappings are learnt developmentally through a process of motor babbling [103], where maps consisting of overlapping fields are used to represent sensory or

motor spaces [102, 47].

Alongside the iCub, a *Sandbox* simulator, is also used in this work, consisting of an artificial visual system, a hand and objects in a grid based environment. The visual system consists of two different visual perceptions, colour and shape. Each of the perceptions contains size, spatial position (coordinates) in the environment and high-level values of the sensory properties. Several different coloured objects, e.g., red, blue, yellow etc., and shapes e.g., sphere, cylinder, cube etc., are used in the experiments.

Figure 3.7 shows the perceptions prepared by the SMC for Dev-PSchema from the raw sensory information perceived through the iCub's sensors. This sensory information is consist of proprioceptive perceptions (type 1) and observed visual features (type 0). The interface controller prepares the received message to construct the current world state containing a set of observations.

```
perception type 1 gaze_x -15.41 gaze_y -55.87 hand 4 grip 25.66
perception type 0 gaze_x -25.62 gaze_y -60.42 sensor_name colour feature_id 4.0 sensor_name
brightness feature_id 29.0 sensor_name edges feature_id 13.0
```

Fig. 3.7 An example of perception sent by SMC to Dev-PSchema, perceived through iCub's sensors.

In Figure 3.8 it can be seen that each item of sensory information is represented as observations with its own coordinates. The gaze coordinates represent the visual egocentric space of the iCub as used in the SMC, and are formed from the combination of eye and neck motor configurations to obtain gaze direction. The depth is handled by the SMC when performing an action, so the gaze space can be considered as 2D (2 dimensional) space in spherical shape centred on the head. This group of observations is called a world state in Dev-PSchema. A partial or full world state may be included into pre or post conditions of a schema depending upon the changes in observations following the performed action.

<observation=proprioceptive, hand=3, grip=25.66, gaze_x=-15.41, gaze_y=-55.87>
<observation=colour, feature_id=4.0, gaze_x=-25.62, gaze_y=-60.42>
<observation=brightness, feature_id=29.0, gaze_x=-25.62, gaze_y=-60.42>
<observation=edges, feature_id=13.0, gaze_x=-25.62, gaze_y=-60.42>

Fig. 3.8 A world state (WS) prepared from SMC perception message.

Figure 3.9 shows an instance of a world state through graphical representation with different types of observations.

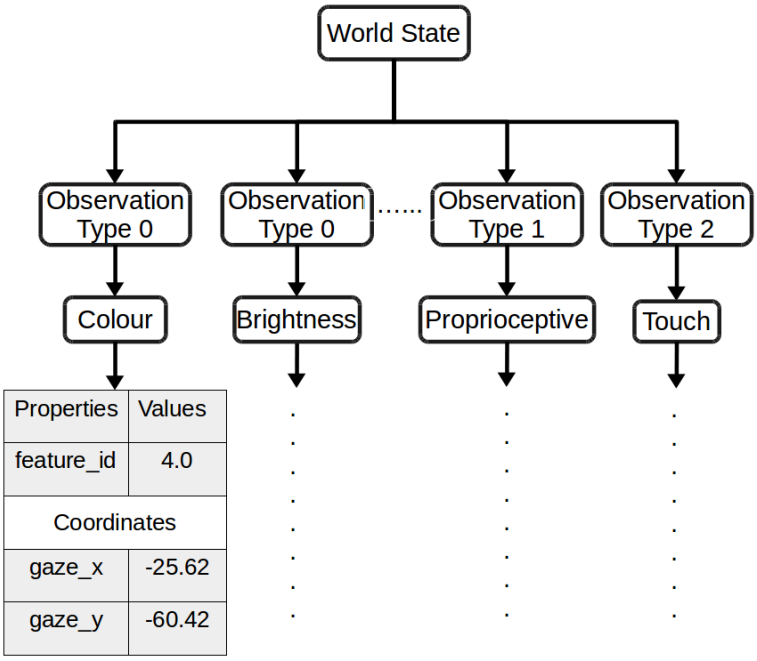


Fig. 3.9 An example of world state (WS) structure with its observations.

In PSchema system [172] all the sensory information and resulting observations were predefined including their properties, as schema templates. If an unexpected sensory information, having no predefined template, was received, it could not be processed. In Dev-PSchema, the template schemas are replaced by abstract schemas with the abstract observations, which enables it to add new sensory information, without predefinition. Thus novel sensory information can be incorporated into the system without any modification. Furthermore, the abstraction in Dev-PSchema enables it to interface with different agents having different sensors without any changes in the system, as demonstrated in this thesis by

interfacing with a simple *Sandbox* environment and a real robot.

3.3 Actions

The end effector or hand of the agent can perform several different actions in the environment resulting in different perceptions. Both the actions and sensory perceptions are specified in a higher level representation in order to maintain the focus on playful interaction rather than the low level sensorimotor control. Actions used in this work are defined as follows:

- Reach (*hand*$*$, *gaze_x*, *gaze_y*)

  Reach action is used to reach the specific position in the environment. Argument "hand" defines the hand of the iCub. We use integer 3 for left hand and 4 for the right hand. Arguments *gaze_x* and *gaze_y* define position in the visual space. In *Sandbox* only one hand is present, hence this argument is not available in sandbox simulator.

- Grasp (*hand*)

  A high-level action that can be used to grasp a 'graspable' object in the environment. SMC/sandbox provides "Grip" feedback with proprioception, indicating between fully open (0.0) and fully close (100.0) grip, depending upon the object grasped. A "Touch" sensor is also added in the perception which provides an average intensity of touch. In iCub's case average touch value is provided by active touch sensors on the fingers and palm of the iCub. In the sandbox the touch average is provided between 50$\sim$90, depending upon the object, when an object is grasped and 100 for grip and touch with no object in the grip.

- Release (*hand*)

  A high-level action that can be used to open the "Hand". The "Grip" value in the

proprioception provides feedback, indicating fully open (0.0) when this action is performed.

- Push (*hand*, *direction*)

  A push action displaces the object present at the hand position. With this action "Hand" performs to-and-fro movement, causing displacement in an object's position only if present at the Hand position whereas "Hand" remains at the same position. Thus the action causes small lateral movement in the specified direction then returns to the starting position. In the iCub, the movement starts from its torso, whereas Sandbox does not show any visual movement, however the action affects the object's position if it lays at the same position as hand. The "direction" defines the direction of the push. Direction 0 sets the push towards left side and 1 towards the right side.

- Press (*hand*)

  This action may result with different high-level observations in the environment i.e., touch, sound, light etc., depending upon the application and the experiment. This action is used with *Sandbox* simulator.

- Squeeze (*hand*)

  A squeeze action applies force on the object at the hand position causing an effect similar to the "Press" action. This action is used in an experiment to demonstrate variation in action selection having similar effects.

- Fixate (*gaze_x*, *gaze_y*)

  Fixate action is used to shift the gaze to the specified position in the environment. Arguments *gaze_x* and *gaze_y* define a position in the gaze space. The action is demonstrated with *Sandbox* environment only. The fixated position represents the fovea, focal point of gaze inspired by human vision. However, all features currently visible on

the retina are returned for inclusion in the current world state. In Sandbox, when this action is used a new perception, fixate, is added into the sensory information indicating fixated position.

At this point we assume that joint movements for each of the actions described above have been learnt by the low-level systems, however their effects on objects in the environment are learnt by Dev-PSchema in bootstrap mode then extended during play mode. Dev-PSchema sends commands for an action to the agent along with its arguments. The low-level system (SMC) translates the action into low level motor commands for the agent. Dev-PSchema, with the abstraction system, is able to add new actions on the fly. As the low-level system discovers new actions, for example a push from a failed reach, then they can be added into Dev-PSchema with a new label. This capability enables Dev-PSchema to interface with different agents with different motor capabilities and actions, without any modification in the system. As with observations, in PSchema [172] all the actions were predefined and new actions could not be added on the fly. Although this capability is available in Dev-PSchema, it has not been demonstrated in the experiments in this thesis. However the interface controller can be programmed to dynamically incorporate any new action discovered by the low-level system.

Algorithm 2 describes the mechanism for execution of a schema action. Once the action is executed, the system updates the count of activations of the schema and its action. The schema and action activations are later used to calculate schema statistics for excitation. The excitation system is discussed in details in Chapter 5.

Algorithm 2 Schema execution algorithm

```
1: procedure execute(Schema S)
2:     Action = S_action
3:     execute(Action)
4:     Increase Schema S activations by 1
5:     Return
6: end procedure
```

3.3.1 Schema Generation

A schema represents the sensorimotor experience, containing action and sensory information from before and after an action. In the original PSchema system, all the observations from the world state were considered as part of the schema. As some of these may not be relevant to the action. Dev-PSchema aims to include only the observations directly relevant to the action. To find the relevant observations, a key detail used is the proprioceptive (hand/manipulator) coordinates. Initially the observations which contain relevant coordinates before and after the action are considered. This mechanism can be seen as learning affordances in the environment using active experiences. Where an observation changes at a distance, this can be considered if applicable.

3.3.2 Associated Observations

It is not always obvious if an action is the direct trigger for a change in an observation, for example an observation changing at a position different to where the action was performed. Such observations are included as *Associated pre/post conditions*. This mechanism resembles the development of tool-use capabilities, where an action with an object causes changes, e.g. pressing a switch to turn on lights or releasing another object into the workspace. Figure 3.10 shows a schema created following an action, generating associated postconditions.

Figure 3.10 shows that Dev-PSchema incorporated new sensory information acquired following an action as associated postconditions. This enables the agent to learn some associations with actions and objects, and develop tool-use like behaviours. As with postconditions, some observations may be added as preconditions, depending upon the observed changes in the environment after an action. For example if an action on an object causes another object, at a different position to disappear this will lead the system to add the disappeared object related observations as associated schema preconditions.

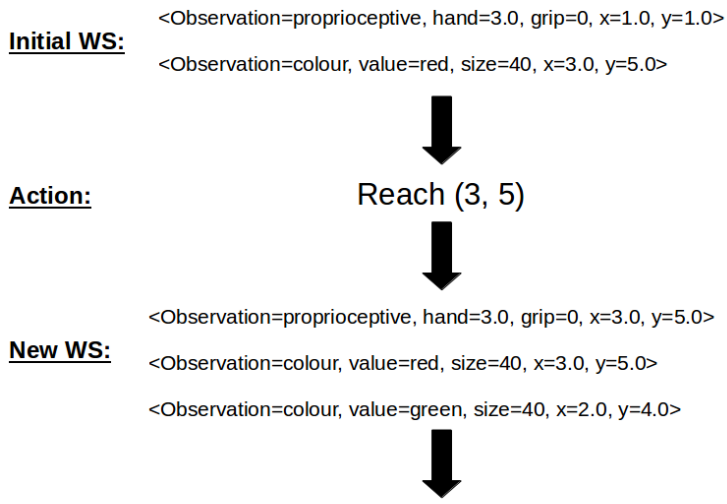


Fig. 3.10 An example of association in a Schema.

Although the capability to link associated observations helps to add most relevant information in an schema, this can cause the agent to include false associations in a noisy environment. In a noisy environment, the schema may include associated pre/postconditions that have no direct association with the schema. To overcome this problem, Dev-PSchema relies on repetitions of the associations. If the associated pre/post conditions are repeated (at least once) with the same action, such associated observations are transferred to concrete pre/post conditions. This will help the agent develop complex associations with repetitive experiences.

3.3.3 Schema Statistics

Dev-PSchema maintains statistics on executions of each schema e.g. activations and successes, for excitation calculations. If the perception(s) in the environment following a schema execution match the schema's postconditions, the execution is considered to be successful.

The number of schema executions and successes are further used to calculate schema success rate. Equation 3.1 gives the calculation for a schema success rate.

$$schema\_success\_rate\ (S_r) = \frac{schema\_successes}{schema\_activations} \tag{3.1}$$

Equation 3.1 will return the value of 1.0 if a schema is found to be successful every time it was executed in the environment.

3.4 High Level Actions - Schema Chains

As an agent gains more experiences and skills, some of the skills can be linked together in order to form higher level skills in a hierarchical structure. For example, individual actions such as reach and grasp can become linked by a single reach→grasp action. Through playful exploration, more complex chains can be learnt that combine basic schemas and form more sophisticated high-level schemas, hence actions.

Chains are seen as sequences of schemas, which the agent discovers by finding the links between preconditions and postconditions of schemas in memory. Chaining helps in achieving states of the environment that are not possible when employing a single schema. For example picking up an object from a reachable position needs two different actions to be achieved; i) reach to object location and ii) grab object. Figure 3.11 shows an example of a two schema chain obtained by linking preconditions and postcondition of two different schemas.

Longer chains are discouraged during the chaining process in order to reduce computational costs and avoid overly complicated chains that are more likely to be unsuccessful.

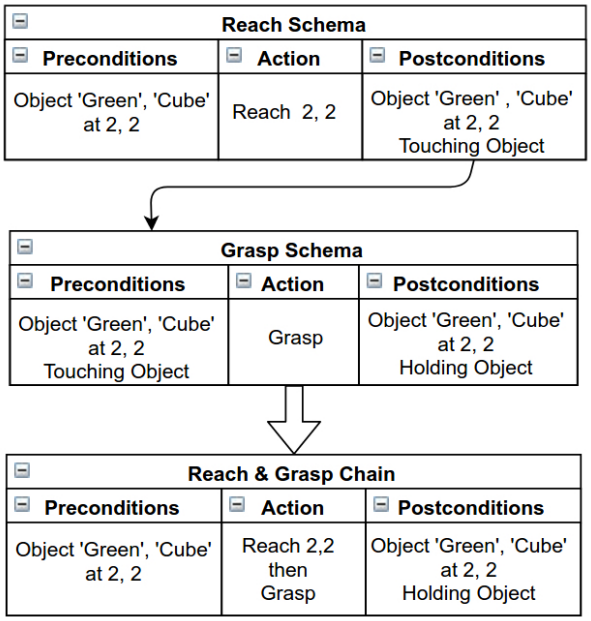


Fig. 3.11 An example of chaining two schemas to create a "2-Schema chain".

Here, a limit of 5 schemas is set. The chaining mechanism is discussed in detail in Chapter 6.

3.5 Update Memory State

Dev-PSchema uses two types of memories; long-term and short-term. The long-term memory consists of learning recorded in the shape of schemas. Short-term memory is updated frequently, in particular before and after the execution of an action. Short-term memory keeps track of recent sensory states, before and after perception of an action. The excitation calculator finds a suitable schema/chain, from all the schemas and chains in the long-term memory, for execution relevant to the state present in the short-term memory. The executed schema/chain predicts the outcome at the end of the execution through the relevant postconditions. A mismatch in the postconditions and the obtained state leads the system to develop new representations i.e., schemas. Thus, a part of the memory update process resembles the *assimilation* and *accommodation* processes of the schema mechanism for cognitive developmental theory. Figure 3.12 shows a flow chart of the underlying process for

updating the memory.

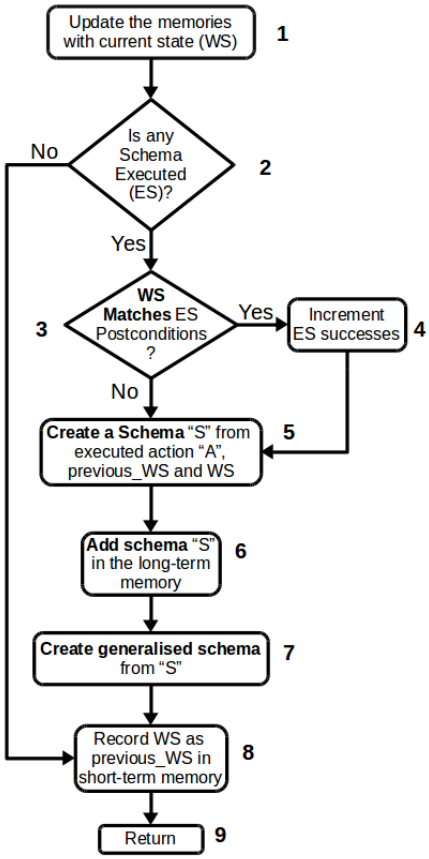


Fig. 3.12 Updating the memory before/after an action. Highlighted processes will be explained in algorithms.

As can be seen in Figure 3.12 the system updates its memory in two ways (block number 2). Prior to any schema being executed, the perceptions are just recorded in the short-term memory as the previous memory (block 8). Following a schema execution (ES), the memory update will follow further processes to identify and perform any necessary updates in long-term and short-term memories (block 3-8). We discuss each of the underlying processes and algorithms in Sections 3.5.1 to 3.5.4. Also, during the updating process the system records statistics for all the perceived perceptions (observations). Observation statistics are used by the excitation calculator, described in Chapter 5.

3.5.1 Matching World States

Memory updates following an action trigger several different processes. The first stage is to confirm the predictions made by the executed schema's postconditions (block 3). Predictions are confirmed if the new world state (WS) matches the postconditions of the executed schema. Algorithm 3 describes the process of matching two sensory states, the current state and the executed schema postconditions in this case (shown as block 3 in Figure 3.12).

Algorithm 3 Matching World States (WS)

```
1: procedure states_match (WorldState WS1, WorldState WS2)
2:     for each observation O1 in WS1 do
3:         set variable Found False
4:         for observation O2 of type O1 in WS2 do
5:             if each property P1 in O1 is same as P2 in O2 then
6:                 set variable Found True
7:                 break; Go to next O1
8:             end if
9:         end for
10:        if variable Found is False then
11:            Return False
12:        end if
13:    end for
14:    Return True
15: end procedure
```

When matching two states, Algorithm 3, initially matches by perception type in both world states (current state world state and postconditions). For example colour perceptions will only be compared with other colour perceptions. If the world states do not have the same perception type, the state are considered not to match. If the same perception type exists on both sides, the value of each property is compared. The algorithm will also confirm the match if WS_1 is the subset of WS_2. This is used especially during the memory update process where postconditions of the executed schema are compared against the current world state where current world state may contain more information (observations) than the executed schema postconditions. A match confirms successful execution of the schema, increasing its total number of successful executions (block 4 in Figure 3.12).

This algorithm is also used by the process of creating schemas and generalisation. A mismatch in current world state and the postconditions leads the system to consider the

algorithm for creating a new schema, resembling to the *accommodation* in cognitive theory. A match will lead to creating a *temporary* schema to compare with existing schemas. As the algorithm matches the two states even if WS_1 is a sub-set of WS_2, the *temporary* schema may contain additional information as compared to the existing schema postconditions. This leads the system to go through the *assimilation* process to incorporate additional information through the temporary schema. The *temporary* schema is added into the long-term memory if none of the existing schemas exactly matches (matching number and types of perceptions).

3.5.2 Schema Creation Algorithm

In the memory update process, the schema creation mechanism is seen as block 3 in Figure 3.12. This algorithm models both *accommodation* and *assimilation* processes, depending upon the changes in the environment after the executed action. Algorithm 4 describes the process of creating a new schema from an executed action and the sensory states before and after the action.

Algorithm 4 Creating a Schema with actions & world states

```
1:  procedure create_schema (WolrdState previous_WS, WorldState current_WS, Action A)
2:      if states_match (current_WS, previous_WS) then
3:          Return                          ▷ Action did not change anything in the environment
4:      end if
5:      S = create_schema_from_action (A)
6:      if Action A has coordinates then
7:          current_coords = get coordinates from A
8:      else
9:          current_coords = get hand coordinates from current_WS
10:     end if
11:     hand_coords = get hand coordinates from previous_WS
12:     S_preconditions = all observations from previous_WS having hand_coords OR current_coords
13:     S_postconditions = all observations from current_WS having current_coords
14:     S_associated_preconditions = complement_WS (current_WS, previous_WS)      ▷ see Algo. 5
15:     for each observation O in previous_WS with no coordinates do
16:         add_observation_in_state( S_associated_preconditions, O)               ▷ see Algo. 6
17:     end for
18:     S_associated_postconditions = complement_WS (previous_WS, current_WS)     ▷ see Algo. 5
19:     for each observation O in current_WS with no coordinates do
20:         add_observation_in_state( S_associated_postconditions, O)
21:     end for
22: end procedure
```

The schema creation mechanism is referred to in block 5 of Figure 3.12. Algorithm 4 returns without creating any new schema if the executed action did not cause any change in the environment. If any changes have been observed following the action, the system proceeds further beginning with creating a new schema with the executed action using Algorithm 1.

In schema creation mechanism PSchema system adds all the observations in a given world state to a new schema's preconditions and postconditions. Unlike PSchema, Dev-PSchema aims to only add directly relevant observations into a new schema. The relevant observations from the current world state are selected by finding all the observations that have the same coordinates as the executed action, if the action contains coordinates. If the executed action does not contain any coordinates then the hand/manipulator coordinates are chosen for comparing the current world state observations' coordinates. Relevant observations for preconditions from the previous world state are selected by finding all the observations that have the same coordinates as either the hand/manipulator coordinates prior to the action or the executed action coordinates.

Furthermore, any observations in the previous and current world states without coordinates are added into associated preconditions or postconditions respectively e.g., touch, sound etc. Any remaining observations that have different coordinates from the directly considered coordinates that changed after the action are also added into the relevant associated conditions. Algorithm 5 describes the mechanism for finding any observations in a given world state (WS_2) not present in the other (WS_1).

Algorithm 5 compares each property value in an observation of a given world state with the second. Any mismatch of a property or a totally different observation will lead the algorithm to add it in a list of observations. The list containing all the mismatched observations is returned at the end.

Algorithm 5 Complement World State

```
1: procedure complement_WS (WolrdState WS1, WorldState WS2)
2:     complement = Empty WorldState
3:     for each observation O2 in WS2 do
4:         match = False
5:         for each observation O1 of type O2 in WS1 do
6:             if each property P2 of O2 is same as P1 in O1 then
7:                 match = True
8:                 break
9:             end if
10:        end for
11:        if match is False then
12:            add_observation_in_state(Complement, O2)
13:        end if
14:    end for
15:    Return Complement
16: end procedure
```

Algorithm 6 describes the process to add an observation in a given state.

Algorithm 6 Complement World State

```
1: procedure add_observation_in_state(WolrdState State, Observation O_new)
2:     match = False
3:     for each observation O in State do
4:         if each property P of O is same as P_n in O_new then
5:             match = True
6:             break
7:         end if
8:     end for
9:     if match is False then
10:        add O_new to the list State
11:    end if
12: end procedure
```

The algorithm compares all the existing observations in the state with the new observation. Any match leads the algorithm to ignore the observation and return without adding it into the state. In case of a mismatch the algorithm adds the observation into the list of observations, the state.

3.5.3 Adding a Schema

A newly created schema may match an existing schema exactly, may contain different/additional information, may contain subset of information or may be totally new. Adding a schema into the (long-term) memory is referred to in block 6 of Figure 3.12. Algorithm 7 describes the

mechanism for adding a schema to the memory.

Algorithm 7 Adding Schema to memory

```
1: procedure add_schema_to_memory (Schema S)
2:     for each Schema S' in Memory do
3:         if Schemas_match (S, S') then
4:             Return
5:         end if
6:     end for
7:     Add Schema S in the Memory
8:     Return
9: end procedure
```

Before adding the schema to the memory, the algorithm checks if the new schema matches any of the existing schemas in the memory. The newly created schema will be discarded if it matches any of the existing schemas and the system move onto the next stage. Algorithm 8 describes the underlying mechanism in schema matching.

Algorithm 8 Matching two schemas

```
1: procedure schemas_match (Schema S1, Schema S2)
2:     if S2 is generalised then
3:         instantiate schema (S2, S1_preconditions)                ▷ See Algo. 16
4:     end if
5:     if NOT states_match(S2_postconditions, S1_postconditions) then
6:         Return False
7:     end if
8:     if NOT states_match(S2_preconditions, S1_preconditions) then
9:         Return False
10:    end if
11:    if NOT actions_match(S2_action, S1_action) then
12:        Return False
13:    end if
14:    Return True
15: end procedure
```

If the existing schema is generalised then any generalised properties are instantiated with the newly created schema preconditions. The instantiation algorithm identifies the generalised properties in the schema and replaces their values with concrete values in the observations of the given world state. The instantiation algorithm initially instantiates the preconditions. The generalised variables in postconditions are instantiated with same concrete value as the generalised variables in preconditions that have matching generalised value e.g. $a. The instantiation algorithm is further discussed in Chapter 4. State matching uses

the same algorithm as world state matching described in Algorithm 3. Algorithm 9 describes the mechanism for matching two actions.

Algorithm 9 Matching Actions

```
1: procedure actions_match (Action A1, Action A2)
2:     for each property P2 in Action A2 do
3:         for each property P1 in Action A1 do
4:             if P2 NOT same as P1 then
5:                 Return False
6:             end if
7:         end for
8:     end for
9:     Return True
10: end procedure
```

Algorithm 9 compares all the properties (if any) of two actions and returns false if any of the properties do not match, otherwise returns true.

3.5.4 Generalisation

Dev-PSchema is also capable of generalising learning outcomes, i.e. schemas, using *inductive inference*, which is based on evidences/samples. The system goes under generalisation every time it creates a new schema, shown as block 7 in Figure 3.12. A generalised schema is developed from a set of schemas having similar actions e.g., reach x 2.20, y -0.25 and reach x -1.20, y -0.05. The generalisation for a given property is said to be true universally if it is true for a whole set of samples. Observation properties in preconditions and postconditions which have different values, in all the sample schemas selected for generalisation, are generalised. We have extended the generalisation mechanism in Dev-PSchema. Where the properties of pre and postconditions are numeric, the generalising mechanism can determine the some linear mathematical relationship between the variables used in preconditions, postconditions and actions. Previously in PSchema [173], the generalisation was limited to recognising change in these values, but not identifying relationships in terms of how the values may have changed. The generalisation mechanism is shown as block 7 in Figure

3.12 of play mode. Figure 3.13 illustrates an example of generalising with two reach schemas.

Experience 1

Preconditions	Action	Postconditions
<Observation = proprioceptive, hand = 3.0, grip = 0, x = 1.0, y = 1.0> <Observation = colour, value = red, size = 40, x = 3.0, y = 5.0>	Reach (3, 5)	<Observation = proprioceptive, hand = 3.0, grip = 0, x = 3.0, y = 5.0> <Observation = visual, colour = red, size = 40, x = 3.0, y = 5.0>

Experience 2

Preconditions	Action	Postconditions
<Observation = proprioceptive, hand = 3.0, grip = 0, x = 2.0, y = 2.0> <Observation = colour, value = green, size = 60, x = 4.0, y = 4.0>	Reach (4, 4)	<Observation = proprioceptive, hand = 3.0, grip = 0, x = 4.0, y = 4.0> <Observation = visual, colour = red, size = 60, x = 4.0, y = 4.0>

Generalised Schema

Preconditions	Action	Postconditions
<Observation = proprioceptive, hand = 3.0, grip = 0, x = $a, y = $b> <Observation = colour, value = $c, size = $d, x = $e, y = $f>	Reach ($e, $f)	<Observation = proprioceptive, hand = 3.0, grip = 0, x = $e, y = $f> <Observation = visual, colour = $c, size = $d, x = $e, y = $f>

Fig. 3.13 Generalised schema obtained from 2 concrete schemas having similar action.

Furthermore, Dev-PSchema generalises properties by the relevant perception types. For example if a property of a colour perception is found to be suitable for generalisation then a common generalised variable will be used to replace the property in all of the colour perceptions that have matching value. For example if a sample schema contains two colour perceptions and they both have matching values for a property, then their values will be replaced by a generalised variable, e.g. $a, if the property is found to be eligible for generalisation. In PSchema a generalisation was based on the property rather than type of the perceptions. Thus if a property in a perception is found to be suitable for generalisation then it will generalise all the property in all perceptions in the sample schema, irrespective of the perception type. If any properties have the same value in all the sample schemas then it remains un-generalised. A generalised schema with at least a single un-generalised property is referred here as a *partially generalised* schema. Generalisation is further discussed in Chapter 4, along with the relevant algorithms and experiments.

Chapter 4

Generalising from Experiences

Generalisation helps to extend learnt behaviours and knowledge toward new situations based on similarities with previous experiences. Thus, generalisation is not about finding no difference between the situations but developing an understanding of the situations that have similarity in consequences [174]. The ability to generalise enables infants to learn and develop knowledge from a few experiences then use it to understand and predict outcomes in novel situations. If the generalised concepts fail, infants adapt and adjust their learning [146]. Thus, generalisation not only helps to reuse knowledge with novel objects and environments but also helps to predict or anticipate the outcome of actions in a situation based on previous experiences with similar situations.

Possessing a complex cognitive system, humans are able to anticipate events and their outcomes. This capability is developed from experiences and has been observed in infants, as young as 3 months old [30]. Anticipation in infants plays an important role in extending understanding of the effects caused by actions on different objects through repeated experiences and generalisation. This capability plays a very important role in an infant's learning process [142]. There is evidence in developmental psychology supporting anticipation of outcomes observed using visual attention in young children and infants.

In developmental psychology, infants are understood to look longer at new or unexpected events or situations after achieving habituation. However, infants have been observed to show a preference for familiar objects rather than the novel, depending upon the time spent with the familiar object [159]. Habituation in infants is achieved through presenting the same object, event or situation to them until they lose interest. Following habituation, infants gaze at new objects, events or situations longer than the habituated [177]. Schlesinger and Casey [167] found that 6 months old infants look longer at impossible events than possible events. The longer looking time provides evidence that the infants anticipated the event results i.e., possible event, however, the outcome of the event i.e., impossible event, is novel and did not meet the expectations. In a similar experiment, Adler and Haith [2] found that 3 month old babies can also predict visual events. During the experiments, infants were habituated to set visual events. After habituation, they were presented with a selection of novel events varying in visual similarity to the habituated events. Authors found that infants anticipation were higher when the novel event had a higher level of similarity to the habituated event. The greater the visual variation in the novel event, as compared to the habituated event, the less predictable the outcomes were. This provides evidence that the anticipation of events and action outcomes depend upon the similarity between the novel situation and previous experiences. Thus infants anticipated event outcomes for new events based on generalisation developed from previous experiences.

The world is full of static and dynamic objects. If one wants to catch a moving object, (s)he needs to predict the object's motion in spatio-temporal space and move the hand to an anticipated position to catch the object. For static objects, (s)he will expect that the object will remain at the same position while in a dynamic situation the object position is predicted over time using previous experiences in similar situations. This behaviour is developed at an early age in humans. von Hofsten et al. [196] found that at 6 months old infants can predict the position of a moving object and use that information to reach for the object. Similarly, Canfield and Haith [30] found that 3.5 month old infants are able to anticipate dynamic symmetric and asymmetric visual events based on their understanding developed based on their experiences.

These studies demonstrate that infants not only possess visual event anticipation but also perform actions based on the anticipation [138]. Thus infants develop generalisation for objects, events and situations and use this knowledge in planning and performing their actions.

This concludes that generalisation is not just used to help anticipate events, it is also very important in developing understanding and knowledge to perform actions in static as well as dynamic situations. Anticipation consists of two main sequential steps; i) observing and predicting the outcome of an event within a static or dynamic scenario, and ii) planning and executing the behaviour based on the prediction. The generalisation capability, in this situation, helps to anticipate the outcome of actions and events based on their level of similarity to previous experiences in similar situations.

A robotic learning model inspired from developmental psychology should also be able to generalise object related actions to efficiently make use of its learning from one object to another, based on similarity in object features. In a real environment, a robot may need to interact with several objects using different actions. The objects in the environment may be similar to each other and the actions may be repetitive. If the robots need to learn each object-action relation then its learning model can be seen as inefficient. Thus for an efficient learning model, the model should be able to generalise learning from a few examples and apply them to anticipate the outcomes for various objects and situations based on similar features to the generalised experiences. The ability to recognise similar objects and features is an important cognitive development. However, in this work, we focus on the high-level decision making process rather than the underlying mechanism for object recognition.

Dev-PSchema is capable of creating generalised schemas based on experiences in a given environment. The system generalises through inference using previously learnt schemas that performed the same action. We evaluate the generalising mechanism of Dev-PSchema by performing two experiments based on generalisations developed through performed actions. Such generalisations are developed through receiving visual perceptions related to the per-

formed actions in a given situation. In the first experiment the agent makes inferences for generalisation based on visual perception and non-visual (e.g. squeezable, making (perceptual) sounds) perception obtained through actions. It should be noted that these experiments are performed on the *Sandbox* simulator and all the sensory perceptions, including auditory, are simulated representations. In this experiment, we demonstrate that Dev-PSchema is able to extend learning to anticipate action outcomes about objects that are defined by their obvious and non-obvious properties. In the second experiment, we demonstrate the capability of the agent learning visual movements of objects following an action and develop generalised schemas. The generalised schemas will later be used to anticipate action outcomes in the similar environments. The agent makes initial predictions about the changes by applying mathematical functions to the variables of properties used in the sensory state and the action.

In Section 4.1 we provide a detailed discussion on generalisation mechanism and algorithms used for creating a generalised schema. In this section, we also provide the *instantiation* algorithm used to instantiate perceived features in a generalised schema to predict its outcome. We discuss experiments with the results in Section 4.2, followed by the discussions of this chapter in Section 4.3.

It should be noted that this Chapter includes parts of two peer reviewed published papers, given below:

- Kumar, S., Shaw, P., Lewkowicz, D., Giagkos, A., Shen, Q., Lee, M. (2016, September). Developing object understanding through schema generalisation. In Developmental and Learning and Epigenetic Robotics (ICDL-EpiRob), 2016 Joint IEEE International Conference on (pp. 33-38). IEEE.

- Kumar, S., Shaw, P., Lewkowicz, D., Giagkos, A., Lee, M., Shen, Q. 2016. Generalising Predictable Object Movements Through Experience Using Schemas. In E. Tuci., A. Giagkos., M. Wilson., J. Hallam. (eds) From Animals to Animats 14: 14th International Conference on Simulation of Adaptive Behaviour, SAB 2016, Aberystwyth,

UK, August 23-26, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9825 14th International Conference on Simulation of Adaptive Behaviour. Springer Nature pp. 329-339.

4.1 Generalisation

Dev-PSchema uses similar schemas, especially having a similar action, to create a generalised schema. Figure 3.13 shows an example of generalised schema from two schemas having a similar action. Dev-PSchema generalises two different levels of schemas based on sensory perceptions present in previous similar experiences. If any of the observation properties in the generalised schema remain un-generalised, such a schema is described as a "partially" generalised schema. Whereas a generalised schema with all properties generalised in it is described as a "completely" generalised schema.

Furthermore, Dev-PSchema is also able to infer a linear functional relation between numerical values of the properties in the schemas used for generalisation. The generalisation mechanism checks for linear additive i.e., $+/-$, relationships between the properties in preconditions and postconditions of a schema. This capability helps to anticipate numerical values of postconditions for a perceived environment using a generalised schema. This feature can be enabled/disabled using a flag when starting Dev-PSchema.

Dev-PSchema performs generalisation process every time a new schema is added. The newly created schema is considered as a *sample* schema to find any similar schemas in the memory for generalisation. Algorithm 10 describes the underlying mechanism of creating a generalised schema.

Algorithm 10 begins by finding the schemas that use a similar action to the sample schema. If functional generalisation is set as *false* then the action similarity is calculated just by comparing types of actions i.e., reach, grasp, release etc., irrespective of coordinate values

Algorithm 10 Generalising Schema algorithm

```
1: procedure generalise (Schema S', Boolean Functional )
2:     Similar_schemas = list of similar_schemas(S', Functional)                ▷ See Alg. 11
3:     if length(Similar_schemas) ≥ generalising_threshold then
4:         S = Copy of S'
5:         for each property P of each Observation O in Schema S do
6:             p_found = False
7:             for each property P2 in type(O) Observation in Schema Si from Similar_schemas do
8:                 if type (P) = type (P2) && Value(P1) ≠ Value(P2) then
9:                     if Functional is True then
10:                        β = P(Value)S_preconditions − P(Value)S_postconditions
11:                        replace P(Value) in S_preconditions with "$α"        ▷ α is an unique character
12:                        replace P(Value) in S_postconditions with "$α + β"
13:                        break; Go to line 5
14:                    else
15:                        replace P(Value) in S_preconditions & S_postconditions with "$α"   ▷ α is an
   unique character
16:                        break; Go to line 5
17:                    end if
18:                end if
19:            end for
20:        end for
21:        Sim = 0                                  ▷ To verify generalisation for adding into the Memory
22:        for each Schema Si in Similar_schemas do
23:            instantiate_schema from_WS (Schema S, WS Si_preconditions)        ▷ See Section 4.1.1
24:            Add schema_similarity (Si, S) to Sim                              ▷ See Algorithm 12
25:        end for
26:        Sim = Schema_Similarity/length(Similar_schemas)
27:        if Sim > similarity_threshold then                    ▷ similarity_threshold = 0.75
28:            S = sort_associations (generalised_schema S, similiar_schemas Similar_schemas)
29:            Add generalised schema S in the Memory
30:        end if
31:    end if
32: end procedure
```

and other properties. If functional generalisation is set as *true*, the algorithm undergoes through a different mechanism to find similar schemas that depends upon the functional relationship between the observation properties in the schema preconditions and postconditions. Algorithm 11 describes the underlying mechanism of finding similar schemas schema.

For functional generalisation, Algorithm 11 finds all relationships present between preconditions and postconditions of the sample schema. A relationship is the difference between the values of a property, of same observation type, present both in preconditions and postconditions of the schema. For example, the size of the colour observation changes from 25 to 20 between preconditions and postconditions of a schema, respectively. The functional relationship between the two is represented as -5 units. The algorithm considers a schema from the memory similar to the sample schema if all the functional relationships in the

Algorithm 11 Finding similar Schemas in the memory

```
1: procedure similar_schemas(Schema S', Boolean Functional)
2:     Similar_schemas = empty list
3:     functions_sample = find_function (S')                    ▷ See Equation 4.1
4:     for each Schema S in the Memory do
5:         if Functional is False then
6:             if Schema S_action type same as Schema S'_action then ▷ e.g., a reach action in both schema
7:                 Add Schema S to the list Similar_schemas
8:                 Go to line 4
9:             end if
10:        else
11:            functions_S = find_function (S)                  ▷ See Equation 4.1
12:            for each function F1 in functions_sample do
13:                similar_found = False
14:                for each function F2 in functions_S do
15:                    if F1 is same as F2 then
16:                        similar_found = True
17:                        break; Go to line 12
18:                    end if
19:                end for
20:                if similar_found is False then
21:                    Go to line 4
22:                end if
23:            end for
24:            if similar_found is True then
25:                Add Schema S to the list Similar_schemas
26:                Go to line 4
27:            end if
28:        end if
29:    end for
30:    Return list Similar_schemas
31: end procedure
```

sample schemas are also present in the schema from the memory. Equation 4.1 describes the mechanism to calculate a relationship for a property P of an observation O_n.

$$Func(O_n,\ P)_m\ =\ \delta_{Preconditions}\ -\ \delta_{Postconditions} \tag{4.1}$$

where $Func(O_n,\ P)$ represents a function calculated for the value δ of the property P in observation O of type n given in preconditions and postconditions. A list of functions is created for total m properties, having numerical values, in observations present in preconditions of the sample schema S', given in Algorithm 10.

Following finding similar schemas, the generalisation mechanism proceeds further if the number of similar schemas is greater than or equal to the threshold. In this work we set a threshold of two schemas to create a generalised schema. The threshold can be varied, depending upon the application and experiments, through the interface controller shown in Chapter 3, Figure 3.1.

The algorithm then takes each of the property values present in the preconditions, post-conditions and action of the sample schema and then compares them with the value of the same property type in similar schemas. A difference of value is considered as non-essential for the specific action hence is generalised. However, a match suggests the property is more relevant in the actions and its effect, hence remains un-generalised leaving the schema partially generalised. If a property is found to be suitable for generalisation, Algorithm 10 generalises the property. In the case of *functional* generalisation, the algorithm finds the property relationship between the values of the property given in preconditions and postconditions of the schema. Equation 4.1 is used to find the relationship for the properties in the sample schema. A property is generalised by replacing its value with a unique alphabetic character in preconditions and with a unique character and the functional relational relationship found for that property in postconditions of the sample schema, see line 12 in Algorithm 10. It should be noted that the properties are only generalised with the $+/-$ additive relationship between their values given preconditions and postconditions of the schema.

In the case of non-functional generalisation, Algorithm 10 generalises properties by replacing their values by a unique alphabetic character, alongside the $ symbol, in both preconditions and postconditions of the sample schema, see line 15 in the algorithm. The $ indicates that the generalised property is either irrelevant for the schema outcomes or its variations do not cause an impact on the outcome of the generalised schema.

Once all the properties present in the sample schema have been considered in the generalisation process, the algorithm calculates the similarity between the newly developed

generalised schema and similar schemas to verify the generalisation. Before calculating the similarity between two schemas, the generalised schema is instantiated with the schema preconditions each of the schema in turn in the similar schemas list. The instantiation mechanism puts the concrete values i.e., an instance value, in the generalised schema with the schema preconditions, with which similarity is being calculated. The generalised post-conditions are instantiated with either the same value as the preconditions or the same value with -/+ the functional relation, depending upon if the functional generalisation has been used in the schema. For example, if a property named size of an observation is used as "$a" in the preconditions and "$a+5" in the postconditions, then a concrete value of 20 from similar schemas will replace the generalised values with 20 and 25 in the preconditions and postconditions respectively. Section 4.1.1, describes in more details the mechanism for instantiating a generalised schema from a given world state. The similarity check provides a test of how much the newly generalised schema matches each schema in the list of similar schemas. Higher similarity confirms the generalised schema being developed is a generalised concept using individual instances in the list of similar schemas. Algorithm 12 describes the mechanism for calculating the similarity between two given schemas. A generalised schema helps to re-use and predict the outcomes through instantiating the generalised schema with the perceived state of the environment. Through instantiation, a generalised schema will provide higher excitation as it provides higher similarity with a perceived state for the excitation calculator. The excitation calculator is an internal mechanism of Dev-PSchema that finds a suitable action to perform in a perceived situation of its environment, see Section 5.1 for details.

Algorithm 12 Schemas Similarity

1: **procedure** *schemas_similarity* (*Schema* S_1, *Schema* S_2)
2: $Sim = 0$
3: *Add* **states_similarity**(S_2*_postconditions*, S_1*_postconditions*) *to Sim* ▷ See Algorithm 13
4: *Add* **states_similarity**(S_2*_preconditions*, S_1*_preconditions*) *to Sim*
5: $Sim = Sim/(Sum\ of\ Number\ of\ observations\ in\ preconditions\ and\ Postconditions\ in\ S_2)$
6: *Return Sim*
7: **end procedure**

Algorithm 12 calculates the similarity between the preconditions world state (WS) and postconditions WS separately. The average similarity is returned for further process. Algorithm 13 describes the mechanism for calculating the similarity between the two given world states.

Algorithm 13 States' Similarity

```
1: procedure states_similarity (WolrdState WS1, WorldState WS2)
2:     Sim = 0
3:     for each observation O2 in WS2 do
4:         Max_sim = 0.0
5:         for each observation O1 in WS1 do
6:             O_sim = sbservation_similarity (O1, O2)          ▷ See Algo. 14
7:             if O_sim > Max_sim then
8:                 Max_sim = O_sim
9:             end if
10:        end for
11:        Add Max_sim to the Sim
12:    end for
13:    Sim = Sim/Number of Observations in WS2
14:    Return Sim
15: end procedure
```

Algorithm 13 returns maximum similarity between the value of each property in observations of a given WS by comparing it with the property value present in similar observation type. For example, colour observation in one WS will only be compared with a colour observation in the other WS. The algorithm will return with maximum similarity (1.0) if the an observation is present in both world states. Algorithm 14 describes the mechanism for the observation similarity calculation.

Algorithm 14 returns maximum similarity i.e., 1.0, if two observation are the same. Minimum value i.e., 0, is returned if two observation of different type and their properties do not match at all. A small tolerance (10%) has been introduced to handle noise in the iCub's perceptions. At this tolerance value, two values of a property are considered equal if the difference between them is within $\pm 10\%$ of the (P_2) value.

The newly created generalised schema in Algorithm 12 is instantiated and compared with each of the schema present in the list of similar schemas. An average of the similarities

Algorithm 14 Observation Similarity

```
1: procedure observation_similarity (Observation O1, Observation O2)
2:     Sim = 0
3:     for each property P2 in O2 do
4:         for each property P1 in O1 do
5:             if type(P1) is same as type(P2) then          ▷ Types include colour, shape etc.
6:                 if P1(value) = P2(value) then             ▷ ±10% of P2(value) in case of iCub
7:                     Sim += 0.5
8:                 end if
9:                 Sim += 0.5
10:            end if
11:        end for
12:    end for
13:    Sim = Sim/Number of properties in O2
14:    Return Sim
15: end procedure
```

calculated for all schemas in the list of similar schemas is used further in the algorithm. If the average similarity is more then the threshold than the new generalised schema is added into the memory. In this work, we used similarity of threshold 0.75, however it can be varied by the user. A higher similarity threshold will result in a generalised schema with the higher tendency of inference representing all the schemas used for generalisation i.e., list of similar schemas. If the generalised schema fails to meet the similarity threshold that means the generalised schema does not represent a general inference for all the schemas in the list of similar schemas. In that case, each schema in the list of similar schemas represents an individual inference for a specific example.

Following the similarity calculations, Algorithm 10 sorts associated observations in the generalised schema. Algorithm 15 describes the mechanism to find associated observations in the generalised schema.

Algorithm 15 begins by creating a new empty schema to copy information from the generalised schema. In the start, it copies the generalised schema action to the newly created schema. The mechanism then takes each observation in the generalised schema preconditions and checks if this type of observation exists in the each of similar schema preconditions. If the observation is present in all schemas then it is added to the preconditions of the new schema. If in the observation is not present in any of the similar schemas then it is added

Algorithm 15 Sorting associated observations in a generalised schema

```
1: procedure sort_associations(generalised_schema S, similiar_schemas Similars)
2:     S_s = New schema
3:     S_s_action = Copy(S_action)
4:     for each observation O_1 in schema S_pre/post_conditions do
5:         found = False
6:         for each schema S' in Similars do
7:             for each observation O_2 in schema S'_pre/post_conditions do
8:                 if type(O_1) = type(O_2) then
9:                     found = True; break
10:                end if
11:            end for
12:            if found = False then
13:                break
14:            end if
15:        end for
16:        if found = False then
17:            Add O_1 to schema S_s_associated_pre/post_conditions
18:        else
19:            Add O_1 to schema S_s_pre/post_conditions
20:        end if
21:    end for
22:    Return S_s
23: end procedure
```

to the associated preconditions. The same process is repeated for the generalised schema postconditions to find the concrete and associated postconditions for the new schema. The mechanism returns the new schema at the end of the process.

4.1.1 Instantiation

In PSchema, the instantiation was performed by putting concrete values (un-generalised values) from a provided WS into the generalised properties. The PSchema instantiation mechanism replaces each generalised property in the generalised schema with the concrete value of the same property from the given WS, irrespective of their observation type. Dev-PSchema instantiation mechanism instantiates the properties by finding concrete values in a similar observation type of the given state. Property values, in the observations, with $ signs are replaced with specific values from the concrete state, resulting in a prediction of the outcomes for the action in the generalised schema. Algorithm 16 describes details mechanism of instantiation.

Algorithm 16 Instantiating generalised schema from a WS

```
1: procedure instantiate_schema_from_WS (Schema S, WorldState WS)
2:     h_WS = Observations from WS with coordinates same as hand coords in WS
3:     h̄_WS = Observations from WS with coordinates different from hand coords in WS
4:     sort observations in excitation order (h̄_WS)                ▷ See Algo. 17
5:     S_WS = Observations from S_preconditions with coordinates same as hand coords in S
6:     S̄_WS = Observations from S_preconditions with coordinates different from hand coords in S
7:     instantiation_values = Empty map
8:     for each observation O_s property P_s in S_WS do
9:         for each observation O_s property P_ws in h_WS do
10:            if P_s same as P_ws then
11:                replace the generalised value v of P_s with P_ws value u
12:                add P_s as key instantiation_values with Pair (index v, value u)
13:            end if
14:        end for
15:    end for
16:    for each observation O_s property P_s in S̄_WS do
17:        for each observation O_s property P_ws in h̄_WS do
18:            if P_s same as P_ws then
19:                replace the generalised value v of P_s with P_ws value u
20:                add P_s as key instantiation_values with Pair (index v, value u)
21:            end if
22:        end for
23:    end for
24:    for each observation property P_s in S_postconditions do
25:        if instantiation_values has key P_s then
26:            Pair P = Pair in instantiation_values with key P_s
27:            if Pair P index is same as P_s value v then      ▷ Any other property will remain generalised
28:                replace value v in schema with Pair P value
29:            end if
30:        end if
31:    end for
32:    Return
33: end procedure
```

Dev-PSchema instantiation algorithm takes hand/manipulator coordinates as a reference for instantiation. Initially, it instantiates all observations of the generalised schema preconditions and postconditions that have the same coordinates as the hand/manipulator with the observations from the given WS. If a generalised property contains any functional relationship, its value is replaced accordingly. For example, a property in an observation to be instantiated containing "$a" and "$a+5" in the preconditions and postconditions of a generalised schema respectively. Let's say the instantiation mechanism finds 10 *units* as the value for the property, which leads to property values of 10 and 15 *units* (i.e., $10+5$ replacing $\$a+5$) in the preconditions and postconditions respectively. Later, all the observations in the preconditions and postconditions of the generalised schemas at a position other than the hand are instantiated with observations from the given WS that are not at the hand position. Figure 4.1 shows an example of an instantiation for a generalised schema in PSchema and

Dev-PSchema systems.

Generalised Schema

Preconditions	Action	Postconditions
Observation=proprioceptive, hand=3.0, grip=0.0, x=$a, y=$b Observation=colour, value=$c, size=$d, x=$a, y=$b Observation=shape, value=$e, size=$f, x=$a, y=$b Observation=touch, average=0.5	**Grasp**	Observation=proprioceptive, hand=3.0, grip=0.75, x=$a, y=$b Observation=visual, colour=$c, size=$d, x=$a, y=$b Observation=shape, value=$e, size=$f, x=$a, y=$b Observation=touch, average = 0.75

WS for Instantiation

Observation=proprioceptive, hand=3.0, grip=0.0, x=2.0, y=3.0
Observation=visual, colour=red, size=30, x=2.0, y=3.0
Observation=shape, value=cube, size=25, x=2.0, y=3.0
Observation=touch, average=0.5

Dev-PSchema instantiation

Preconditions	Action	Postconditions
Observation=proprioceptive, hand=3.0, grip=0.0, x=2.0, y=3.0 Observation=visual, colour=red, size=30, x=2.0, y=3.0 Observation=shape, value=cube, size=25, x=2.0, y=3.0 Observation=touch, average=0.5	**Grasp**	Observation=proprioceptive, hand=3.0, grip=0.75, x=2.0, y=3.0 Observation=visual, colour=red, size=30, x=2.0, y=3.0 Observation=shape, value=cube, size=25, x=2.0, y=3.0 Observation=touch, average = 0.75

Fig. 4.1 Instantiating a generalised schema from a given state (WS)

Figure 4.1 shows an example instantiation for a generalised schema from a given WS in Dev-PSchema systems. In Dev-PSchema each observation is instantiated with a value matching the observation type. Hence, colour and shape observations are instantiated individually with colour and shape observations from the given WS respectively.

Furthermore, Dev-PSchema also makes use of coordinates during the instantiation process. Taking hand/manipulator coordinates as reference, it instantiates all observations at the hand position. The observations having coordinates different to the hand coordinates are sorted according to their excitation, see Section 5.1 for the observation excitation. The observations with the highest excitation are used first for instantiating the generalised schema. Algorithm to calculate observation excitation is described in Chapter 5, Algorithm 15.

4.2 Generalisation: Experiment and Results

To demonstrate and evaluate the generalisation mechanism in Dev-PSchema we performed two experiments related to types of generalisation and anticipation with instantiation using function generalisation. To demonstrate generalisations, partial and complete, we used a limited set of the perceptions, where the perception of self is provided with a colour perception for the agent's hand, without the grip, and its positions. The touch perception, obtained while touching or grasping, only appears when an object is touched, without the average touch value, see Section 3.3 for details. A new perception, "holding" replaces the touch perception when an object is grasped. Both, touch and holding perceptions are binary and without any properties. These changes will demonstrate the generalisation based on the visual perceptions only. In these experiments, a Dev-PSchema enabled agent uses an excitation system to select a suitable action to perform in the environment. The excitation system calculates excitation for all the schemas/actions present in the memory to interact with the environment. When an object is introduced in the simulator, the excitation system activates the action schema relevant to the current sensory state. For example, an object at position (1, 1) will trigger the system to fixate/reach at that position, due to recalling the schema where it previously fixated/reached that position. The excitation system is discussed in detail in Chapter 5.

4.2.1 Experiment 1: Object Understanding through Generalisation

This experiment demonstrates the generalising capability of Dev-PSchema for developing a basic concept of affordance through exploratory play. The experiment consists of five stages performed in a simulated environment *Sandbox*. The first two stages of the experiment, bootstrapping and object familiarisation, provide the foundation for the schema knowledge at the beginning of the experiment, see Section 3.1.1. During the bootstrapping stage, schemas for basic actions (saccade, reach and grasp) are created through motor babbling. In the object familiarisation stage, an object is presented in the environment for interaction. The agent uses basic action schemas to explore the object. This stage ends at the point where

the agent grasps the object. The last three stages are described here as three parts of the experiment. In the first part, a new/second object is introduced enabling the system to build some generalisation based on the experiences with both objects. The 2nd and 3rd parts are designed to evaluate the failure and success of the generalised schemas, as well as the agent's capability to re-learn and adapt. The flow of this experiment is shown in Figure 4.2.

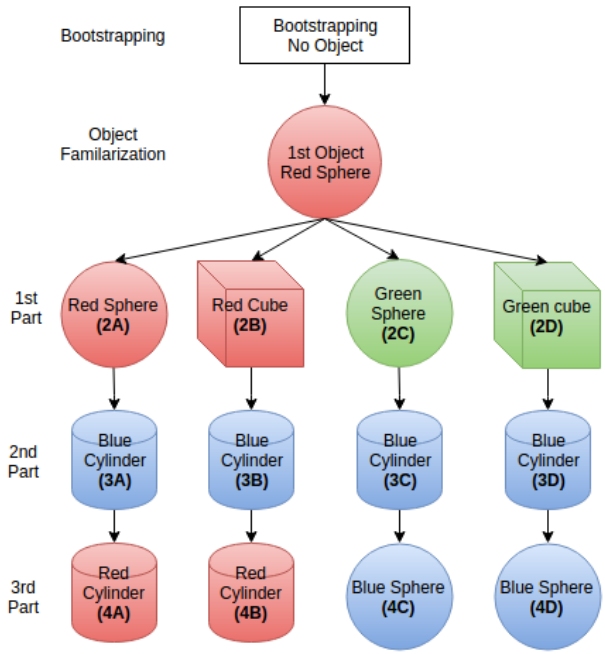


Fig. 4.2 Experiment flow diagram

The environment for this experiment is set in the Sandbox simulator and organised as a 5×5 grid of discrete cells. An object is contained in a single cell, with no overlap. A simulated hand (manipulator) is used as an end effector to interact with the objects in the environments. It is capable of reaching towards 9 reach positions (defined by a 3×3 area), which are a subset of the total positions in the world. Visual information is observed and grouped in two different ways, namely by colour and shape along with a 2D position in the space, x and y dimensions respectively. Dev-PSchema receives a "touch" observation (perception) from the Sandbox when the hand performs a grasp action or touches an object placed in the same position as the hand. Each object in the simulator is represented by a colour, shape, position and stimulus response when it is grasped. We divided the objects

in the environment into two categories. The first category consists of spheres and cubes, irrespective of their colour, which responds with a "sound" observation when grasped. In the second category, objects consist of just cylinders with various colours and respond with a "light" observation when grasped. Sandbox prepares all sensory information, represented as high-level perceptions and shares with Dev-PSchema. Similarly, actions are defined in high-level language i.e., reach, grasp etc. In this experiment, we used "Reach", "Grasp" and "Saccade" actions, see Section 3.3 for details.

4.2.1.1 Bootstrapping

As mentioned before, the system is initially bootstrapped to enable it to saccade and reach towards positions in the space provided by the simulator. Notice that the reach space is a subset of the overall space, as is the case with infants at non-locomotion stage. Each of the positions in the space is observed through saccades and fixations, and visited by the hand. Moreover, when the hand moves followed by the eye movement, the system stores schemas for both the reach and saccade actions. In the end, the grasp action is performed and a schema is recorded for it. It should be noted that no preconditions of performed actions are recorded during bootstrapping, as behaviours are considered to be (reflexive) motor babbling actions rather than stimulated through some perceptions. Thus this stage provides a basic set of action schemas enabling the agent to start to interact with the objects in the environment.

4.2.1.2 Object Familiarisation

The object familiarisation stage follows the bootstrapping stage, in which a red sphere from category 1 is introduced. The Sandbox sends the colour and shape perceptions of this object to the agent (Dev-PSchema). Using the recently learnt schemas, from bootstrapping, the excitation calculator finds the saccade action towards the object position to be most excited followed by a reach action to the same position. Once the eye and hand are at the position of the object, the grasp action is found to be most the excited. The hand position in the

bootstrap grasp schema (x=3, y=3 in Figure 4.3) is different from the current position of the object (x=1, y=1). However, schemas that do not specify coordinates as part of their action, e.g., grasp, push, etc., can be applied in any position of the environment, taking the position of the hand instead. Therefore, based on the similarities between the observations and postconditions, ignoring coordinates, along with schema statistics related to previous executions, the bootstrap grasp schema receives the overall highest excitation of all the schemas currently in the memory. Schema statistics are discussed in detail in Section 3.3.3. Figure 4.3 shows the new schema learnt, following the grasp action providing additional observations that were not anticipated by the existing bootstrap grasp schema.

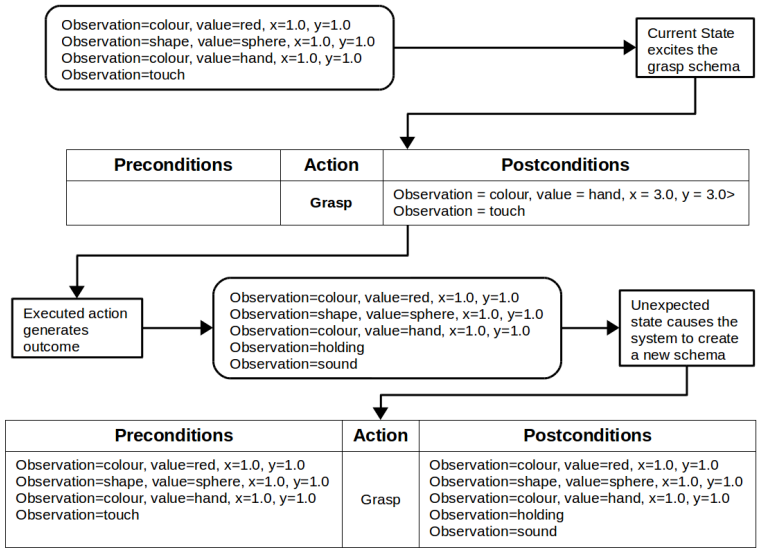


Fig. 4.3 Grasping the first object in the environment during the object familiarisation

A new *grasp schema* is created using and extending the basic bootstrap schema, coupled with new observations acquired after the action. It should be noted that the new schema also contains the *sound* observation, a high-level representation of a sound resulting from the grasp action. The system stores this schema in the memory and ignores the generalisation process as there is only one grasp schema in the memory, excluding bootstrap schemas. Once this process is completed the object is removed from the environment in order to prepare for the next stage of the experiment.

4.2.1.3 First Part: Building Generalisation

The agent builds generalised schemas based on the object interactions performed previously and at this stage. Another object from the same category is introduced in the environment varying in shape and/or colour as compared to the initially experienced object. Thus from this stage and onwards four different branches are generated as shown in Figure 4.2. We introduce different objects varying in either shape or colour or both to demonstrate the variation in generalisation based on the experiences. This may be seen as a way to expose different individual infants to different objects after having a similar experience with the first object. Figure 4.4 shows the process of building schemas when a second, new object (2A or 2C as they are tagged) of the same shape and similar or different colour are introduced.

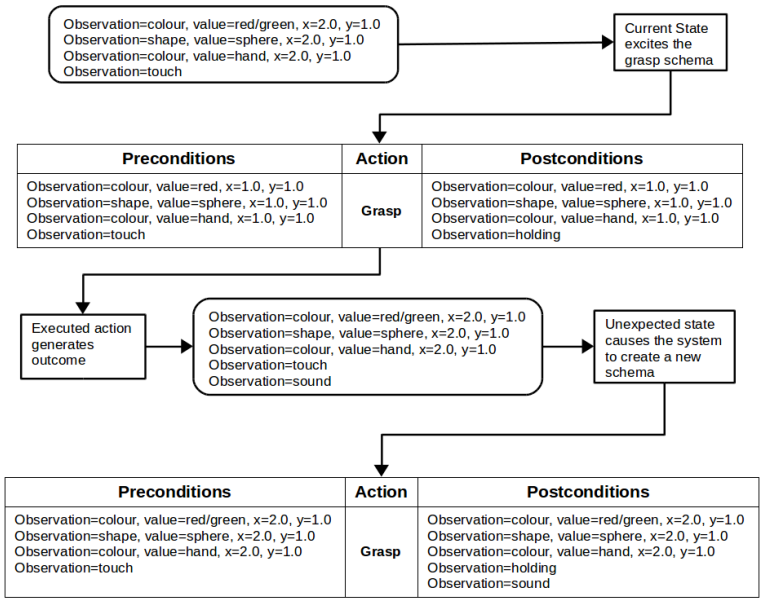


Fig. 4.4 Second object, same shape

Introducing the second object with a different or matching shape at a different position, excites the agent to reach and grasp, and create new schemas. The new grasp schema is un-generalised. However, creation of the two similar grasp schemas, when the first and the second objects are being presented, triggers the generalisation mechanism based on the information the agent has managed to collect. In the case of 2*A*, only the position of the

object is different, and in 2*C* the colour is different. As the other properties are the same, they are not generalised so only a partially generalised schema is produced for both cases. In the case of 2*C*, the change in the object colour triggers the generalisation mechanism to generalise the colour property. However, in the case of 2*A*, the previously perceived hand colour causes the mechanism to generalise the colour property. Similarly, in the case of 2*B* and 2*D*, shape and position of the object varies from the previous schema, causing the agent to generalise the shape and position. The change in the colour of the object at 2*D* also causes the agent to generalise the colour property. Therefore, the agent creates a complete generalised schema, generalising colour, shape and position of the object, in both cases i.e., 2*B* and 2*D*. The agent undergoes the same schema building process as to that of Figure 4.3 and the grasp schema created with the first object becomes the excited schema. Figure 4.5 shows the schema built when a second object (2*B* or 2*D* as tagged) of different shape and of a same or different colour from the initially experienced object, is introduced.

Preconditions	Action	Postconditions
Observation=colour, value=red/green, x=2.0, y=1.0 Observation=shape, value=cube, x=2.0, y=1.0 Observation=colour, value=hand, x=2.0, y=1.0 Observation=touch	Grasp	Observation=colour, value=red/green, x=2.0, y=1.0 Observation=shape, value=cube, x=2.0, y=1.0 Observation=colour, value=hand, x=2.0, y=1.0 Observation=holding Observation=sound

Fig. 4.5 Schema generated for the second object having a different shape

At this point in the experiment, the agent develops both, complete and partial, generalised schemas as the threshold for the generalisation process i.e., two similar schemas, has been achieved. Figure 4.6 illustrates the generalised schemas developed through the second object experience.

Figure 4.7 visualises the partial and complete generalised schema building in object shape versus colour dimensions.

Figure 4.7 shows that Dev-PSchema created partially generalised schemas when it experienced a second object with the same shape as the first object. However, it created a completely generalised schema when the second object is a different shape to those previously

Preconditions	Action	Postconditions
Observation=colour, value=$a, x=$b, y=$c Observation=shape, value=sphere, x=$b, y=$c Observation=colour, value=$a, x=$b, y=$c Observation=touch	**Grasp**	Observation=colour, value=$a, x=$b, y=$c Observation=shape, value=sphere, x=$b, y=$c Observation=colour, value=$a, x=$b, y=$c Observation=holding Observation=sound

Preconditions	Action	Postconditions
Observation=colour, value=$a, x=$b, y=$c Observation=shape, value=$d, x=$b, y=$c Observation=colour, value=$a, x=$b, y=$c Observation=touch	**Grasp**	Observation=colour, value=$a, x=$b, y=$c Observation=shape, value=$d, x=$b, y=$c Observation=colour, value=$a, x=$b, y=$c Observation=holding Observation=sound

Fig. 4.6 Partial (top) & complete (bottom) generalised schemas corresponding to generalisation from 2A & 2C and 2B & 2D respectively

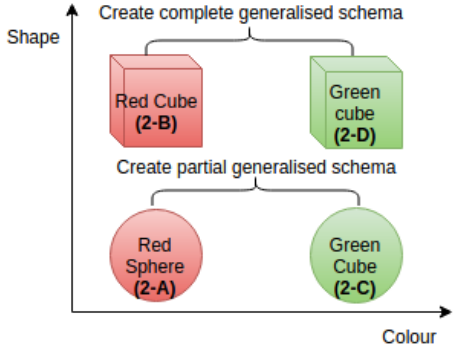


Fig. 4.7 Generalisation over shape versus colour

experienced.

4.2.1.4 Second Part: Adapting Generalisation

In the second part of the experiment, a third object from category 2 is introduced. This object, a blue cylinder, differs in both, shape and colour, and provides a different perceptual response, i.e., light, when grasped to the previously experienced objects. As the new object provides new perceptual information, the anticipated observation i.e., sound, from the generalised grasp schema, based on previous experiences, no longer matches. The mismatch, along with the inability of the system to observe matching schemas in terms of postconditions, causes the system to adapt the changes and build a new generalised schema. Figure 4.8 shows the new concrete schema created, after both the, partial and complete, generalised schemas fail to match.

Preconditions	Action	Postconditions
Observation=colour, value=blue, x=2.0, y=1.0 Observation=shape, value=cylinder, x=2.0, y=1.0 Observation=colour, value=hand, x=2.0, y=1.0 Observation=touch	**Grasp**	Observation=colour, value=blue, x=2.0, y=1.0 Observation=shape, value=cylinder, x=2.0, y=1.0 Observation=colour, value=hand, x=2.0, y=1.0 Observation=holding Observation=light

Fig. 4.8 Third object, different non-visual observation

This new schema also demonstrates the over-generalisation from the previous round. Although this new object was not previously experienced, the system is found to use a generalised grasp schema for it, in an attempt to obtain similar consequences related to its grasping. However, the outcome of the grasping led to different consequence to those predicted which indicates that the particular schema is overgeneralised, and a new schema is necessary. This resembles the accommodation process expressed by Piaget [147] and new knowledge is created when existing knowledge fails to solve the problem, as discussed in Chapter 1.

4.2.1.5 Third Part: Testing Generalisation

In this part, a blue sphere (object of category 1), and a red cylinder (object of category 2), are introduced at branches (C, D) and (A, B), respectively, according to the experiment's flow as depicted in Figure 4.2). Both objects are introduced after achieving complete or partial generalised schemas for category 1 and un-generalised schema for category 2 objects. While interacting with the new objects, the agent creates a partial generalised schema for the red cylinder. This is because the shape of this object allows the system to distinguish preconditions and postconditions from previously failed generalisations. It is found that no additional schemas are created for the blue sphere, as the agent is able to deal with the object using the partial or generalised schemas already acquired using previous experiences. At this stage, the agent undergoes accommodation and creates a new schema for the red cylinder and remains in equilibrium for the blue sphere, and thus creates no new schema.

4.2.2 Discussion on Experiment 1

This experiment demonstrates the ability to generalise experiences with the objects given in an environment using visual features of the objects and non-obvious properties obtained from interactions with them. The effectiveness of the proposed generalising tool, Dev-PSchema, is evaluated by analysing the generalised schemas, obtained during the experiment. In this experiment, a high-level of perceptual representation of objects, with a clear distinction between colour and shape, is used. In developmental psychology it is observed that infants are very sensitive and do respond to visual features of objects such as shape, colour, size and pattern [67, 86, 180, 16, 203, 187, 131, 22]. However, it is also observed that infants rely more on an object's shape than on its colour as far as recognition and generalisation are concerned. Bomba and Siqueland [22] report that infants at the age of 3–4 months, are capable of categorising objects by shape. Similarly, Tremoulet et al. [187] observed that 12 months old infants rely only on object shape for recognition, but consider colour as well for object individuation. These studies provide an evidence that infants rely more on object shape than their colour for object recognition, individuation and categorisation.

Infants have also been observed to rely on visual features of the object in generalising. Graham and Poulin-Dubois [67] found that 4-10 years old children rely on object shape to generalise verbal labelling for them. Reliance on shape for generalisation has also been observed in young infants. Baldwin et al. [13] found that 9 and 16 months old infants use the shape as a visual cue to generalise non-obvious properties associated with the objects. During the experiment, infants were provided objects that produced some sound when squeezed. After a 30 second experience, the infants were provided with a novel object having similar and non-similar shape and colour to the experienced object. Experimenters found that the infants performed similar behaviours with the novel objects that were similar in shape in order to obtain the previously experienced non-obvious property, irrespective of colour similarity. These results demonstrate infants reliance on shape over the colour of the objects for manual experiences to obtain non-obvious properties.

"Why is shape so important for such reasoning?". Graham and Poulin-Dubois [67] believe that shape is a perceivable and integral part of the object representation, which does not require extensive experience in terms of verbal representation to be gained. Wilcox [203] argues that infants consider shape features of an object relevant when attempting to predict the outcome of acting on them. Similarly, Nicholson and Humphrey [131] believe that although linked, both colour and shape information are encoded in the human brain, independently of each other, rather than as part of a single representation. Colour-based representations speed up the recognition process but shape-based have a stronger influence on it. These findings suggest that infants rely more on the shape-related features than the surface-related ones (e.g. colour and texture) for object recognition or differentiation and categorisation. The findings also suggest that two perceptions (shape-and surface-related) are processed separately. Differentiation and recognition also help infants to generalise about objects. Thus, designing a system with separate representations for object shapes and colours is supported by developmental psychology.

The generalised schemas obtained in this experiment demonstrate the increased reliance on shape of the object rather than the colour for generalisation. Although tags for shape and colour are provided by the low-level system (Sandbox here), the agent utilised this information to create partially and complete generalised schemas to identify which property of the objects is more important than the others to produce the particular outcome in the environment. The agent created partially generalised schemas when it experienced objects that had similarity in shape and completely generalised schemas when it experienced objects that differed in shape irrespective of colour. Thus this experiment demonstrates the capability of the system to develop different levels of generalisation (un-generalised, partial and complete generalised) through experiences. The generalised schemas also help to predict the outcome of the action by instantiating it with the perceived sensory state.

4.2.3 Experiment 2: Predicting Object Movements through Generalisation

To evaluate the functional generalisation in Dev-PSchema we performed an experiment with visual activity in a simulated environment. We used the same 5×5 Sandbox environment discussed in Section 4.2.1, and limited the available actions to just the "Saccade" action to fixate on any of these positions. The experiment begins with bootstrapping to build saccade schemas to each of the visual positions of the environment.

To investigate the anticipation for visual events in the system, we introduce an object in the environment which moves one position to the right when fixated. Following the bootstrapping, the object is introduced that triggers, through the excitation mechanism, the agent to fixate at that position. The excitation mechanism finds the bootstrap schema to fixate at the object position by getting higher similarity between the perceived state and the schema, see Section 5.1 for details. Once the object is fixated and the agent learns a new schema, the environment is reset by removing the object.

This process is repeated with another object having different visual representations i.e., colour and shape, to obtain a generalised schema. The schemas with specific object representations (concrete schemas) and generalised schemas will be used to evaluate the performance of Dev-PSchema in terms of finding the functions between the variables. To test the predictions from the generalised schema we introduced a novel object in the environment to instantiate the generalised schema.

4.2.4 Results

The first object, referred to as object 1, at a particular position in space reminds the system about previously fixating at that position during the bootstrap process, resulting in the bootstrap schema to fixate on that position being highly excited. The system executes the

most excited schema i.e., saccade to the object position which results in a new schema being added to the memory describing the differences in pre and post conditions from the executed bootstrap schema. Figure 4.9 shows the process of obtaining a new saccade schema following the fixation on the first object.

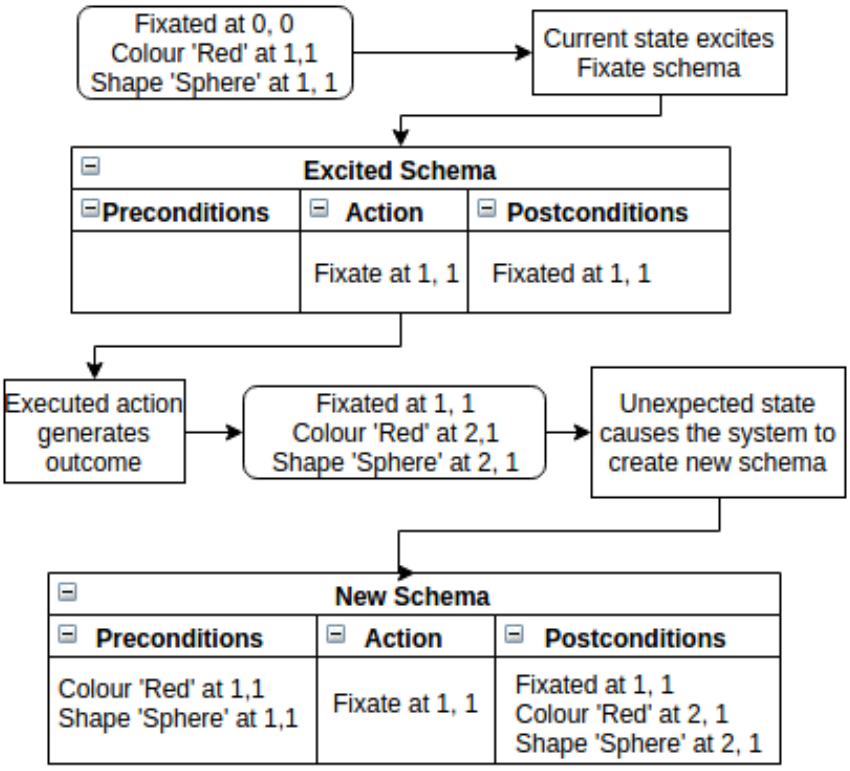


Fig. 4.9 Schema created while fixating on the first object

After fixating the object and developing a schema with the obtained perceptions the first object was removed from the environment and a second object, object 2, which possessed the same movement property, but was of a different shape and colour, was introduced at a different position. Following the same process, as shown in Figure 4.9, a new schema is created for fixating on object 2 with the concrete details associated with this experience. Two concrete examples of the saccade schemas trigger the generalisation process, resulting in a new generalised schema. Figure 4.10 shows the generalised schema and schemas used to

create it.

Preconditions	Action	Postconditions
Observation=fixated, x=0.0, y=0.0 Observation=colour, value=red, x=1.0, y=1.0 Observation=shape, value=sphere, x=1.0, y=1.0	**Fixate X=1.0, y=1.0**	Observation=fixated, x=1.0, y=1.0 Observation=colour, value=red, x=2.0, y=1.0 Observation=shape, value=sphere, x=2.0, y=1.0

Preconditions	Action	Postconditions
Observation=fixated, x=1.0, y=1.0 Observation=colour, value=green, x=2.0, y=3.0 Observation=shape, value=cube, x=2.0, y=3.0	**Fixate X=2.0, y=3.0**	Observation=fixated, x=2.0, y=3.0 Observation=colour, value=green, x=3.0, y=3.0 Observation=shape, value=cube, x=3.0, y=3.0

Preconditions	Action	Postconditions
Observation=fixated, x=$a, y=$b Observation=colour, value=$c, x=$d, y=$e Observation=shape, value=$f, x=$d, y=$e	**Fixate X=$d, y=$e**	Observation=fixated, x=$d, y=$e Observation=colour, value=$c, x=$d+1, y=$e Observation=shape, value=$f, x=$d+1, y=$e

Fig. 4.10 Schema for object 1 (top), object 2 (middle) and generalised schema (bottom)

Variables in both schemas for object 1 and 2 have the same action i.e., fixate to object position, but have different values for the visual features. In the obtained schemas the visual position in the environment is represented by numerical values hence considered for functional generalisation. With the activated functional generalisation flag, given in Algorithm 10, the generalisation mechanism recognises the matching change in the values and is able to apply this as a function on the positional values in the new generalised schema. Figure 4.10 shows that concrete schemas created through visual interaction with the objects contain a change in the position of the object when it is fixated. These changes are the same in both examples hence the generalisation process finds a matching relationship between the values of the preconditions and the postconditions of the given property i.e., the visual position here. The generalised schema shows the postcondition value of x coordinate is a transition “$b+1” as compared to original coordinate in the preconditions “$b”.

To evaluate the functional generalisation, a novel object was introduced in the environment. The novel object excites the new generalised schema, which is instantiated from the perceived sensory state. Following the instantiation process, the generalised schema predicts the outcomes of the schema. Figure 4.11 shows the perceived sensory state (left) and instantiated generalised schema from that state (right).

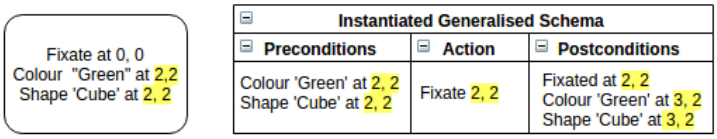


Fig. 4.11 State for the 3rd (left) object and instantiated generalised schema (right)

Figure 4.11 shows that the post-condition of the instantiated generalised schema predict that the object will move across one position from its current position following a fixation on it. Thus, the system shows the capability of anticipating the future state by making inferences from the current state using previous experiences. This prediction may fail during further exploration, which will result in failure of the generalised schema. This failure will affect the excitation of the generalised schema making it less excited, and therefore less likely to be selected again in the future.

4.2.5 Discussion on Experiment 2

This experiment demonstrates the ability of Dev-PSchema to anticipate the outcome of actions through generalisation, based on previous experiences. Dev-PSchema uses generalised schemas to apply the learning in novel situations, the third object introduced in the environment in this experiment. This result is consistent with the behaviours in infants. Adler and Haith [2] found that 3 months old infants are able to anticipate visual content and the visual position of an object after experiencing a similar event.

Dev-PSchema is also able to find the linear mathematical relationship between the numerical property variables represented in a generalised schema. This capability may be considered as the "causal anticipation" as the system finds the causal relationship between the action and its outcome through changes in numerical values. Infants have also been observed to build an understanding of causal relationships between their actions and changes in the environment. Haith et al. [72] found that 3.5 months old infants can develop anticipation for

visual events after very short experiences and develop expectations rapidly even though they have no control over the event.

In conclusion, this experiment demonstrates the capability of the system to develop functional relationships between numerical properties common to the precondition and post-condition. The functional relationship helps to anticipate the outcome of a visual event having similar visual contents to those previously experienced in a given environment.

4.3 Discussions and Conclusions

Visual anticipation is seen as a very important step in developing knowledge about objects and events in a given environment. To interact with a static or dynamic environment we predict the movement of the environment and changes caused in it by any action performed on it. In a dynamic environment we adjust our movement to achieve an anticipated target goal and in a static environment the action effect is predicted before performing it [199, 118]. Anticipation has been observed in young infants as well, developed through experiences and generalising those experiences. Generalisation helps to develop a set of general concepts from experiences and use those general concepts to anticipate the outcome of an action in a similar situation/environment [181].

In these experiments, Dev-PSchema created new schemas when none of the previous schema matched the associated action's outcome, in terms of postconditions. Schemas with similar actions are processed further to build generalised schemas, as object-action concept, leading to similar outcomes in similar situations. This behaviour is seen in humans as well [67, 203, 131]. The process of building schemas and generalising in this work draws inspiration from Piaget's theory, see Section 1.4.1. The system builds new schemas once the outcome of an action differs from the anticipated outcome, similar to the accommodation process in Cognitive developmental theory [145]. This expected outcome leads the agent

to develop a new schema, hence providing opportunities to explore further. Shepard [174] believes that generalisation is the result of experiencing situations that have similar consequences in an environment. Coupled with this, it is reported that differences in situations help infants to build new knowledge. Stahl and Feigenson [181] believe that learning is associated with the unexpected outcomes, leading to further explorations. In experiment 1 of this Chapter we observed that Dev-PSchema created a new schema when the outcome of the executed generalised schema fails to meet the actual outcome in part 2 of the experiment. The new schema associates the new outcome with the action. Although the failed generalised schema represents a concept true for a few examples used to build it, it failed to build a representation, in the generalised schema, true for a broader range of situations. Such generalisation is called over-generalisation and can be seen in infants. Infants have been observed to overgeneralise lexical information [67, 61] as well as visual information [203, 154]. The infants, in these studies, have been observed to use previously learnt words and actions for the new objects/situations that are not suitable. These examples show that the infants over-generalise their experiences and apply the same knowledge on irrelevant objects/situations. This leads to the construction of new knowledge, through either building a new level of generalisation or developing the exceptions for the generalised concepts.

The results show that Dev-PSchema is able to construct knowledge using perceptual information that is obtained while acting on the objects. The knowledge, initially reflecting a particular task, is further developed to demonstrate a general concept (e.g., concrete grasps to a generalised grasp). The experiments demonstrated the use of Dev-PSchema for schema generalisation, based on perceptions obtained while acting on the objects and observing repeated perceptual information i.e., sensory state, associated with them. This ability to generalise helps the system to further utilise learnt skills for new situations that happen to share similar perceptual features with those previously experienced. Also, the system demonstrated a way to tackle the issue of over-generalisation, as it is able to create new variations of knowledge and deal with specific types of objects with distinguishing features.

With the addition of functional generalisation in the system, it is also capable of finding linear relationships between the numerical values of the sensory perceptions. This helps to anticipate the visual features of the environment following an applied action. Finn and Levine [51] and Agrawal et al. [4] have demonstrated similar visual anticipating systems, where the agents predict object position following an action applied to it. In contrast to Dev-PSchema, where anticipation is obtained by generalised schemas developed through active experiences, the systems [51, 4] are trained through neural networks. Thus they need large data sets or a large number of experiences.

Although the current system (in Dev-PSchema) is only able to find the additive translation (+/-) between the properties present in the action and states i.e., pre-conditions and post-conditions, it demonstrates the capability of Dev-PSchema for building such relationships which can be extended further in the future.

In conclusion, the experiments demonstrated the capability of generalisation to represent basic object-action concepts in Dev-PSchema. The generalised schemas not only help to extend the behaviour for use in new situations but also helps to anticipate changes caused by the action presented in generalised schemas.

Chapter 5

Simulated Infants and Play Behaviour

Dev-PSchema uses an excitation mechanism to generate play behaviours, modelled on the behaviours observed in infants. This chapter describes the parameters of the excitation mechanism/system and the infants' play characteristics relevant to the modelled parameters.

Piaget's developmental cognitive theory proposes staged learning in humans, gaining new knowledge by building on existing knowledge. According to the theory, children develop different cognitive skills at different ages [145], as discussed in Chapter 1. The first stage in his proposed developmental theory, referred to as the sensorimotor stage, is about learning through ego-centric sensorimotor experiences. Such experiences, gained through actions performed and related sensory perceptions, help to build early stage knowledge about the performed actions and the related objects, see Section 1.4.1.

During infancy, infants spend most of their awake time in playing. Through the play they interact with their environment and surrounding objects. They build and develop their knowledge through exploring their own actions, and learning the resulting effects on the objects in their environment. That is the reason play behaviour is seen as an important factor in cognitive development [153, 132]. In addition to learning and understanding the environment, the play provides a foundation for academic and social learning [73].

Infants appear to be very interested in their surrounding environment and tend to perform a wide variety of free play activities in order to explore it. Their actions are not constrained by any predefined rules other than those that are related to their physical capabilities. Nevertheless, physical constraints do help them to scaffold learning, as the infants gradually understand the different elements which are related to the behaviours they perform. In addition to the exploratory play, infants demonstrate exploitation behaviours during play. They perform similar play behaviours repetitively to reproduce the interesting effects they had with their previous experiences in similar situations. Furthermore, they explore their environment and objects in it, extending their learning into novel and similar situations through a process of generalisation [13, 201].

As infants' cognition develops their play behaviours develop with it, starting from exploratory play through to practice play, then developing further in pretend play and finally to rule based play. The type of play behaviour seen in early infancy is called exploratory. Exploratory play has a few core characteristics to identify as they include being pleasurable and enjoyable, with no extrinsic goal, being actively engaged and non-literal characteristics [73]. It can be concluded that play behaviour provides the fundamental motor and cognitive capabilities for humans to grow and survive in their environment.

To understand the play behaviours, we need to understand the causes that generate such behaviours. In Section 1.3 we discussed that the intrinsic motivations are seen as the reason for exploratory play behaviours observed in the early infancy which are triggered by novelty, change and ambiguity [116]. However, continuous exposure of the similar situations or environments make infants habituated to it [177]. Thus novelty, change (in the environment or objects) and ambiguity do provide motivation for exploratory behaviours in infants, however these motivations decrease as the explorations end with either no change or, repetitive or non-interesting effect in the environment. Thus infants tend to explore and seek out the novel opportunities for learning through their play behaviours, however sometimes they maintain

their focus on familiar objects [57, 59, 81].

A learning model for artificial agents, inspired from infant development, should develop knowledge through exploratory play behaviours as humans do in their early infancy. The Dev-PSchema system provides an open-ended learning mechanism through exploratory behaviours. The system uses an excitation mechanism to select interesting actions to perform on objects perceived in the environment, then learns the outcomes related to the different actions. The objects in the environment are defined with the perceptions containing underlying properties, see Chapter 3 for details. The action selection (i.e., excitation) mechanism depends on the object perception and the schema statistics for finding suitable actions to perform in the environment. The object perception statistics include the number of times the object appeared in the environment $C(O_e)$ and the number of times used in the schemas $C(O_s)$. Whereas the schema statistics include success rate S_r and the time steps where the schema was executed T_S.

Apart from generating exploratory play behaviour for a perceived state in the environment, the excitation mechanism in Dev-PSchema enables it to vary its action selection, hence generate different exploration path by tuning weights for the excitation parameters. Similarity, novelty and habituation are the three main parameters to control behaviours generated with Dev-PSchema. It should be noted that these parameters are defined slightly differently to those used in developmental psychology. They are discussed in detail later in this chapter in Section 5.1. The statistics associated with the perceived sensory information and applied actions also affect the action selection mechanism, hence behaviours produced.

In this Chapter, we demonstrate the capability of Dev-PSchema to generate variations in behaviours by tuning the weights applied to the excitation parameters. We also demonstrate how the past experiences and behaviours affect the action selection in the future. We evaluate this capability of the system through two experiments. In one experiment we demonstrate variations in the behaviours by changing weights for the similarity, novelty and habituation parameters. In the second experiment, we demonstrate variations in the behaviours through

changing weights for the schema statistics and a perceived sensory state i.e., world state (WS). The experiments are performed in the *Sandbox* simulator, utilising reach, squeeze and press actions (see Chapter 3 for details) on various objects, represented in the environment.

In Section 5.1 we discuss the excitation mechanism of the system and its underlying processes. In Section 5.2 we provided details about the experiments performed and the obtained results. Finally, in Section 5.3, we provide a conclusion based on the obtained experimental results.

It should be noted that this Chapter includes parts from the published peer reviewed paper, given below:

- Kumar S., Shaw P., Giagkos A., Braud R., Lee M.H., Shen Q. Developing hierarchical schemas and building schema chains through practice play behaviour. Frontiers in Neurorobotics. 2018;12:33.

5.1 Excitation Calculator

A Dev-PSchema enabled agent calculates excitation of each schema, hence action, in the memory by comparing the perceived world state (WS), with the postconditions in each schema. The excitation is based on the similarity, novelty and habituation for the perceived world state. A combination of these weighted factors is further combined with a weighted calculation based on the schema statistics. Thus the overall excitation represents the agent's action selection based on the similarity, schema statistics and the novelty for exploration, through the novelty and habituation pair. Varying such weights, will allow the agent to demonstrate variations in behaviour selection corresponding to the different simulated individual infants with different preferences in a given environment. Thus a higher weighting for similarity will lead the agent to demonstrate more predictable behaviours i.e., show preferences towards similar actions and objects. Whereas with higher novelty/habituation

weights, it will demonstrate more exploratory behaviours and preferences for novel objects.

The agent calculates the excitation of all schemas in the memory for a perceived state of the environment and selects the schema that has the highest excitation. The excitation calculation begins with finding the similarity between the perceived objects and the objects i.e., observations, present in the postconditions of the schema. The similarity calculation is followed by the similarity, novelty and habituation calculations of the perceived objects that are calculated through incorporating their statistics. The agent further applies the schema excitation using its statistics. Algorithm 17 describes the mechanism for calculating the excitation for a perceived state of the environment i.e., world state (WS).

Algorithm 17 finds sequences of actions, described to as schema chains, before calculating excitation for each schema in the memory if possible. The chaining mechanism is discussed in details in Chapter 6. The algorithm, later, calculates the excitation for all the schemas and the chains in the memory. Either a schema or a chain with the highest excitation is returned for execution in the environment. Schema chains and their excitations are further discussed in details in Chapter 6.

When calculating excitation for each schema in the memory, the mechanism finds the similarity between the schema postconditions and the perceived world state. The mechanism calculates the highest match between each observation in the perceived world state and the postconditions of the schema. The equation to calculate the similarity is discussed in Section 5.1.1. The novelty and habituation of each observation present in the perceived world state are calculated using the equations discussed in Section 5.1.2 and 5.1.3 respectively. The combination of the novelty and the habituation of an observation is also referred to as the *observation excitation*.

A weighted combination of similarity and, novelty and habituation is further applied to the calculated excitation value by considering previous experiences of the schema. This com-

Algorithm 17 Excitation Calculation

```
1: procedure get_most_excited_action (WorldState WS, Boolean chain_encouraged)
2:     schema_excitations = new list of pairs
3:     Chains = empty list; max_schema_excitation = 0.0; most_excited_schema = None
4:     for each schema S in Memory do
5:         Chain = First chain in find_path(WS, S_postconditions)        ▷ See Alg. 18 for details
6:         Add Chain to Memory Chains, if any
7:         for each observation O1 in WS do
8:             max_sim = 0.0
9:             excitations = empty list
10:            for each observation O2 in Schema S_postconditions do
11:                if similarity (O1, O2) > max_sim then
12:                    max_sim = similarity (O1, O2)
13:                end if
14:            end for
15:            φ = ω1 × max_sim + ω2 × [Novelty(O1) − Habituation(O1)]
16:            Add φ to Excitations
17:        end for
18:        overall_exciatation = ω3 × Avg(excitations) + ω4 × λ        ▷ For λ see Eq. 5.7
19:        Add pair (key S, value overall_exciatation) to list schema_excitations
20:        if max_schema_excitation < overall_exciatation then
21:            max_schema_excitation = overall_exciatation
22:            consider S as most_excited_schema
23:        end if
24:    end for
25:                                                  ▷ Calculating chain excitations
26:    max_chain_excitation = 0.0
27:    most_excited_chain = None
28:    for each chain C in Chains from Memory do
29:        chain_excitation = calculate_chain_excitation (C)
30:        if max_chain_excitation < chain_excitation then
31:            max_chain_excitation = chain_excitation
32:            consider C as most_excited_chain
33:        end if
34:    end for
35:    if max_chain_excitation < max_schema_excitation then
36:        return most_excited_schema
37:    else
38:        return most_excited_chain
39:    end if
40: end procedure
```

bination represents an excitation for the perceived object. Schema statistics i.e., activations and successes, are used to calculate the schema excitation. A weighted combination of an objects' excitation and a given schema excitation represents the overall excitation for the schema. This combination is discussed in detail in Section 5.1.4. Changing in weights for both combinations enables the agent to demonstrate variations in behaviours.

5.1.1 Similarity

The similarity is calculated by matching the degree of similarity between the perceived objects (in world state) with those present in the postconditions of previously learnt schemas. The mechanism matches individual properties present in a schema's postconditions and perceived objects' perceptions such as colour or shape. Although the algorithm calculates matches between one perception of the perceived object and all the perceptions present in the postconditions individually, it only considers the highest match between the two perceptions. The match returns a value between $0 \sim 1$, where 1 indicates the exact match. Equation 5.1 is used for the calculation of similarity:

$$Similarity = \frac{\sum_{i=1}^{C(\rho)} \max_{1 \leq j \leq C(\zeta)} [Sim(\rho_i,\ \zeta_j)]}{C(\rho)} \tag{5.1}$$

Function *Sim* in Equation 5.1 returns the similarity between the i^{th} property of the object's perception ρ, that is ρ_i, and the j^{th} property of the schema's object perception (ζ_j). $C(\rho)$ represents the count of the number of properties in the perceived object and $C(\zeta)$ is the count of the number of properties found in a schema object perception. If a property appears in both states but the values are different, then *Sim* will return a partial match, i.e., 0.5. However, in cases of numerical parameters, a match between $0 \sim 1$ will be returned. The final match value is normalised by taking a ratio between the sum of perception matches and the total number of properties in the perceived object. Figure 5.1 show an example of a perceived world state and the schema postconditions used for similarity calculation in Equation 5.1.

5.1.2 Novelty

The novelty of the perceived object is calculated by considering how frequently perceptions, that describe an object, are appeared in postconditions of the schemas in the memory, in

World State
$C(\rho)_1 : \rho_1, \ldots, \rho_r$
$C(\rho)_2 : \rho_1, \ldots, \rho_s$
$\ldots$
$C(\rho) : \rho_1, \ldots, \rho_t$

Preconditions	Action	Postconditions
$\ldots$		$C_1(\zeta) : \zeta_1, \ldots, \zeta_o$
$\ldots$	$\ldots$	$C_2(\zeta) : \zeta_1, \ldots, \zeta_p$
$\ldots$		$\ldots$
$\ldots$		$C(\zeta) : \zeta_1, \ldots, \zeta_q$

Fig. 5.1 Left: Perceptions in the current world state. Right: A schema postconditions containing $C(\zeta)$ number of perceptions, where each perception ζ contains j number of the properties

connection to the running time-steps as shown in Equation 5.2.

$$\tau_1 = \frac{C(O_s)}{C(O_e)} \tag{5.2}$$

where $C(O_s)$ represents the number of times the object perception O appeared in schemas and $C(O_e)$ represents the number of times O it was perceived in the environment. τ_1 calculated in Equation 5.2, represents the ratio between the total number of times an object's perception is used in a schema postcondition against the total number of times it is observed (either before or after an action is performed). Higher τ_1 indicates that the object perception has been used in the schema postconditions for most of the time it had appeared in the environment. Equation 5.3 describes the calculation for the novelty.

$$Novelty = (1 + cos(4.75\tau_1))/2 \tag{5.3}$$

The novelty equation is designed to express a smooth curve for values between 0 and 1 for τ_1, as shown on the right in Figure 5.2. The cosine is scaled between 0 and 1, with the period reduced such that at $\tau_1 = 1.0$ the value is 50%. The scaling coefficient i.e., 4.75, is used for reducing the cosine period from 2π to approximately 75° to keep its value in

positive coordinates of the graph for all the values. Thus Equation 5.3 provides positive values between 0 and 1 for any input value of τ_1.

The novelty of the perceived object transitions from the maximum to the minimum and then back up to the 50% over the values of τ_1 from $0 \rightarrow 1$. Initially, the novelty of the newly perceived object will be the maximum. As the object appears more in schemas or is played with more frequently its novelty reduces. If the object is not played with for a period of time (i.e., not included in new schemas), its novelty gradually increases.

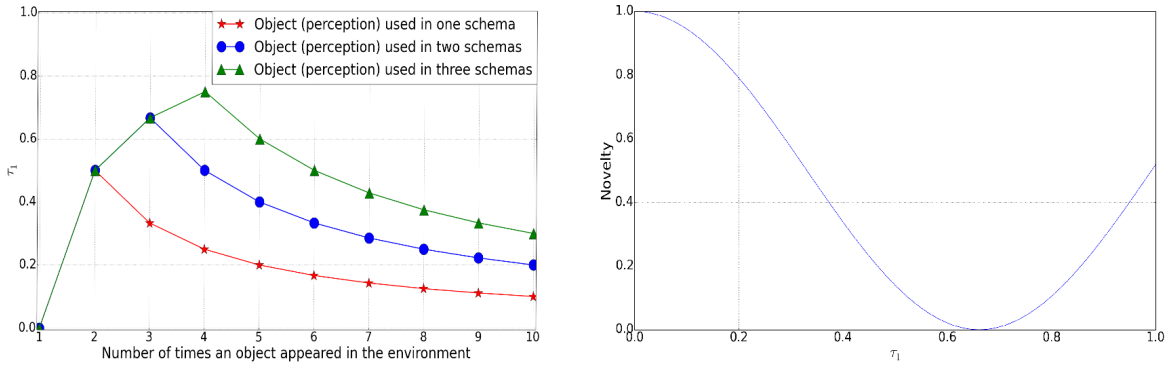


Fig. 5.2 Left: Value of τ_1 for an object perception used in 1, 2 or 3 schemas continuously against the number of times it appeared in the environment. Right: Novelty of an object perception over the range of value for τ_1

The graph on the left-hand side (LHS) in Figure 5.2 shows the value of τ_1 for the objects used in 1, 2 and 3 schemas as 0.5, 0.667 and 0.75 respectively when appearing continuously in the environment. The value of τ_1 starts reducing as the object perception is not used further in schemas. Thus, the agent will be interested in the object after a while as its novelty increases. However, τ_1 increases to 0.5 for an object which is used in one schema as it appears in the environment. This will help to maintain the novelty of the object, to some level, which is used only once in a schema, providing an opportunity to explore it further. After that, the value of τ_1 will be increased to 0.67 if the object is used twice as it appears three times in the environment, as shown on the LHS of the graph in Figure 5.2. Therefore, the novelty of such an object will be 0. However, if an object is continuously

used in schemas, while it appears in the environment continuously, its τ_1 and the novelty will be increasing. For example, if an object is used three times in the schema while it appeared four times in the environment, its τ_1 will be 0.75, hence the novelty of 0.0436. Thus, the novelty of an object increases towards the value of 0.5 as it is continuously used in the schemas. Therefore, another parameter, "habituation", is introduced for calculating the object excitation in combination with the novelty. The habituation parameter, described in Section 5.1.3, starts affecting the object excitation once it is used at least twice in the schemas.

5.1.3 Habituation

Habituation, in combination with the novelty, will enable the agent to draw its attention from an object, that has been interacted with recently and continuously, towards a different object that has not been interacted with at all or for a while. Habituation (for an object) depends on how recently schemas that contain the object perceptions are used (executed) in the environment. The agent is expected to be more habituated, hence less interested, with an object/situation that reoccurs after interacting with the environment. This is inspired by developmental psychology, where infants become habituated with objects or events after a period of exploration or observation [77, 175, 39, 89]. Habituation at a given time-step is given by Equation 5.4.

$$\tau_2 = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \frac{T_{S_i}}{T_c}, \ if \ n > 0 \\ 0.0 \end{cases} \tag{5.4}$$

where n is the total number of schemas that contain the object perceptions and have been executed at least twice, T_s is the time step when such a schema S was last executed and T_c is the current time step. If schemas containing the object perception have not been executed more than twice or the object perception never appeared in a schema(s) then $\tau_2 = 0$ and habituation for the perceived object remains 0. This keeps the agent's interest in objects

which either have not been explored much or never been interacted with. Also, notice that τ_2 is used to calculate the habituation over a period of time steps. Thus, the value of τ_2 increases as a schema containing the object perceptions was executed recently, as shown in Figure 5.3. Conversely, on the right-hand side of Figure 5.3, τ_2 decreases when the object perceptions do not occur for a period of time steps or a schema(s) containing the object perception has not been used for a long time. Equation 5.5 presents the formula for the habituation.

$$Habituation = 1.0 - e^{(-5\tau_2)} \tag{5.5}$$

Similarly to novelty, the coefficient of the exponential is designed to smooth the curve for the range 0–1 (as shown on the right-hand side of Figure 5.3). The coefficient (i.e. -5) also helps to increase the habituation for an object rapidly for smaller values of τ_2 and decrease it slowly, as can be seen on left-hand side of Figure 5.3. Habituation is expected to increase during frequent interactions with the environment that lead to the same object perceptions being captured, which in turn allows the agent to select actions that promote interactions with different areas of the environment.

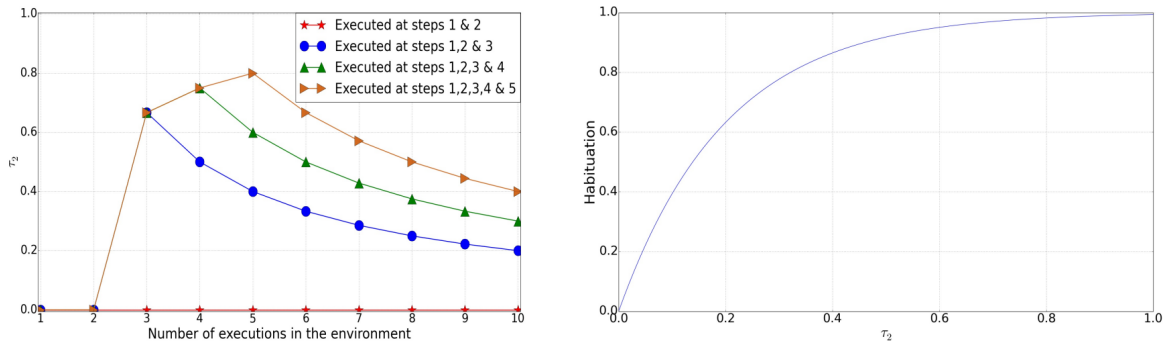


Fig. 5.3 Left: Value of τ_2 for an object perception in schemas used in execution steps [1 & 2], [1, 2 & 3], [1, 2, 3 & 4] and [1, 2, 3, 4 & 5] against the execution steps. Right: Habituation of an object (perception) over the range of values for τ_2

The graph on the left in Figure 5.3 shows τ_2 for the perceived objects against the number of an action executions performed in the environment. τ_2 for a perceived object increases as the schema, in which it is used, is executed in recent time-steps. However, τ_2, hence habituation, starts reducing as the schemas are no longer executed. A similar effect is also observed in infants. Infants show a decrease in their interest as they interact with any objects or observe the environment for a longer period of time. However, this interest is restored as when the object reappears after a certain interval following the first interaction with it. The graph on the right in Figure 5.3 shows that habituation is directly proportional to τ_2, hence the schema containing the object perception execution.

In developmental psychology, "habituation" is defined as a decrement in a response for a repeated stimulation [155]. Thus a stimulation is said to be novel, when compared to the previously experienced stimuli, if it causes a change in the response. Thus these two terms, habituation and novelty, are related to each other. The excitation mechanism here is modelled on the habituation paradigm in developmental psychology, thus novelty and habituation are considered as a pair. However, both, novelty and habituation, are calculated separately with different parameters hence defined separately.

5.1.4 Total excitation

The total excitation represents excitation for the perceived objects, calculated by combining the similarity, novelty and habituation, as shown in Equation 5.6. This allows the agent to select an appropriate object to interact with, by utilising previous experiences associated with all objects in the environment. The excitation of an individual object, based on its perceptions in the environment, is represented by ϕ in the system.

$$\phi = \omega_1 \times Similarity + \omega_2 \times (Novelty - Habituation) \tag{5.6}$$

where

$$\omega_1 + \omega_2 = 1 \quad \& \quad 0 \leq \omega_1,\ \omega_2 \leq 1$$

5.1.4.1 Object excitation

In Equation 5.6, novelty and habituation are combined as they are both related to experiences associated with the currently perceived objects, whereas the similarity considers all experienced perceptions of the objects which the system has previously interacted with. The combination of novelty and habituation is weighted with ω_2, whereas similarity is weighted with ω_1. Both weights sum to 1, thereby a proportion is allocated to each component. By varying the weights, we can simulate different artificial infants with different preferences (e.g., novel versus favourite toy) in a given situation of the environment. Applying a higher weight to ω_1 will make the agent more likely to interact with similar objects which have a higher degree of similarity to previously interacted objects. Whereas with higher values of ω_2, the agent will be more likely to interact with the novel or less familiar objects. This can also be seen as a preference towards either exploration or exploitation.

In exploration, the agent will tend towards interacting with different objects following a few interactions with one. If the environment contains a static set of objects, the agent will tend to interact with one after another, cycling back to the first one. In exploitation mode, the agent will tend to interact with more familiar objects rather than novel and less habituated objects. The exploitation mode simulates an infant with a strong preference towards interactions involving the familiar objects.

5.1.4.2 Schema excitation

Alongside the object-related excitation, the agent calculates the excitation of each schema in the system, in order to select the appropriate schema to be executed. Thus, this excitation is related to the possible actions that could be performed for the each object, rather than the

object perception alone. Equation 5.7 gives the calculation of the schema excitation λ.

If the perception(s) in the environment following an action matches the postconditions of the schema, the execution is considered to be successful. A success rate S_r is maintained to record the proportion of time that the expected outcome of a given schema has been achieved. This can also be considered as a reliability measure for each schema.

$$\lambda = S_r \times e^{-1.1\frac{T_s}{T_c}} \tag{5.7}$$

In the equation above, T_s is the last time step on which a particular schema was executed and T_c is the current time step. A coefficient to the exponential power is used as a smoothing factor to obtain an exponential response over the values of the ratio between schema executions and the current time. The coefficient (-1.1) gives λ value about 0.33 (considering S_r to be 1.0) when the schema was executed in the last time step (T_c equal to T_s). This provides some excitation to the schema even if it was executed in the last time step, as the schema may be producing interesting results. For example, squeezing an object that produces sound may be an interesting action for the agent and it may want to repeat it to get the same effect.

5.1.4.3 Combined excitation

Ultimately, the excitation for each schema is calculated by considering each object that is present in the environment. Equation 5.8 gives the final excitation of a schema combined with all object excitations.

$$Excitation = (\omega_3 \times \frac{\sum_{i=1}^{m} \phi_i}{m}) + (\omega_4 \times \lambda) \tag{5.8}$$

Where:

$$\omega_3 + \omega_4 = 1 \quad \& \quad 0 \leq \omega_3,\ \omega_4 \leq 1$$

where m is the number of all the perceived objects, ϕ_i is the excitation of the ith object and λ is the particular schema's excitation. Notice that due to Equation 5.7, a schema that is being executed repeatedly results in a lower excitation value for λ, which in turn contributes less to the final excitation. In a similar vein, schemas that are never used become more excited than their recently executed counterparts. This enables the agent to explore the environment by performing different actions. The weights of the average object and schema excitation are defined by ω_3 and ω_4 respectively.

The schema with the highest excitation competes with the most excited chain. Algorithm 17 returns either the schema or chain with the highest excitation for execution. The algorithm describing the mechanism for the chain excitation and chain execution is given in Chapter 6. The schema execution mechanism for the most excited schema is given in Algorithm 2.

5.2 Experiment and Results

To demonstrate and evaluate the excitation mechanism in Dev-PSchema we performed two experiments. The first experiment demonstrates the effect of variations in object excitation weights, ω_1 & ω_2, on preferences for the object to interact. The second experiment demonstrates the effect of variations in schema weights, ω_3 & ω_4, on the choice of the action in a given environment. Both experiments demonstrate the capability of the system to simulate individual infants, having varying preferences and behaviours provided with the same object(s), in the same situation.

The experiments are performed in the *Sandbox* simulator, discussed in Chapters 3 and 4. The simulator contains a manipulator/hand to perform different actions in the environment, and simulator provides high-level descriptions for the objects in the environment along with

proprioceptive information i.e., touch and hand grip. The objects in the environment are represented through visual perceptions, especially shape and colour. The agent performs high-level actions in the environment selected by the excitation mechanism.

Although the set of actions used in the experiments are limited, they are sufficient to demonstrate the playful capabilities of Dev-PSchema enabled agent similar to an infant at play. The agent is provided with an initial set of predefined bootstrap actions. In a developmental sensorimotor system, these actions may be learnt through motor babbling, as discussed in Chapter 3.

5.2.1 Experiment 1: Novel vs Familiar Preference

This experiment is inspired by the study of "Young children's preference for unique or owned objects" by Gelman and Davidson [59]. The study investigates the infants' preference for a well-known object (a favourite toy) rather than a new identical object or a novel, non-identical object. In the study, most of the time infants tend to select their own, well known, objects when they are given a choice of two. Interestingly, the infants are found to select the identical or novel object when they are asked to select an object for the experimenter.

To replicate the behaviour of infants in the experiment only the reach schema, hence action, is used. The agent's preference is expected to be demonstrated by utilising several reach related schemas that are gradually learnt by interacting with the objects in the environment.

The experiment starts with a single object, a red cube, presented to the agent. With the single reach schema in memory, the agent is most excited to interact with the object by reaching towards it. Once reaching is performed successfully, we reset the environment and return the hand to its initial position. The experiment is divided into two stages. The first stage is for familiarisation, that is the agent reaches for the same object three times, to decrease the object's novelty and increase its habituation. The second stage is the test

condition, where both, the familiar and a novel, objects are presented to the agent. This stage is further divided into four parts for each of the novel objects introduced. For each object combination, the weightings for similarity, novelty and habituation, ω_1 and ω_2, are varied to show the change in preference. Notice that ω_3 and ω_4 remain 0.6 and 0.4, respectively, in all the variations of the experiment, to keep the focus on object excitation (Equation 5.6) rather than the schema excitation (Equation 5.7). A slight weighting is given to the value of ω_3 over ω_4 to keep excitation dependence on the similarity and habituation/novelty rather than schema statistics i.e., activations and successes.

5.2.2 Results: Experiment 1

Following the familiarisation stage, along with the original object (i.e., the red cube), we introduce four different objects one by one. Each of the new objects either contains at least one common property i.e., shape or colour, to the red cube or contains no matching property. A blue cube, a red ball, a red cube and a blue ball are used, following the familiarisation. Each of the new objects is introduced after the familiarisation stage, without any more experience being acquired. We expect that the agent will prefer to reach for the novel object when it is introduced after a few experiences with the previously introduced object, at equal weights for the similarity and novelty/habituation pair. However, by changing the parameter values, we expect the agent will reach for the familiar object rather than the novel one. The initial weight for similarity, ω_1, and novelty and habituation, $\omega 2$, are set to 0.5, then the weight ω_1 is increased in steps of 0.1, whilst maintaining $\omega_1 + \omega_2 = 1$ (Equation 5.6), until the observed behaviour flips towards the familiar object.

In this experiment, only one schema (reach) is used. Thus the object used in the environment was only added in the one reach schema and τ_1 values for all the cases of this experiment were recorded as 0, 0.5, 0.33 and 0.25 in four execution steps, from 1 to 4 respectively. Below is a discussion of the observed behaviour of the agent following the initial experience and perceiving the novel object over different values of the weights for

excitation parameters.

5.2.2.1 Novel object with matching properties (same colour & shape)

An identical object, matching all the properties except at a different position, to the red cube is placed in the environment. Although similarity and novelty/habituation are equally weighted, the agent draws its attention to the novel object, as its new position gets higher novelty than the identical position. With just 10% increase in the similarity weight ($\omega_1 = 0.6$), the agent's preference switches to reaching towards the familiar object, at the familiar position. Figure 5.4 shows the excitations of two reaching decisions (reach familiar and reach novel) in two simulated individuals, after reaching to the familiar object during stage one. In the figure, the lines represent the excitation for the two reaching decisions in the different individuals. Each pair of lines, having identical colour and symbol, represents the excitations for reach actions towards the familiar object (dashed line) and novel object (continuous line), in an individual. The action with the higher excitation, among each pair, is the individual's decision of reach action for either novel (continuous line) or familiar object (dashed line).

For each weighting, (colour and marker in Figure 5.4), the executed action is the one with the highest excitation. The first three executions in the figure represent the familiarisation stage of the experiment. The dashed lines represent the reach for the familiar object, whereas the continuous lines represent the reach for the novel object. Note the novel object is only introduced following the completion of the familiarisation stage (three executions). The enclosed figure shows for the novel object at equal weightings (red star) the excitation of the "reach for novel" object is higher, whereas, with a similarity weighting of 0.6 (blue circle), the excitations are almost the same, but with "reach for familiar object" marginally higher. At this point, the agent prefers to reach for the familiar object rather than the novel one. Although, the gap between the two excitations, for novel vs familiar, is small, the agent follows the winner takes all rule, hence the size of the gap is not important. Thus, a slight increase in the weight for similarity (ω_1) enabled the agent to prefer the familiar object over

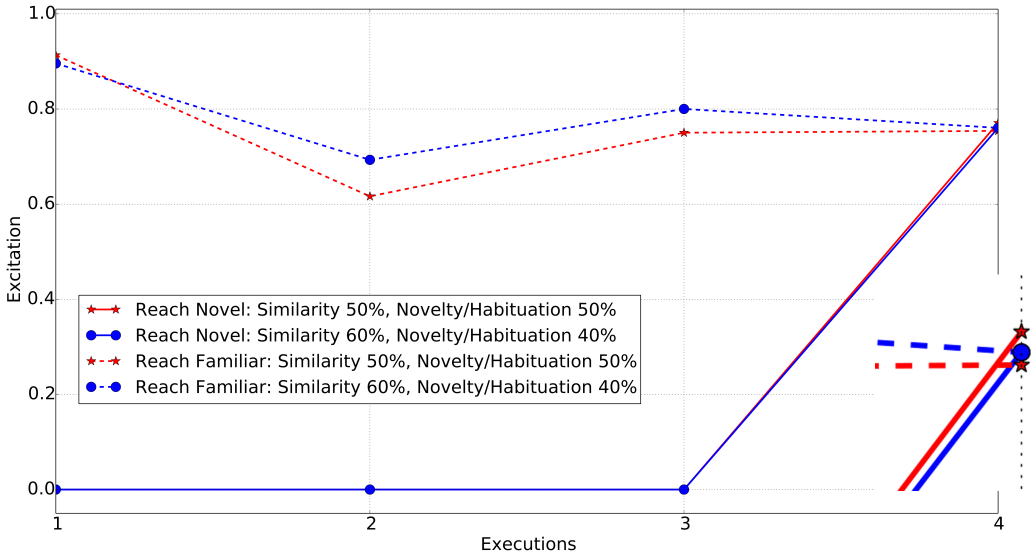


Fig. 5.4 Reach actions for Familiar (dashed line) vs. Novel (continuous line) (identical in colour and shape) object. The enclosed figure shows the excitations at the 4^{th} execution.

the novel.

Thus, Figure 5.4 shows excitation for reach actions in the two individuals, with different weights for the excitation parameters. One individual (red stars), having $\omega_1 = \omega_2 = 0.5$, prefers to reach for the novel object, at step 4, after reaching towards the familiar object for three times previously. Whereas, the other individual (blue circles), having $\omega_1 = 0.6$ and $\omega_2 = 0.4$, prefers to reach for the familiar object at step 4, over the novel one, even after reaching for it previously.

5.2.2.2 Novel object with change in the single property

In this part, after the familiarisation stage, the agent is introduced with a novel object, varying in one feature, either colour or shape. Varying just ω_1 from 0.5 to 0.7, it is observed that the agent interacts with the novel object, i.e., the blue cube or the red ball after being familiarised with the red cube. The novel feature i.e., colour or shape, of the novel object attracts the agent's attention, therefore it prefers to reach for the novel object over the familiar one even

for the higher similarity weights i.e., ω_1 from 0.5 to 0.7. Changing ω_1 further to 0.8, and ω_2 to 0.2, the agent's behaviour switches from interacting with the novel object to interacting with the familiar one during the test condition.

The excitation of the novel object staying higher and a greater weighting towards the similarity is required to cause the shift in behaviour. The additional variation in the object properties results in the agent interacting with the novel object instead of the familiar one, until a higher weighting towards the similarity parameter is applied to draw the agent's attention towards the familiar object. At this level (similarity weight $\omega_1 = 0.8$), the low weight to the novelty/habituation parameters ($\omega_2 = 0.2$) counters the excitation generated from the different properties. Figure 5.5 shows the excitation of the "reach novel vs. familiar object" schemas for the different values of the excitation parameters.

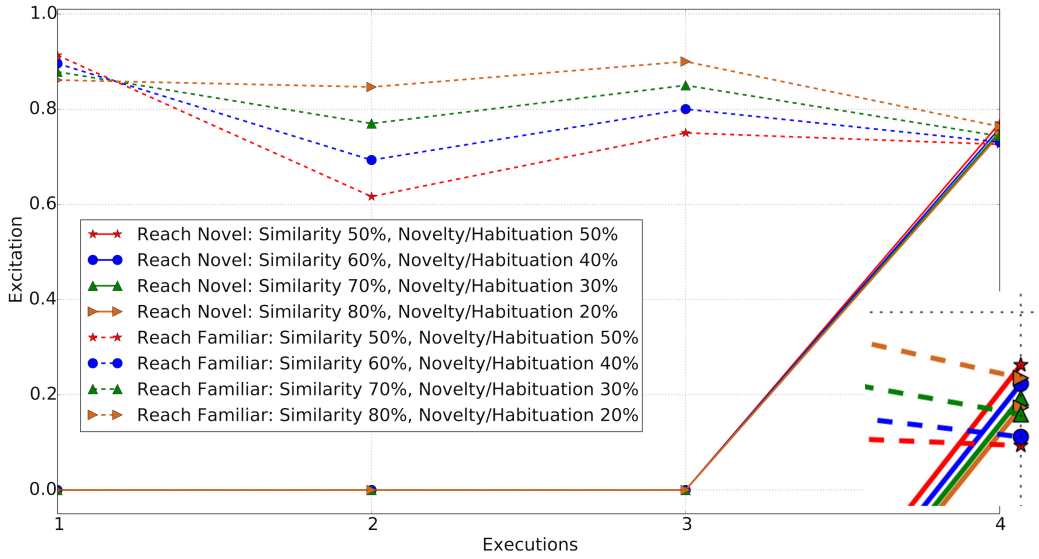


Fig. 5.5 Reach actions for Familiar, in dashed line, vs. Novel (change in either colour or shape) object, in continuous line. The enclosed figure shows the excitations at the 4th execution.

Changing the similarity weight values allows several individuals to be simulated. For weights in the range $0.5 - 0.7$ for ω_1, the agents are found to be interacting with the novel

object, however, each of these has different excitations for the reach actions towards the novel and familiar objects. When the similarity weight is set to 0.8 or above, the agent is more likely to interact with the familiar object rather than the novel one. Furthermore, it is anticipated that both the object and schema excitation weights (i.e., ω_3 and ω_4 in Equation 5.8) will cause the agent to habituate with the same object and action when the agent is allowed to interact with the world for a longer period of time.

Thus, Figure 5.5 shows that the excitations for reach actions in the four individuals (stars, circles, triangles and right arrows), with different weights for the excitation parameters. The three individuals (stars, circles and triangles), having $\omega_1 = 0.5 - 0.7$ and $\omega_2 = 0.5 - 0.3$, prefer to reach for the novel object after reaching towards the familiar object for three times previously. Whereas, the other individual (right arrows), having $\omega_1 = 0.8$ and $\omega_2 = 0.2$, prefers to reach for the familiar object, over the novel one, even after reaching for it previously.

5.2.2.3 Novel object with change in the both properties (colour & shape)

If the agent is presented with a novel object having different shape and colour, following the familiarisation, it requires a greater weight to the similarity to draw its attention towards the familiar one. The novel object provides higher excitation due to novel shape and colours, being at a different position. The results show that the agent interacted with the novel object after familiarisation whilst the weight ranged between 0.5 and 0.8 for ω_1, by 0.1.

When ω_1 was set to 0.9, the agent had a higher preference for familiar features, hence drew its attention towards the familiar object and reached for it. At this point, although, the novel object has higher novelty, the agent prefers to reach for the familiar object due to the higher weight for similarity. Figure 5.6 shows the excitations for schemas for reaching towards the familiar and novel objects.

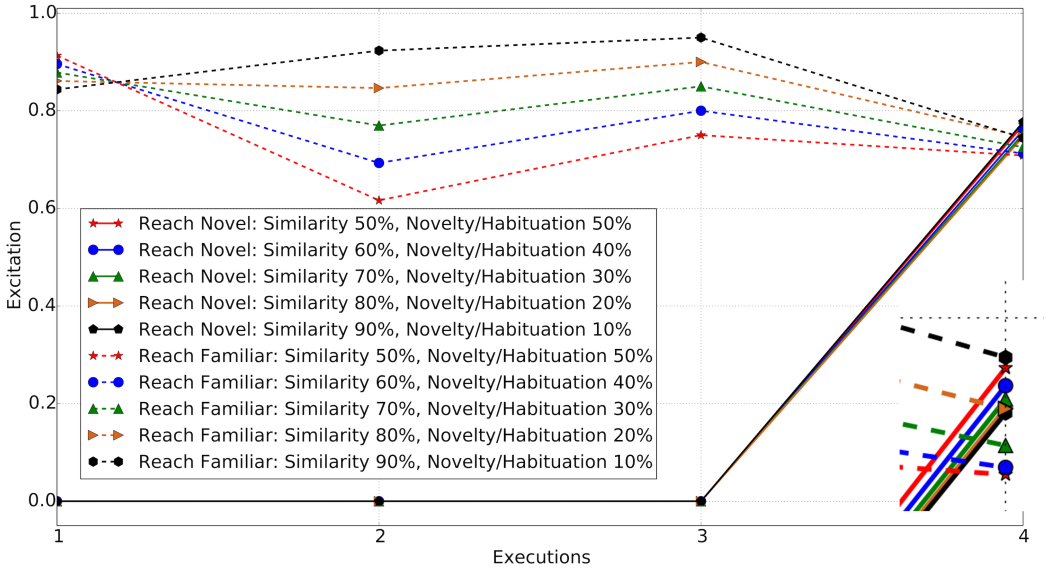


Fig. 5.6 Reach actions for Familiar, in dashed line, vs. Novel object, in continuous line (changed in both colour and shape). The enclosed figure shows the excitations at the 4^{th} execution.

Figure 5.6 shows that increment in the similarity decreases the excitation of the schema related to reaching towards the novel object, whereas that for the familiar increases. However, the "novel object" remains more interesting for the agent to interact with until the similarity is weighted 90% (0.9) of the total object excitation. When ω_1 is set to 0.9, the similarity component makes the overall excitation of the familiar/similar object higher than that for the novel object, causing the agent to reach for the familiar object rather than the novel one.

Thus, Figure 5.6 shows excitation for reach actions in the five individuals (stars, circles, triangles, right arrows and hexagons), with different weights for the excitation parameters. The four individuals (stars, circles, triangles and right arrows), having $\omega_1 = 0.5 - 0.8$ and $\omega_2 = 0.5 - 0.2$, prefer to reach for the novel object after reaching towards the familiar object for three times previously. Whereas, the fifth individual (black hexagon), having $\omega_1 = 0.9$ and $\omega_2 = 0.1$, prefers to reach for the familiar object, over the novel one, even after reaching for it previously.

From Figures Figs. 5.4 to 5.6, it is evident that the agent's preference in the environment changes with the variation in the excitation weights ω_1 and ω_2. A weighting towards ω_1 will increase preference towards the familiar objects. However, as the difference between the familiar and novel object increases, so do the required weighting towards ω_1, in order to draw the agent's attention towards the familiar object.

In this experiment, the agent is shown to express different behaviours for the novel object, while the weights of the similarity and excitation parameters change. A summary of the points at which the changes occur is given in Table 5.1.

Matching Properties	Behaviour Change Sim / Nov-Hab (ω_1 / ω_2)
Two	0.6 / 0.4
One	0.8 / 0.2
Zero	0.9 / 0.1

Table 5.1 Summary of the weightings at which the observed behaviour changed the preference from novel to familiar.

5.2.3 Experiment 2: Action Preferences

This experiment is inspired by the study of "Stimulus variables which affect the concordance of visual and manipulative exploration in six-month-old infants" by Steele and Pederson [182]. This study investigates habituation in the infants through continuous visual and manipulative experiences. It was observed that the infants' engagement in both experiences decreases with each trial, hence both engagements were habituated with the continuous experience. In this experiment, we demonstrate action switching in a Dev-PSchema enabled agent for a set of perceptions experienced continuously. We consider the action switching behaviour as the action habituation, observed in the infants with the continuous manual engagements [182].

By varying the excitation parameters described in Section 5.1, several different behaviours emerge from interacting with the environment. In this experiment, we vary the weights ω_3 and ω_4, keeping ω_1 and ω_2 constant (0.5 each). We examine the agent's preference for either favour a recently executed action or switch to a different action during a series of executions. For this experiment, we use the same agent and the environment described in Experiment 1 above. To demonstrate the behaviour with different preferences for actions, we provide the agent with two different actions, "Press" and "Squish", which produce the same outcome in the environment. The purpose of this experiment is to demonstrate the variations in the behaviour of the agent by changing ω_3 and ω_4, whilst keeping ω_1 and ω_2 constant. Having the same outcome/postconditions and the object for both actions give the same similarity and novelty/habituation. Thus, the excited schema (hence excited action) depends only on the schema excitation, described in Equation 5.7, calculated through incorporating the schema statistics.

We only use one object in the environment for this experiment to control the variation in object excitation, and place the end-effector at the same position as the object to remove the reach action from this experiment. Each action, squeeze or press, responds with a new observation, "press", in the environment, which provides the same similarity value for the both action schemas.

5.2.4 Results: Experiment 2

We let the agent play with the object using the actions and record which action is selected at each execution. The agent's observed behaviours for different values of ω_3 and ω_4 are shown in Figure 5.7.

Figure 5.7 shows the most excited schema, hence the action, for each execution at the different pair values of the ω_3 and ω_4 for 10 executions. From the results, it is evident that the agent shows different behaviours as the weights vary, thus representing as different

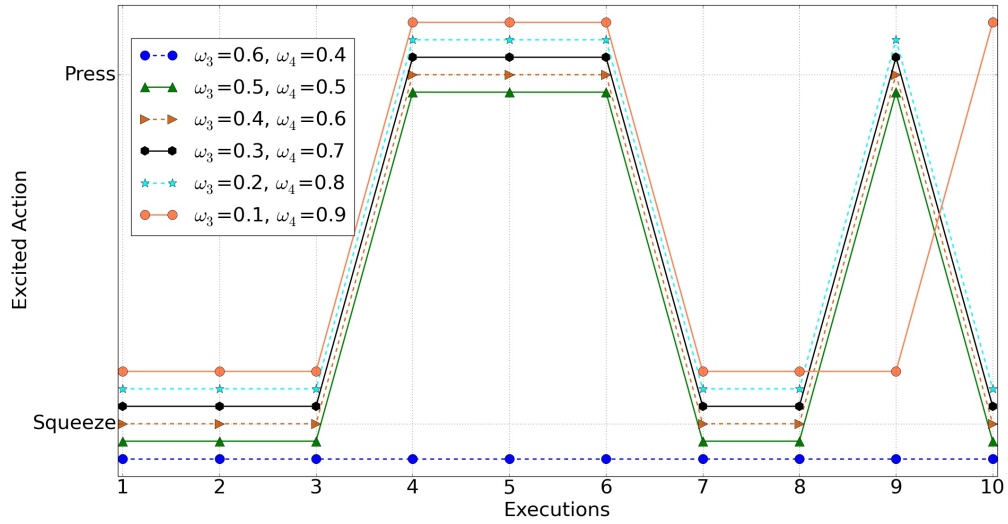


Fig. 5.7 Excited schema action for different values of ω_3 and ω_4. Lines off-set for visibility

individuals. As the weight shifts towards ω_4, the agent becomes increasingly inclined to frequently switching between actions, rather than to explore the effects of the previous action further.

For all the ω_4 weights the agent initially performs the squeeze action three times consequently, except for $\omega_4 = 0.4$. Following that, it performs the press action as it gets more excitation than the squeeze action. After performing the press action three times, the agent prefers the squeeze action again as it gets more excitation. At this point both actions have been executed an equal number of times, however the press action has been executed more recently. At values 0.5-0.8 for ω_4, the agent starts alternating between the two actions. However, at the value of 0.9 for ω_4, the agent still performs the squeeze action for the third time before switching action again, as the press action has been executed recently resulting lesser excited than the squeeze action.

In this experiment, only one object and two different actions, press and squeeze, were used. For novelty/habituation calculation τ_1 was recorded as 0, 0.5, 0.67, 0.5, 0.4, 0.33,

0.28, 0.25, 0.22 and 0.2 for executions steps 1 to 10 respectively. Whereas, values of τ_2 for execution steps 1 to 10 were recorded as 0, 0, 0.5, 0.67, 0.75, 0.8, 0.83, 0.85, 0.88 and 0.89 respectively. However, novelty/habituation remained the same for both actions and the excitation was only based on the individual schema excitation.

5.3 Discussion and Conclusions

A habituation paradigm is widely used in developmental psychology experiments to test infants' ability to identify or recognise objects [203, 163, 168], or events [86, 158] based on visual perceptions. These studies show that infants tend to look longer towards the novel objects or novel and unexpected events than those with which they are familiar or able to predict. However, infants have been observed to have favourite objects for interaction and play [57, 81]. Also, it has been observed that young children prefer their favourite toy over new toys, even in the presence of a brand new version [59]. In the experiments by Gelman and Davidson [59], young children were asked to select a toy from a choice of their own or a new toy (identical and non-identical). They preferred their own toy when they were asked to choose a toy for themselves and preferred the novel object when they were asked to select one for the experimenter.

In another study, Sigman [175] investigated exploratory behaviours of the pre-term and full-term infants at the same conceptional age. Following the object familiarisation, the infants were provided with the same object along with another novel one. The experimenters observed that the both groups explored the novel objects more than the familiar object. However, the pre-term infants explored the familiar object for longer than the full-term infants. Similarly, Ruff [161] examined behaviours of 7-month and 12-month infants with a set of objects over a period of time. Different activities such as examining, mouthing and banging, were recorded during the experiment. It was observed that the examining activity occurred before the other activities when a new object was presented and the examining

activity decreased over the period of time. Furthermore, the 7-month old infants spent more time on examining and mouthing than the 12-month old infants. The activity time for bashing the new object was, also, found to be increasing over the trial for the both age groups.

Steele and Pederson [182] investigated the effect on visual fixation and manipulation with toys across 10 continuous trials in 26 weeks old infants. They were presented the same toy for the 1^{st} to 7^{th} and 10^{th} trials and a novel object was introduced at 8^{th} and 9^{th} trials. Fixation and manipulation time was found to be decreasing at each trial. However, fixation time was increased at the 8^{th} trial when a novel object was introduced, different in either colour, shape, texture or shape and texture. The manipulation time was increased when the novel object consisted of a different shape and texture. However, the manipulation time was found to continue decreasing when the novel object only differed in colour.

These studies, [57, 175, 59, 81, 182, 161], demonstrate that over a period of continuous interaction with an object an infants interest declines leading to them increasingly seeking out novel objects and events. However, they show variations in their behaviours or decisions [59, 182]. This demonstrates that the attention for different objects depends upon the individual preferences and experiences. Excitation and attention are seen as important factors for individual behaviours in developmental psychology. Colombo [38] considers alertness, object features, spatial orientation and endogenous control as the basic factors that affect visual attention in the environment. Although vision is the least developed sense at birth, humans have evolved to rely heavily on this sense [176]. These experiments are concerned with the last three factors of visual attention; object features, spatial orientation and endogenous control. Object features and relevant spatial orientation are inseparable from one another. The endogenous control factor in visual attention is responsible for holding attention and engagement. The novelty-familiarisation pair is used in developmental psychology to investigate visual attention in humans. To investigate the attention in the first experiment, the simulated infant is initially familiarised (habituated) with a visual stimuli or event and then is presented with a novel and the familiar objects side by side, a procedure seen during

experiments with human infants [203, 59, 182, 163, 168].

In particular, Steele and Pederson [182] found that the infants' engagement with the objects decreases with each trial, hence the objects get habituated. The infants tend to engage more with the objects if they are novel visually. In experiment 1, we demonstrated that the agent tends to interact with the novel objects if they differ more from the habituated. Thus the weight required for the similarity, to drive the attention towards a familiar object, increases as the visual change in the novel object increases (see Section 5.2.2).

From the results in developmental psychology experiments, discussed above, it is evident that children show different behaviours for novel and familiar objects depending upon their experiences and preferences. This effect was reproduced within the experiments here by changing the weights of the excitation parameters. The results demonstrate the capability of the system to generate different behaviours when interacting with novel vs familiar objects. The agent also transfers the habituation from the habituated objects to a perceived object having similar perceptions. This results in low novelty, hence higher habituation, for the newly perceived object, as observed in the first part of Experiment 1 (i.e., same colour and shape). Therefore, only a small change in favour of the similarity weighting triggers a change in observed behaviour. However, as the novel object becomes increasingly different (hence interesting), the novelty value of it becomes increasingly higher, requiring a greater weighting on similarity to cause the change in agent's behaviour, as observed in Section 5.2.2.3 (Experiment 1). Similar behaviours were observed in infants where they show more interest in novel objects than the familiar [182].

This behaviour of the artificial agent can be compared with the infants' behaviours. While many of the parameters were controlled, particularly in Experiment 1, it should be clear that within the pairs of weights, a higher weighting on ω_1 will drive the agent to spend longer exploring the same object, and a higher weighting on ω_4 will encourage the agent to try different actions. By adjusting each of the weights, different behaviours can be simulated.

This could be considered as modelling different infants' preferences, or different external conditions under which the agent is acting. Currently, the weights are fixed at the start of an individual experiment, but in the future allowing the agent to vary these, could generate a shift from exploratory play behaviour to more exploitative or focused behaviour. Figure 5.8 shows the possible object preferences and behaviours of the agent for different values of the excitation parameter weights i.e., ω_1, ω_2, ω_3 and ω_4.

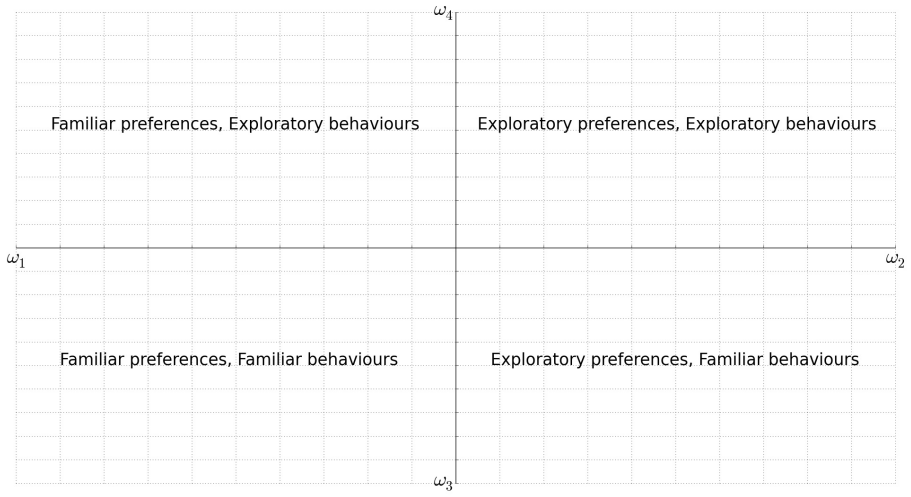


Fig. 5.8 Behaviours of the agent over the axes for ω_1, ω_2, ω_3 and ω_4

In these experiments, we have presented the excitation mechanism and demonstrated the effect of varying weights related to similarity, novelty and habituation. As the agent interacts with the objects, variations in the weights lead to the expression of different exploratory behaviours. Experiment 1 illustrates the variation in behaviours of the agent by changing the weights of the similarity and novelty/habituation pair (ω_1 and ω_2), while keeping the object and schema excitation weights constant (ω_3 and ω_4). Similarly, experiment 2 demonstrates the variations in behaviours of the agent by changing the weights of the object and schema excitation (ω_3 and ω_4), keeping the similarity and novelty/habituation weight pair (ω_1 and ω_2) constant. This aspect of the system enables Dev-PSchema to simulate different

individuals with individual behaviours rather than a single simulated agent with average behaviour. It also enables the agent to switch behaviours from exploratory to more focused behaviour and vice versa.

Researchers in developmental psychology mostly represent average results in their work [67, 13, 59]. However, individuals demonstrate different behaviours and preferences, contributing to the average result presented in that study [3, 186]. Similarly, in robotics applications, inspired from developmental psychology, researchers simulate an average behaviour represented in developmental psychology studies [172, 141, 125, 83, 6, 136, 92]. In contrast, in these experiments we have demonstrated the ability of the system, Dev-PSchema, to simulate different individuals with different behaviours rather than an average behaviour and preference. The variations among simulated individuals in their behaviours lead them to develop skills through different developmental paths. A similar development is also observed in infants. Thelen et al. [186] observed variations among four individual infants in their behaviours and paths for developing reach onset. During the experiments, each infant demonstrated different levels of activities and different preferences for the movement patterns. This observation is also backed by Adolph et al. [3]. In their study, [3], the authors observed variations among individuals while developing different locomotive skills. The results also demonstrated variations among the infants in decision making during various locomotion tasks, e.g., crawling down a slope.

In conclusion, we have demonstrated and evaluated the capability of Dev-PSchema to simulate different individuals through the experimental results reported above. We have focused on the excitation mechanism and its parameters to demonstrate its importance in the agent's behaviours. The agents' behaviours show attention, interest and their preferences in the environment.

Chapter 6

Forming Higher Level Actions

Most actions performed by humans on a day to day basis are defined by high-level objectives and actions. These objectives are typically achieved by a series of low-level motor actions or a sequence of actions, referred to as primitive actions. For example, drinking water is a high-level objective, which can be achieved by a sequence of actions, such as reaching for the glass, grasping it, filling it with water, opening of mouth while transporting the glass to the mouth, and adjusting the angle of the glass in the mouth to enable comfortable drinking. In this example, a sequence of primitive actions is executed to achieve the overall objective, drinking water. These primitive actions are continuous and often inseparable from each other whilst predominately maintaining the sequential ordering, with some occasional overlap between actions. For example, grasping an object involves reaching and curling the fingers around the object. These two actions are executed in a particular temporal order, however, a pre-grasp shape will have been formed as the arm reaches to the glass [194].

From developmental psychology, we have extensive evidence that sequences of actions are planned and executed as a single high-level action. This capability is seen in the early stages of life in humans. Piaget believed that the ability to plan and execute a sequence of actions to achieve short objective appears in early infancy. According to him, infants are able to achieve short objectives at the fourth sub-stage, Coordinating Secondary Schemes, (8 – 12 months old) of the cognitive developmental theory [147], see Section 1.4.1 for details. For

example, at this age infants can reach for an object by avoiding or removing any object in the path [119]. The objective, reaching for the target object, is achieved through planning and performing the actions starting from moving the hand forward towards the target either by avoiding the obstacle or removing the obstacle first.

Weigelt and Schack [199] found that 3-5 year old infants change their hand orientation to grasp an object based on the goal in mind. The children, in the experiment, were asked to pick and place one end of a two coloured cylindrical object. It was observed that the children were picking the object in such a way that they could easily place the appropriate end as directed. Efficiency in the task increased with age, however, younger children were also found with the capability to achieve comfort for the end state, a final phase of a series of actions or situations. This ability has also been investigated in younger infants. In a similar experiment, McCarty et al. [118] found that 14 and 19, but not 9, month old infants planned the sequence of actions before execution. The researchers found that the older infants reached and grasped the spoon in such a way that offers clean transportation to their mouth. Achieving a proper grip to get the spoon to their mouth provides an evidence for action planning in young children.

In the psychological studies discussed above, the subjects performed sequence(s) of actions; reach, grasp, pick, transport and place, on the different objects. The action sequences were structured from a given state of the environment to the objective (state), i.e., the goal. Initially, psychologists focusing on sequential actions believed that such action sequences were actually reflexive chains, labelled to as "reflex chains" [160]. In reflexive chains, a sequence is executed in such a way that each action outcome triggers the next action in the sequence. This theory defines that sensory feedback plays an important role in sequence execution. Later, Lashley [98] raised a question on the *reflexive chain* theory regarding the action sequences. He believed that action sequences can be executed even when feedback is interrupted, as most of the sequences are executed so quickly that individual actions within the sequence cannot be triggered by the sensory feedback. In addition, errors in behaviours containing sequential executions suggest that the entire behaviour is planned as a single

sequence rather than each step triggered by feedback [160]. According to Lashley's theory, action sequences are planned before execution and do not depend upon intermediary sensory states.

Rosenbaum et al. [160] reviewed Leshley's claims about action sequencing. Referring to the psychological studies on end-state-comfort, the authors believed that in such action sequences, the subjects used immediately generated plans in the following similar tasks. The hand-path priming experiments, where the hand follows a previously used path in a task [79], provide further evidence that the action sequences are potentially pre-planned and independent of intermediary sensory feedback. van der Wel et al. [190] in their experiments found that subjects anticipated sensory states rather than interrupting the sequence to get the sensory state in an obstacle avoidance experiment. They found that subjects raised their hand higher when an obstacle was expected, than when no object was expected during an object transportation task between two points. They also found that subjects continued to raise their hand, even after passing the obstacle. In a similar study, Jax and Rosenbaum [79] found that subjects used previously generated sequence plans irrespective of the feedback during the sequence execution. In the experiment, subjects were asked to transport an object while avoiding the obstacle. They found that subjects used a more curved path when there was no obstacle present in the path, having previously completed the same task involving obstacle avoidance, than the subjects who previously completed the same task involving no obstacle in the transportation path. Similarly, Kent et al. [87] in their experiment asked their subjects to grasp a two coloured object with their thumb and index finger, specifying the colour and side their thumb should touch. They observed that subjects who previously performed the same task, tended to use a similar grasp to perform the same action again, even when the grasping constraint was no longer present. Whereas a random group used more efficient approaches to perform the task in the same situation. Similar to Kent's [87] experiment, Dixon et al. [43] asked the subjects to grasp from a particular part of the object and rotate it. They found that the subjects reused action sequences that had been generated from previous

trials, rather than planning a new and efficient sequence to achieve the objective.

From the above discussed studies, it is suggested that previously generated action sequences are re-used in repeated situations, with limited need for constant sensory feedback. These studies also suggest that action sequences are planned in advance and are executed as a single continuous action after some repetitions, rather than as a series of individual primitive actions. In psychology, it is also believed that a repetitive sequence of actions is performed by a high-level motor command from the brain which emerges from practice and experience [85, 53, 185]. Through repetitions, an action sequence often leads to being performed as a single smooth movement [53]. In repeated action sequences, such actions become part of what is referred to as a *motor program* [85, 185]. Motor programs are related to memory and learning, and defined to be a set of motor commands structured before executing a sequence of actions, which can be executed with limited or no peripheral feedback [185, 129], as if they were a single fluid action. Each action in a motor program is executed in quick succession to form a continuous action, with limited or no peripheral feedback [160, 98]. Interruption of a motor program typically results in the individual restarting the full sequence, rather than being able to continue from the point of interruption.

With age, action sequencing becomes an important part in our daily routines. Action sequences are created, adapted and executed to achieve different objectives. Without the learning and recall of commonly used action sequences, our minds would constantly be caught up with planning fine motor actions required to achieve each task. Thus action sequences created in one task may be re-used in a similar situation or environment without recreating and planning a new sequence, as discussed above. In robotics, robots may also need to achieve objectives via a sequence of primitive actions. Thus, the robots should be able to develop complex actions using basic actions which have been applied in the environment repeatedly. We have developed a chaining mechanism in Dev-PSchema, enabling it to create and use action sequences, referred to as *schema chains*. The chains are discovered with exploration and available for re-use in similar situations, triggered through the excitation

mechanism discussed in Chapter 5.

In Section 6.1 we describe the Dev-PSchema mechanism for creating and executing schema chains. In Section 6.2 we provide details of experiments, and the results, performed to evaluate the chaining mechanism. Finally, in Section 6.3 we evaluate the chaining mechanism based on the experimental results and provide a conclusion. We would like to acknowledge that parts of this chapter, particularly Experiment 2 (Section 6.2.3), have been published in a peer reviewed paper, given below:

- Kumar S., Shaw P., Giagkos A., Braud R., Lee M.H., Shen Q. Developing hierarchical schemas and building schema chains through practice play behaviour. Frontiers in Neurorobotics. 2018;12:33.

6.1 Schema Chains

As an agent, equipped with Dev-PSchema, gains more experiences and skills, some of the skills can be linked together in order to form higher level skills. For example, individual actions such as reaching and grasping can become linked by a single reach→grasp action. This combination is developed through creating a sequence of existing action schema, referred to as schema chains or simply chains. Through playful exploration, more complex chains can be learnt that combine basic actions and form more sophisticated high-level actions, hence skills.

Chaining helps in achieving distant states that are not possible when employing a single schema. Schema chains are found during the excitation calculation process, described in Chapter 5. Schema chains are found to achieve postconditions of a schema containing the preconditions that do not match the current state of the environment. Thus, the aim of the chaining mechanism is to achieve a state (schema postconditions) in the environment through a sequence of schemas. The chaining mechanism finds links between the preconditions and

postconditions of two different schemas in the memory to develop a chain. Figure 3.11, shows an example of a two schema chain obtained by linking the preconditions and postconditions of two different schemas. Longer chains are discouraged during the chaining process in order to reduce computational costs and avoid overly complicated chains that are more likely to be unsuccessful. Here, a limit of 4 schemas is set.

In PSchema chains were only found and executed to achieve a target state provided by an external agent [172]. Dev-PSchema calculates the chains itself through the excitation mechanism and executes if any of the discovered chains is found to be most excited. Thus providing an opportunity for Dev-PSchema enabled agents to develop high-level actions through a combination of basic actions and make use of them in play behaviour. Algorithm 18 describes the mechanism for developing chains using an initial and target sensory state.

Algorithm 18 finds a link between an initial state (WS) and a target state through schemas in the memory. In the problem solving mode, the target state is provided by the user. The problem solving mode is further discussed in Chapter 7. The chaining process starts by finding the schemas that have similar preconditions as the initial state, usually a current environmental state. The schemas S_s, line 2 of algorithm 18, contains preconditions which are a subset of the current environmental state, WS. The algorithm finds links between postconditions in schema S and any other schema, S', in the memory. Any schema(s) linking S and the target state form a chain from S to S' that contains postconditions matching the target state, a schema postconditions in the play mode or user-defined state in the problem solving mode. The algorithm adds all the possible chains, for a given state of the environment, and returns a list of all possible chains to achieve the target state. Each calculated chain is provided with an estimated success rate, which helps to select a suitable chain. In the play mode, the estimated success rate is calculated by taking the average success rate of all the schemas present in the chain, whereas in the problem solving mode the agent calculates it through equally weighted similarity between first schema preconditions and the current state of the environment, combined with the average success rate of the schemas in the chain. The

Algorithm 18 Schema chain calculation

```
1: function find_path (WorldState WS, WorldState Target, Boolean Problem_Solving)
2:     S_s = schemas with similar preconditions as WS
3:     Chains = empty list; Chains_found = empty list of pairs
4:     for each schema S in S_s do
5:         Start = S; currentChain = [S]
6:         while Start_postconditions ≠ Target do
7:             for each schema S' in Long-term Memory do        ▷ Optimisation applied, see Section 6.1.1
8:                 if Start_postconditions ≅ S'_preconditions then
9:                     Append S' to currentChain
10:                    if S'.post ≅ Target AND length(currentChain) < 5 then
11:                        Add currentChain to Chains; break        ▷ Go to line 4 for next S
12:                    else
13:                        Start = S'
14:                    end if
15:                end if
16:            end for
17:            if length(currentChain) ≥ 5 then
18:                break                                        ▷ Restrict chain length
19:            end if
20:        end while
21:    end for
22:    for each chain C in Chains do
23:        chainProb = 0.0; excs = empty list
24:        for each schema S in Chain C do
25:            chainProb += S_r(S)                              ▷ See Equation 3.1
26:        end for
27:        chainProb = chainProb/length(C)
28:        if Problem_Solving is True then                      ▷ This is False by default
29:            sim = states_similarity(1st schema_preconditions in C, WS)
30:            chainProb = (sim × 0.5) + (chainProb × 0.5)
31:        end if
32:        prob = chainProb × C_r                               ▷ C_r = 1 if chain not exists/executed previously
33:        add Pair [chain C, prob] to Chains_found
34:    end for
35:    return Chains_found
36: end function
```

estimated success rate in both cases, problem solving and play, is further weighted with the chain success rate, if the chain already exists in the memory, see line 32 in the algorithm. The chain success rate (C_r) is a ratio between chain successes and its activations, if any. Any newly created chain has C_r set to its maximum value i.e., 1.0.

In the play mode, the agent uses the chain with the highest estimated success rate for the schema excitation with which the chaining process began, Line 5 in Algorithm 17. In problem solving mode, the user is provided with list of all possible chains and their estimated success rate. The agent performs a user selected chain in the environment in problem solving mode.

6.1.1 Chain Optimisation

If a schema preconditions do not match the current state of the environment the chaining mechanism is executed when attempting to calculate its excitation, then the excitation calculator triggers the chaining mechanism. To reduce the computational cost we mark *child* (i.e. successor) and *parent* (i.e. preceder) schemas for each schema in the memory. Whenever a new schema is constructed, its *child* and *parent* schemas are found through comparing the preconditions and postcondition with each other schema in the memory. This provides a limited set for searching schemas to build a chain. Furthermore, a limit has been applied on potential chain length, see Algorithm 18. The child/pairing process and limit on the chain lengths help to reduce computational cost consumed in calculating excitation for each schema in the memory.

If a schema A is found to be a parent for a schema B, then the schema B is labelled as a child to the schema A. They are then used to limit the search algorithm during the chaining process, enabling the algorithm to cycle through a limited set of schemas rather than all the schemas in the memory, see line 7 in Algorithm 18.

6.1.2 Chain Excitation

During play mode, the agent calculates excitation for all the schemas and chains existing in the memory for a given state in its environment. Excitation for each chain, existing in the long-term memory which was found and used previously, and existing in the short-term memory found through Algorithm 18, is calculated through Algorithm 19.

Initially, the algorithm finds if the chain is relevant for the current state. Any irrelevant chains get minimum excitation i.e., 0.0. The algorithm follows two different paths for excitation calculation depending upon the user-defined play mode, either “encouraged chains” or not. In “encouraged chains” mode the agent prefers to use chains over the individual schemas.

Algorithm 19 Chain excitation calculation

```
1:  function calculate_chain_excitation (WorldState WS, Chain C, Boolean chains_encouraged)
2:      MostExcitedChain = None
3:      ChainExcitations = empty list
4:      if Not states_match (First schema preconditions in chain, WorldState WS) then
5:          Return 0.0                ▷ Chains irrelevant to the current state get minimum excitation 0.0
6:      end if
7:      if chains_encouraged is True then                          ▷ Flag is set false by default
8:          for i = 0 to length(C) − 1 do
9:              given schemas s_i and s_{i+1} in chain C
10:             ChainEx =Diff(s_i, s_{i+1})
11:             Add ChainEx to ChainExcitations          ▷ In the code this is used for sorting
12:         end for
13:         ChainEx = ChainExcitations/(2 × length(C))
14:     else
15:         chain_sim = state_similarity (First schema preconditions in chain C, WS)
16:         chain_schema_excitations = empty list
17:         for each schema S in Chain C do
18:             Add schema S excitation from list schema_excitations to chain_schema_excitations
19:         end for
20:         chain_excitation = (0.7 × chain_sim) + (0.3 × Avg(chain_schema_excitations))
21:     end if
22:     ChainEx = ChainEx × C_r                          ▷ For C_r see line 32 in Algorithm 18
23:     Return ChainEx
24: end function
```

The function **Diff** (line 10 in the algorithm), returns an excitation based on the changes in the preconditions of schema s_i with the postconditions of the following schema, s_{i+1}, in the chain. This provides a chain excitation based on the expected changes being caused by the chain. In the other mode, chain excitation is calculated through the combination of the average success rate of all the schemas in the given chain and the similarity between the first schema preconditions in the chain and the current state of the environment.

The chain success rate depends upon the success rate of individual schemas within the chain. An average success rate of individual schemas is calculated so the chain can compete with individual schemas for execution. The probability of success rates may also be calculated through multiplying individual success rates, however, this will reduce the chain excitation further, hence the chain will be less excited than the most excited schema. Consider a chain of three schemas, having success rates of 0.6, 0.7, 0.8 respectively. The probability of the three will be 0.336, whereas the average success rate will be 0.7. Thus average success rate, rather than the success probability, will help the chain to compete for

execution with the individual schemas.

The combination of the chain similarity and average schema excitation is further weighted to calculate final excitation of the chain. In this work all weights are kept constant, 0.7 and 0.3 for similarity and average success rate of schemas respectively. The overall chain excitation is further weighted with its success rate.

6.1.3 Chain Execution

Schemas in a chain are executed in sequential order. After execution, a chain is considered successful if the resulting sensory state matches the last schema's postconditions. The chain execution mechanism is inspired from developmental psychology, performed either as a reflexive chain or a motor program. Both chain execution modes are described below:

6.1.3.1 Reflexive Chain

Newly created chains are executed in the reflexive chain mode. Sensory feedback is acquired at the end of every executed step (a schema) in the chain to compare with the expected postconditions of the executed schema. If it does not match with the expected state then the schema chain is considered unsuccessful. The term "match" means all the observations in the postconditions are obtained as an outcome. As the chain excitation depends upon the chain success rate, the chain is less likely to be selected for execution next time if it is unsuccessful.

6.1.3.2 Motor Program

If a chain is successfully executed repeatedly, then it is considered reliable and therefore becomes automatic, in a sense that it behaves as a singular continuous higher-level action called a *motor program*. As such, the chain is used to achieve a certain condition that results

from a hierarchy of actions. In the experiments, we revise at least four successful repetitions and a success rate higher than 80% to render a chain repeatable enough to be considered as a motor program. Motor programs are executed sequentially without the need of intermediate verification of the world state. That is, only the last action's resulting postconditions are used for the evaluation of the motor program. Consequently, if the validation fails the motor program's success rate is negatively affected turning it to a standard chain, a reflexive chain.

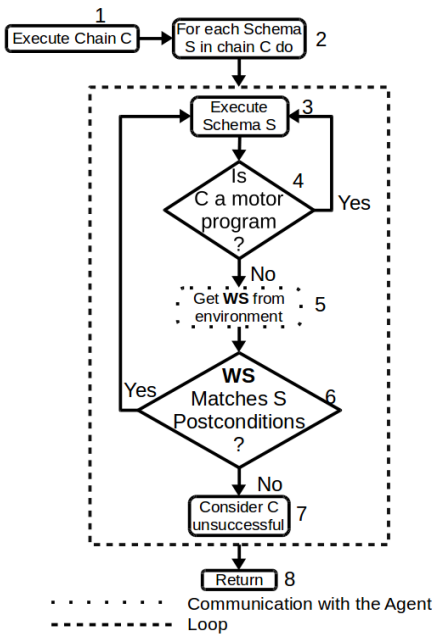


Fig. 6.1 Flow chart for a chain execution.

Figure 6.1 shows a flow chart for the execution of a chain in both modes i.e., reflexive chain and motor program. The figure shows the action from each schema in a chain is performed in sequence. Following each execution, the mechanism validates the outcome if the chain is in the reflexive chain mode. However, the validation mode is avoided if the executed chain is being considered as a motor program. Algorithm 20 describes the execution process of a chain either as a reflexive chain or as a motor program.

The chain execution mechanism requests a sensory state update, line 10, if the chain is executed in the "reflex chain mode", reflecting the action sequencing in humans discussed above. In the case of a motor program, the chain schemas are executed in sequential order,

Algorithm 20 Chain execution algorithm

```
1: procedure execute_chain(chain C)
2:     Increase C activations by 1
3:     if C not in the Memory then
4:         Add C in the Memory
5:     end if
6:     C_prob = C_successes/C_activations
7:     for Schema S in chain C do
8:         execute(S)                          ▷ See Algorithm 2 for schema execution process
9:         if C_successes < 5 OR C_prob < 0.8 then
10:            update_world_state of system     ▷ See Chapter 3, Section 3.5
11:        end if
12:    end for
13:    Return
14: end procedure
```

avoiding validating the sensory state. During the execution of a motor program, although the external state of the environment may not be directly monitored by the agent, the internal proprioceptive system is still active and the chain can still be interrupted if something unexpected was perceived.

We evaluated the chaining mechanism in Dev-PSchema using two experiments. The first experiment demonstrates the capability of a Dev-PSchema enabled agent to develop and execute chains in its environment. The chains are executed in the reflexive chain and motor programs modes following the successful repetitions. The second experiment demonstrates the agent developing schema chains through playful exploration and reusing them in the environment.

6.2 Experiments and Results

To test the chain-building capability of Dev-PSchema, we interfaced it with a simulator, *Sandbox* and a robot, iCub. The first experiment is performed with the simulator only, whereas the second experiment is performed with the simulator as well as with a real robot. To encourage the Dev-PSchema equipped agent to employ chains, rather than individual actions, we set the "*chain_encouraged*" flag signal to true in Algorithm 19.

6.2.1 Experiment 1: Demonstrating the Reflex Chain & Motor Program

This experiment demonstrates the development of schema chains and their executions, in a simulated environment, in both modes; reflexive chain and motor program. The simulated agent is able to perform "Reach", "Grasp" and "Press" actions in the environment. These actions are described in detail in Section 3.3. Initially, the agent performs all these actions in bootstrap mode to develop bootstrap schemas. Using bootstrap schemas, the agent then interacts with the environment and develops the schema chains. Later, one of the newly created chains is re-used in the environment to confirm it as a motor program. This experiment is inspired from psychological studies about reusing previously generated movements [79, 190]. In these studies, the subjects have been observed to re-use previously used reach trajectories to move an object avoiding an obstacle even if there was no obstacle in the path. In this experiment, the 'obstacle' is the requirement to press the button.

To test the chain-building and execution capability of the agent, we created different test scenarios. The objective is for the agent to create a chain of schemas to reach an object 'button' and press it, which will result in a second object being introduced into the environment. The agent should also create a chain to reach for the second object when it appears and grasp it. An object, Red button, is introduced in the environment after bootstrapping. The button can be pressed, but cannot be grasped. Pressing the button causes a red cube to appear in the environment.

The observation of the button reminds the agent about the hand at that position and the reach to that position was found to be the most excited schema. The press action is found to be most excited after the "Reach", which introduces another object, cube, in the environment. The new object triggers the reach schema towards the red cube. Figure 6.2 shows the sequence of actions as executed in the simulated environment.

Fig. 6.2 a) Button in the environment. b) Button is reached. c) Button is pressed, resulting in second object in the environment. d) Second object is reached.

This sequence of actions is neither directed nor previously learnt, but discovered by the agent itself through the exploration triggered by the excitation mechanism. Once this sequence is performed through individual actions, the environment is reset as it was at the beginning of the sequence. At this point, the agent creates two different chains; 2-schema chain with "Reach button" and "Press button" actions, and 3-schema chain with "Reach button", "Press button" and "Reach cube" actions. The excitation mechanism finds the 3-schema chain to be the most excited. Subsequent steps are shown in Figure 6.3.

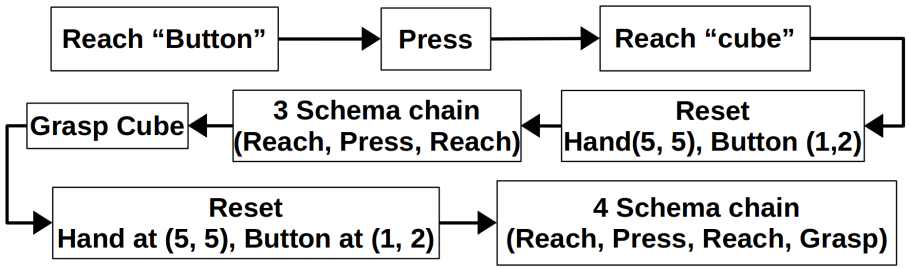


Fig. 6.3 Steps performed after bootstrapping.

Once the 3-schema chain is performed, the "Grasp" action is found to be the most excited and therefore executed. To test the chain execution as a motor program, we considered adding the "Grasp" action in the original sequence of actions, prior to the first reset. To create the 4-schema chain, consisting of the "Reach button", "Press button", "Reach cube" and "Grasp cube" actions, we reset the environment. This helped the agent to link "Reach button" to the sequence up top the "Grasp cube" schema, using the chain creation mechanism described in Section 6.1. Once the 4-schema chain is created, we let the system play in the environment, by resetting the environment every time the chain is completed (the cube is grasped), until the

chain becomes a motor program, after four successful repetitions. To test the newly created "motor program" of the agent, we used three test conditions, described as below:

Test I.	Second object, red cube, appears in the environment when the first object, red button, is reached rather than reached and pressed.
Test II.	Second object does not appear in the environment at all.
Test III.	Second object is present at the beginning of the experiment.

The results for this experiment are discussed in Section 6.2.2 below.

6.2.2 Experiment 1: Results

Once the 4-schema chain is created, following Figure 6.3, we let the agent interact with the environment using the excitation mechanism. Figure 6.4 shows excitations of schemas and chains at each execution step, from the very first step, i.e., reach action towards the button. Once the red cube is reached, at execution step 4, we reset the environment by sending the hand/end-effector to its initial position and removing the second object. In the following execution step (i.e., step 5), the 3-schema chain becomes the most excited chain. The agent executes this chain and end-up reaching the red cube, as shown in Figure 6.3.

Following the 3-schema chain execution, the agent identifies the grasp schema as most excited at execution step 5. After grasping the red cube, we reset the environment and let the agent explore the environment again. At this point, execution step 7, the agent creates the 4-schema chain and found it most excited. Every time the agent grasped the cube we reset the environment and let the agent play again. This process is repeated in execution steps $7-10$, with the 4-schema chain being highly excited in these executions, enabling the chain to be successful 4 times leading to the 4-schema chain becoming a motor program. Although the 4-schema chain is highly excited in these execution steps, its repeated executions will gradually reduce its excitation whilst other schemas or chains that have not been executed

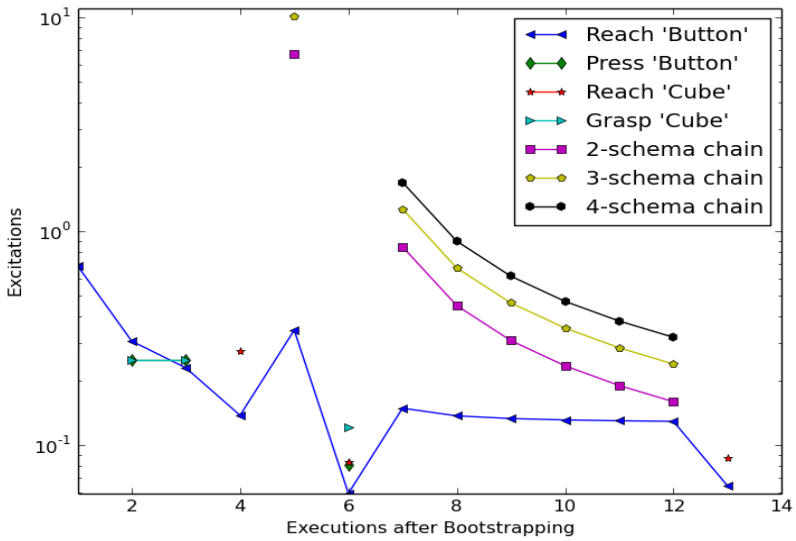


Fig. 6.4 Excitations of schemas in sequences along with 2, 3, and 4 schema chains. Executions are shown following the bootstrapping process, up to the 3rd test. At missing points in lines, excitation is either 0 or schema/chain do not yet exist.

for a while may increase in excitation sufficiently to become the most excited.

Once the chain reaches the criteria to be considered as a motor program, we start our test conditions. The results for each test condition are discussed below:

Test I. In this condition, the red cube appears in the environment when the button is reached with the 4-schema chain. At this point, execution step 11, the chain executes in motor program mode. Hence, the appearance of the cube before the press action on the button does not affect the chain execution. At the end of the chain execution, the agent validates the results by comparing the postconditions of the last schema in the chain with the obtained sensory state. As the red cube was present in the environment and the agent successfully grasped it, the execution of the chain is considered successful.

Test II. In this condition, the red cube does not appear in the environment. As the chain is executed as a motor program, the agent does not notice the absence of the red cube until it attempts to grasp the cube. At the end of the chain execution, the agent validates the results by comparing the postconditions of the last schema in the chain with the obtained sensory state. Since the red cube was not present in the environment, the final grasp action did not result in the expected observation i.e., holding the cube, causing the chain to be unsuccessful. At this point, the chain was previously executed 5 times successfully and failure on sixth time (at execution step 12) reduced the success rate (C_r) of the chain. However, it is still higher than the threshold i.e., 0.8, therefore the chain, "Reach button", "Press button", "Reach cube" and "Grasp cube", still continues to be executed as a motor program.

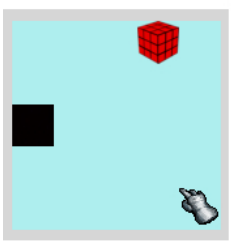
Test III. In the last condition, the red cube is introduced in the environment when reset from the previous execution. As the result, the environment is perceived with two objects in it rather than one, as it does in all 12 previous executions. This new situation in the environment triggers a different excitation for all schemas and chains present in the memory. At this point, the agent is excited to reach directly for the red cube, rather than the button as it has done previously.

The results of this experiment show that the agent learnt schema chains of various length through exploratory behaviours in the environment. The experiment also demonstrated successful use of chains in both modes; reflexive chain and motor program. The experiment shows that a chain is executed as a motor program after previous successful executions. Furthermore, it is also demonstrated that a motor program may be changed back into a reflexive chain mode after failing to acquire the expected results as it causes decrement in the chain success rate C_r, as seen in Test `II`.

6.2.3 Experiment 2: Develop Chains through Exploration

This experiment demonstrates playful behaviour for exploring an environment, discovering action outcomes then creating schema chains to form higher level behaviours. This experiment is also performed in a simulated environment, *Sandbox*. We will use the same environment with different objects. The agent is provided with "Reach", "Grasp" and "Release" actions to interact with the environment. The experiment is then repeated on an iCub humanoid robot to show the application of Dev-PSchema in a real world scenario. This experiment also demonstrates transferability of the system between two different platforms, Sandbox and iCub, without any major changes in the system. Only 10% tolerance was introduced in the similarity calculations of the system to handle noise in the sensory information perceived by the iCub, see Section 4.1 for details.

The experiment contains two stages. In the first stage, we introduced an object (red cube in simulator), and a hole in the environment and let the agent play with it. The hole in the environment is perceived as an object with colour and shape, however, it cannot be interacted with through grasping. The agent will not get any touch perception when it reaches toward it and when attempting to grasp, the hand will close fully to a fist. When an object with a similar shape as the hole is released in the hole, it disappears from the environment. Figure 6.5 shows the environment for the first stage of the experiment and perceived state by the agent.



Sensor = hand, x=5.0, y=5.0
Sensor = visual, colour = "red",
shape = "cube",
x=4.0, y=1.0
Sensor = visual, colour = "black",
shape = "cube",
x=1.0, y=3.0

Fig. 6.5 Left: Simulator environment containing end-effector, hole and an object. Right: description of the sensory state of the environment

During the first stage of this experiment, the agent is allowed to freely play with the objects in the environment. The first stage ends when the agent drops the object in the hole. It is worth mentioning that the aim to drop the object in the hole is decided by us (experimenter), but not specified to the agent. The agent is neither programmed with this aim nor contains any schema to perform this specific action. At the start, the agent only contains the raw actions (Reach, Grasp, Release), without any understanding of the effects the actions will have on either object in the environment. Thus as all the actions provide the same excitation, then the first action is selected from the list of available actions. It should be noted that the agent does not get any reward or penalty for any successful execution. Its behaviours are only based on the internal excitations, thus demonstrate modelling of play behaviours in infants. We expect that during a period of playful exploratory behaviour, the agent will achieve the aim of the experiment.

In the second stage of the experiment, the environment is reset to evaluate the ability of the agent to exploit the knowledge gained during stage 1 and apply chains of higher level actions. We anticipate that the agent will be able to create a chain of four actions (reach for the cube, grasp, reach for hole, release) to pick and drop the object in the hole in a single execution rather than the exploratory play it did in the first stage. It should be noted that the agent is still able to generate and reuse chains during the first stage of the experiment. The excitation parameter weights used in this experiment for the simulator are 0.5, 0.5, 0.6 and 0.4 for ω_1, ω_2, ω_3 and ω_4 respectively, see Chapter 5 for details. We made a slight change in the weights of the ω_3 and ω_4, to encourage the agent to become habituated with schema actions quickly during play and therefore try different schemas, hence different actions.

This experiment was also repeated using the iCub humanoid robot [122]. With iCub, a low-level system is responsible for (i) providing high-level action commands and (ii) preparing and maintaining visual, proprioceptive and tactile perceptions. The only change we made to the Dev-PSchema is to add a small tolerance (10%) of similarity to account for some variation from the robot sensors, see Section 4.1 for details. The expected sensory state

information is undefined, enabling the system to respond to new and previously unknown states or actions that may become available from the low-level system. This, therefore, shows the ability of Dev-PSchema to be applied to different and more complex settings.

The agent receives high-level sensory information from the low-level sensory-motor control (SMC) system. iCub vision and motor control systems are discussed in details in [93]. A high-level diagram of the overall system is provided in Appendix A. Different high-level actions are available for the robot to interact with the object in the environment, enabling Dev-PSchema to send high-level action commands to the SMC, which are then translated it into low-level motor commands. In terms of actions, the reach, grasp and release commands are available after they are learnt in a developmental approach as documented in previous research (Law et al. [103], Shaw et al. [171, 170], Lewkowicz et al. [107]). Reaching in iCub is learnt by employing an approach that is inspired from child hand regard in infancy [157]. This learning approach consists of random arm movements that trigger eye saccades on the visually stimulating hands. Once fixated, mappings are learned between the reaching space and the visual space, i.e., the gaze space of the robot [62]. Further information regarding the iCub's sensory and motor systems is provided in [62, 107, 103, 171, 170, 47].

At the beginning of the experiment, the robot is given time to visually explore its intermediate space by performing saccades to stimulating targets. For the needs of this experiment, green and red patches on the retina visually attract the robot's attention. Gaze coordinates are reported based on the combined proprioceptive eye and head positions for the fixation. The gaze coordinates act as the equivalent of the world coordinates in the Sandbox simulator. Subsequently, all colour information that is found within the foveal area of the retina (i.e., the circular region depicted in Figure 6.6), are grouped as part of the same visual perception. The rationality behind this grouping is that at this stage, visual targets that are found in the fovea are considered to be part of the same object in the world. Along with the HSV colour model values (i.e., Hue, Saturation and Value), the size of the colour patch is also calculated followed by the fixation target's depth. The low-level feature extraction mechanism employed

in this experiment is discussed in [62]. Although the gaze space is two-dimensional, an estimation of the depth of the fixation is measured for the reach system. Depth is calculated after the eyes converge or diverge, depending upon the object position, to focus on the same object.

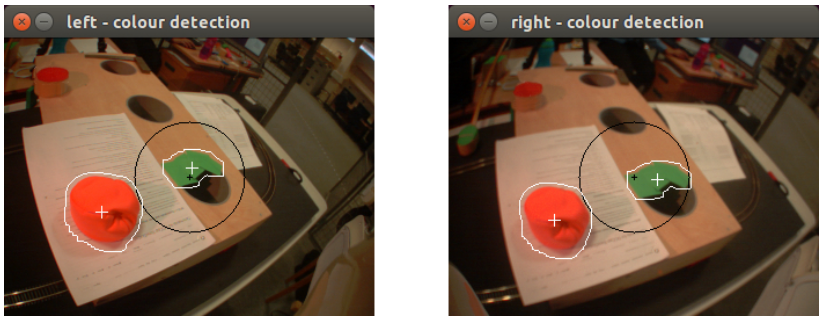


Fig. 6.6 Perceived colour patches by the iCub from the left and the right eye.

As with the visual perceptions, tactile information is analysed by the low-level system, in order to prepare tactile perceptions for Dev-PSchema. A tactile perception consists of the touching hand identification as well as the areas that received tactile information on it (i.e., the 5 fingertips and the palm). Finally, proprioception perceptions are sent for each hand of the robot. They consist of the position of the hands in the gaze space and value related to the current hand grip. The latter reflects the hand's open and close configuration in percentage (i.e., 0% fully open, 100% fully closed).

Unlike the Sandbox simulator, where the world state is provided by the software, visual changes in the real world cannot be fully captured unless the robot visually revisits the areas of interest. Notice that previously generated visual perceptions may no longer be available due to several real-life phenomena. For instance, an object is perceived differently while it is partially or fully hidden from the eye cameras while the arms move within the reach space, or when the object has moved while an action is performed. Bearing in mind that not all the visual perceptions are found in the retina at all times, meaning that substantial head movement may be required in order to update their information, the robot needs a way to update the world state perceptions after each action. To tackle this practical issue, the

low-level system keeps a short term memory of the gaze targets with which it previously engaged, and iterates through them at the end of every action. Hence, having access to up-to-date world state perceptions and actions, the associated Dev-PSchema mechanisms can efficiently operate.

sensor = colour, hue = 3.84, saturation = 88.27, value = 76.24, area = 70.0, gaze_x = 34.06, gaze_y = -51.81

sensor = colour, hue = 22.45, saturation = 56.81, value = 43.84, area = 25.79, gaze_x = 45.46, gaze_y = -50.98

sensor = proprioceptive, grip = 100.0, hand = 4.0, gaze_x = 44.01, gaze_y = -49.62

Fig. 6.7 Experimental set up for the iCub & perceived sensory state.

The experimental set up used for this experiment is seen in Figure 6.7. A red soft toy is placed on a wooden board that contains a hole, big enough to ensure a successful drop. The hole is marked with a green colour tape in order to be visible to the robot. Thus, visual perceptions of both targets are sent to Dev-PSchema containing their coordinates in the gaze space. In order to match the simulator's experiment, one robotic arm is utilised, limiting the amount of proprioceptive and tactile perceptions to the right hand only. To speed up the experiment, the robot is only allowed to saccade to the given target positions. Figure 6.6 shows how targets are perceived by the eye-cameras from the environment. Using the iterative mechanism mentioned above, the visual perceptions of both the red and green targets are updated to constitute a fresh world state for Dev-PSchema's post-condition matching and excitation calculations.

For the experiment with iCub we used a value of 0.5 for all the parameter weights (ω_1, ω_2, ω_3 and ω_4). Equal weights (0.5) for the similarity and novelty/habituation pair will encourage the agent to interact with the less habituated and more novel object, having the

same similarity. For ω_3 and ω_4, this encourages the agent to switch objects and schemas, hence actions, frequently. Values from iCub perceptions were all normalised between $0 \sim 1$ with 10% tolerance to account for noise from the raw sensors.

6.2.4 Experiment 2 Results

The results for this experiment with the simulator and iCub robot are presented below.

6.2.4.1 Results: Simulator

During the first stage of the experiment, the agent playfully explored the two objects through available actions in the environment. As new experiences were gained, new schemas describing these were formed. These new schemas had high novelty and were therefore often selected as the next action, resulting in a playful behaviour that repeats interesting actions, thereby also confirming their effects. Initially, the agent focused its attention on the cube, learning the effects of reaching, grasping and releasing it. These actions were then combined into various chains that were tested before the attention switched to the hole. At this point, it was still holding the object, which it discovered moved with its hand. Attempts to grasp the hole made no difference, allowing the release action to become the most excited again, and finally dropping the object in the hole.

Figure 6.8 shows the excitations of different schemas and chains created during the playful behaviour, showing that the agent played with the object using different actions. Before each action execution, the agent calculates the excitation of all the actions (schemas) and schema chains. Either a chain or a schema action with the highest excitation is executed in the environment.

The figure shows the winning action at each execution in the experiment. During the play, the agent also created some chains and executed them to obtain interesting results from the combination of actions in the chains. The continuous lines in the figure show the excitations

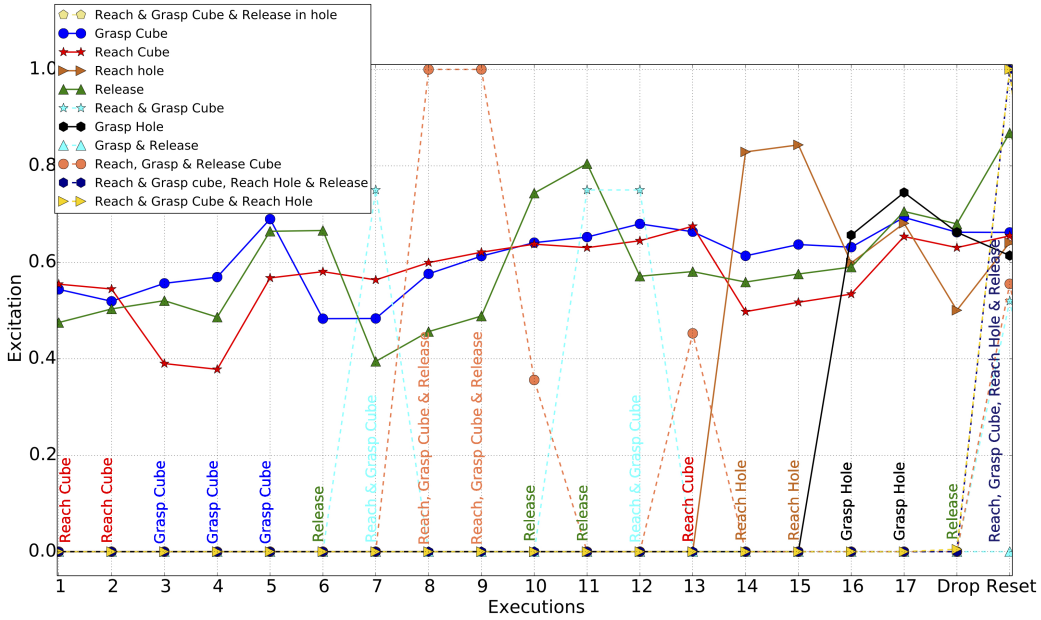


Fig. 6.8 Schema and chain excitations (Simulator). The most excited schema/chain at each execution is specified across the bottom.

of the schemas and the dotted lines represent chain excitations. Initially, there are no chains available for the agent. Once the agent performs the grasp action, it created the "Reach & Grasp chain" and executed this at the 7^{th} execution. Similarly, once the agent released the object, it discovered the "Reach, Grasp & Release" chain. The chain was then executed twice as it had the highest excitation at the 8^{th} and 9^{th} execution.

At the 18^{th} execution, the agent drops the cube in the hole after moving it towards the hole position. With this drop, the agent finished the first stage of the experiment, the discovery of dropping an object in the hole. Once the agent reached the first stage aim, we reset the environment, for the second stage of the experiment, by placing the object back at the same position as shown in Figure 6.5, and the hand back to its starting position. At this point, the agent already had experience of dropping the object in the hole, so this stage evaluates the agent's ability to reuse that knowledge. Through the excitation calculation mechanism, the agent created the 4-schema chain "Reach, Grasp Cube, reach Hole and Release" following stage 1. It calculated the excitations of all the schemas and the chains, and this 4-schema

chain (dropping the cube in the hole) was found to be most excited. This is due to it being a new chain and also making the highest difference within the environment. Execution 19 ("Reset" on X-axis) in Figure 6.8 shows the excitations of all the schemas and chains for the given environment.

Figure 6.8 shows that at the final execution, the excitations for all the schemas were less than the 4-schema chain. However, two other 3-schema chains i.e. "Reach, Grasp Cube & Reach Hole" and "Reach, Grasp Cube & Release at Hole" have the same excitation as the 4-schema chain. The agent, in this condition, picks the longest chain (4-schema chain) to execute, to bring more changes in the environment. During the chain execution, the agent checks the sensory feedback to confirm if it is getting the expected outcome at the end of the action in the 4-actions (schemas) chain. Thus the chain is executed in the "Reflex Chain" mode here.

6.2.4.2 Results: iCub

The experiment starts with the robotic arm at what we refer to as the home position. Having the arm raised next to the head and thus outside the robot's visual field, it is ensured that the initial acquired visual perceptions reflect a world state of inactivity. Figure 6.6 shows the experimental set up for the iCub robot, perceived through the robot's vision system. In the beginning, both targets, the object and the hole, are equally exciting for the robot, therefore, it initially selects to reach towards the hole target. Grasp happens to be the next exciting action to be performed, and due to the perception changes at both visual and proprioceptive levels, new schemas are generated. These new experiences are repeated followed by a release action, an order which leads to the creation of a schema chain in the system; "Grasp→Release". The related excitations are depicted at the Y-axis of Figure 6.9, whereas the X-axis shows the order of schema (i.e., action) execution.

After a number of executions related to the hole target, habituation occurs, therefore, the robot reaches towards the ball (10^{th} execution). After a successful grasp action, the

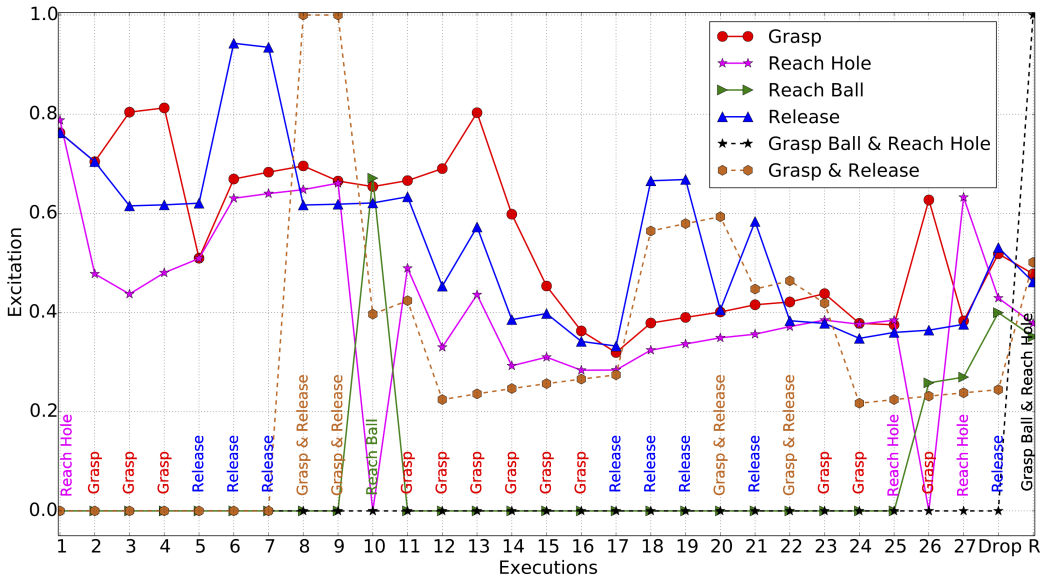


Fig. 6.9 Schema and Chain excitations for iCub. The most excited schema/chain at each execution is specified across the bottom.

world state is updated with the red ball to be ultimately perceived differently due to the grasping hand partially covering it. Subsequently, the sudden change to the visual perceptions offers a lot of new stimulation, fostering the creation of new schemas. As a result, grasping again becomes the most exciting action to perform, while holding the object. This repeating behaviour is akin to squeezing an object, which in turn results in several changes in visual as well as proprioceptive perceptions. However, after a number of grasp actions, the system habituates and a release is selected for the 17th execution.

Once released, the object drops on the wooden board again giving different visual perceptions. A new post-release schema that reflects the new world state is learnt for iCub to repeat, and after a few executions, it ultimately utilised the "Grasp→Release" chain to interact with the object. It is then observed that the robot moves its arm to the hole coordinates while holding the ball at the 25th execution, followed by a release command being issued at the 28th execution which causes a successful drop of the ball into the hole, finishing the first stage of the experiment. As the purpose of this experiment is to demonstrate learning of the schema chains in Dev-PSchema, rather than finding how and when the chains are developed.

Therefore we proceed to the second stage of the experiment after completing the first stage, without repeating the first stage.

For the second stage of the experiment, the robot is expected to utilise the previously learnt schemas and schema chains in order to express a similar playing behaviour. Thus, without specifying a particular objective state the aim is to evaluate the ability of the system to link past experiences and actions from its repertoire with the environment and to succeed in dropping the ball into the hole. In contrast to the simulator, the robot's performance differs in this stage. The amount of noise in the real world is found to play an important role in delaying the process of appropriate schema selection for execution. The significant variation between schemas makes it difficult for the robot to directly link between them to create a chain for dropping the object in the hole. However, it is anticipated that with generalisation over the variation in perceptions, the generation of a full chain for dropping the ball in the hole would be possible given sufficient time for exploration. Nevertheless, subsets of the desired full chain are generated and repeated by the system, such as the "Grasp→Reach" and "Grasp→Release" chains.

6.3 Discussions & Conclusions

Both experiments demonstrated that Dev-PSchema is able to develop high-level actions itself through a chain of schemas, rather than providing instructions or observations. The agents use high-level actions as schema chains to achieve a high-level objective starting from a sensory state in the environment. The agents demonstrate re-use of the schemas chains in similar situations. From psychology, we have evidence that humans use previously generated movements in similar situations. Kent et al. [87] found that action movements from previous tasks influence the current reach to grasp task. The subjects, children and adults, were initially asked to grasp a rotating object with fingers on a particular side. In the test condition, experimenters found that the subjects used previously generated movements

in the current task. However, a group of subjects who did not perform the initial task chose the most efficient way to achieve the target grasp. Thus it can be concluded that the subjects who performed the task initially re-used previously generated motor plans due to the cost of change in action planning. In another similar study, researchers found that previously generated motor plans were considered for use in later trials but not just for the immediately following trial [43]. Dixon et al. [43] found the subjects considered multiple previously generated action plans, taking into consideration context as well as contextual similarity to the current situations in the environment.

The concept of action chains and motor programs is also inspired from developmental psychology, such as the work by Lashley [98] investigating the hierarchical organisation of behavioural plans. He believed that the concept of a motor program was being ignored over the concept of reflexive chains. The theory of reflexive chains proposes the serial order of behaviours with sensory feedback, which contributes to the excitation for each of the sequential building blocks of the chain, as discussed in the beginning of this Chapter. On the other hand, the motor program theory proposes the serial order of the actions in the behaviour where the sensory feedback is ignored. The agent demonstrated the conversion of a reflexive chain into a motor program after 4 successful executions. In a motor program, actions are executed continuously, one after the other, without any sensory feedback. The motor programs in Dev-PSchema are modelled on the sequential behaviours discussed in [160, 98]. According to Lashley [98] and Rosenbaum et al. [160] sequential actions are composed of sequential actions as hierarchical behaviours. Repetitions in sequential behaviours make them a singular unit, without any break in between underlying actions for sensory feedback.

Lashley believed that spending more time at the beginning of sequential actions than the time in between the behavioural elements and errors support that actions sequences are hierarchically organised plans for behaviours, hence the theory of the motor program. He believed that the longer time spent at the beginning of sequential actions provides an evidence for planning the entire sequence and leading to a shorter gap between the behavioural elements,

which is not enough to receive feedback and plan the following step. More recently, Lashley's work has been reviewed by [160]. This review suggests the identification of key-frames in sequential behaviours. The behaviours between these key-frames could be considered as short chains being executed as motor programs, thus considering motor programs as a bunch of actions between two key frames in a sequential behaviour. The authors also reported that the execution time for actions between key-frames is significantly reduced following 4 to 6 repetitions in sequential behaviours. Similarly, authors in [87, 85, 53, 193] found a decline in response time as a sequence is repeatedly performed. However, accuracy decreases after 15 executions [85]. The error in a sequential behaviour is related to the behavioural planning at the start of the behaviour [98].

In the first experiment, the agent demonstrated the execution of a motor program, a chain converted from the reflexive chain after the successful executions in the environment. The agent executed the chain of actions without confirming any peripheral states of the environment between the actions. Thus, the chains executed in tests conditions of the experiment can be considered to have acted as a motor program. To make a motor program from a schema chain, the system not only relies upon the number of times the schema chain has been successful but also upon the success rate. For a schema chain to be a motor program it must be successful at least four times, with a success rate of more than 80%. Furthermore, developmental psychology supports thoughts of key frames containing a few actions in sequential behaviours thus supporting the limit in chain length [160]. In Dev-PSchema, the generated chains are limited to a maximum length of 4 action schemas.

The second experiment, presented in Section 6.2.3, has demonstrated the play behaviour of the agent in the environment and examined the potential effects of the actions on different objects. The agent was able to create a new schema while grasping the ball in the simulator, and multiple different grasp schemas were learned by the iCub due to changes in perception and the environment. For both the simulator and iCub, the agents did not create any new schemas for grasping the hole as this does not make any change in the environment. This

behaviour shows that the agent is capable of learning the effects of its actions on different objects. Thus the agent learns the behaviours with objects through exploration. Furthermore, the agent reuses learnt schemas during the exploitation stage. This stage reflects the sensori-motor stage of Piaget's theory [147], where infants are described as re-using or repeating their learnt behaviours involving their bodies on the interesting objects.

The second experiment also demonstrated the capability of the system to be integrated with different platforms, transferred from the simulator to the iCub robot in a laboratory environment, without making any major changes to the system. In both experiments, we demonstrated that the agent shows playful and exploratory behaviours. While Dev-PSchema also enables the agent to simulate different individuals with different preferences while weights of the excitation parameters remained constant during the experiment in this chapter.

Furthermore, in the Dev-PSchema system, the excitation of each schema decreases with each execution if a similar environmental state is observed repeatedly. Thus similar situations in the environment become less exciting to interact with. This is what can be seen in human infants. Infants prefer to see a novel situation and pay attention towards them. They get bored with similar situations and pay less attention to it [177]. However, new situations in the environment can trigger previously used schemas. We can see in Figure 6.4 (execution 13) that "Reach" action for cube was found excited in Test `II` condition. Although this schema had previously been used, independently and in the schema chain, the new state in the environment triggered it again.

We demonstrated that the Dev-PSchema system provides the capability of creating schema chains from previous experiences, schemas, by finding links between pre and post conditions of different schemas. Schema chains are then compared with the other chains and the schemas in the memory to find the most excited for a given sensory state of the environment. In PSchema, the system was only able to find a schema chain when a target sensory state was defined [172]. Our system is able to find chains automatically by finding

links between the current sensory state of the environment and schema context. We also developed the system to consider chains as motor programs, which are found successful repeatedly. In the case of the "reflex chain" mode, at each execution in the chain, the perceived state of the environment is tested and compared with expected sub-target (post-conditions of last executed schema). A failure in the chain will not decrease successes of the chain but it will decrease the success rate of that chain. Thus a motor program can be changed back as "reflex chain" if it fails to achieve the objective state repeatedly. With these developments in Dev-PSchema, it is able to extend the play mechanism by finding an objective which can be achieved through a sequence of actions rather than just a single action. The high-level objective, postconditions of highly excited schema or postconditions of the last schema in the highly excited chain, is set by the agent itself through the excitation mechanism. This sequence of actions, schema chain, can be defined as a high-level action which is achieved through different actions.

Chapter 7

Shaping Learning

Previously, in Chapters 4 to 6, we demonstrated knowledge development in the artificial agents through intrinsically motivated active explorations. Apart from, developing object-action pair knowledge in the shape of schemas, the agents demonstrated learning high-level skills (behaviours) through schema chains. In this chapter we demonstrate the capability of Dev-PSchema to solve user defined problems, described as a set of perceptions by an external agent, using schema chains. This capability can be used to test the learning developed by the agent through explorations. The problem solving capability of the agent provides an opportunity for an external user to shape the agent's knowledge. The agent is provided with an opportunity to learn skills, increasing in complexity, and use the skills to solve a problem provided by the external agent.

To demonstrate the problem solving capability in the agent, we performed two different experiments. The first experiment demonstrates how the agent interacts with the external agent to solve the problem. Furthermore, this experiment also demonstrates how a user can shape the agent's knowledge. This experiment also demonstrates the capability of the agent to develop different solutions for a single problem. In the second experiment, we demonstrate the agent's learning capability to develop associations between objects and the performed action. The learning is tested with the experiment to recreate a given state in the environment. In Section 7.1 we describe the problem solving mechanism in the agent. The experiments and

results are discussed in Section 7.2 and finally, the discussion and conclusion are provided in Section 7.3.

7.1 Problem Solving-Interaction

In robotic applications, the robots often do not need to learn but instead solve problems provided by an external agent, either a human, another robot, or by itself. In Chapter 6 we demonstrated that the agent is capable of forming chains to make high-level actions to achieve a more distant state, which is a postcondition of one of the schemas in the memory. The agent finds the series of actions that lead to the objective state from a perceived state. When an external agent specifies the desired state this could be considered as shaping the knowledge. The mechanism and design of the Dev-PSchema system limit its knowledge development to that gained through active explorations only, however, it is capable of abstract interaction with an external agent. This interaction is one-way, from an experimenter to the agent, through the perceptual description, as shown in Section 3.2.

Section 3.1.3 briefly describes the mechanism for problem solving in the Dev-PSchema enabled agent. The agent is provided with an objective state by an external agent, as a descriptive sensory state similar to the sensory state presented in schema preconditions/postconditions. The agent finds chains that lead to the objective state from the currently perceived sensory state (WS). The agent may end up with multiple solutions, chains, that achieve the objective. At this point, the agent requires external feedback to select the solution to be executed. The agent provides all the solutions with their estimated probabilities of success, referred to as excitation, to the external agent.

Algorithm 21 describes the mechanism for problem solving when provided with the current state of the environment and the target state. The mechanism finds all the possible solutions through the chaining mechanism provided in Algorithm 18, with the active problem

solving flag. This flag lets the agent calculate the probability of success of the found chain to find suitable solutions for the problem. If the agent finds more than one solution, it hands over the control to the external agent, the user, to select the suitable solution. The user selected solution is executed either as Reflex chain or motor program depending upon the selected chain, see Chapter 6 for details. Once the agent completes the whole sequence of actions, it confirms the success by comparing the latest perceived state with the objective state. This affects the excitation in problem solving applications. Although the external agent does not provide the feedback regarding the success in solving the problem, the agent itself decides the success of the solution by comparing the objective state with the perceived state at the end of the behaviour. This enables the agent to shape its learning further from the experience it gained during the problem solving.

Algorithm 21 Problem solving mode

```
1:  function solve_problem (WorldState WS, WorldState Target)
2:      chains = find_path(WS, Target, True)      ▷ Problem solving flag is set True for Algorithm 18
3:      if length(chains) > 0 then                                          ▷ If any solution exists
4:          if length(chains) > 1 then                                ▷ If more than one solutions found
5:              Get user selection for the solution to be executed
6:              execute_chain(user selected chain)                          ▷ See Algorithm 20
7:          else
8:              execute_chain (chains)                                      ▷ See Algorithm 20
9:          end if
10:         update_world_state                                  ▷ See Chapter 3, Section 3.5
11:     else
12:         Return                                                          ▷ No solution found
13:     end if
14: end function
```

In the following experiment, we demonstrate that the agent learns throughout their behavioural experiences, selected either independently during exploration or by the external agent during problem solving. The problem solving mechanism in Dev-PSchema is adapted from the PSchema system [172]. In PSchema, the agent finds a solution for a problem defined by an external agent, followed by its execution. However, in Dev-PSchema, the agent responds with more than one solutions, wherever possible, and waits for the external agent to select one of the solutions to execute. This may be seen as help from the external agent to shape the acquired knowledge of the agent through guiding in problem solving. Furthermore, in PSchema a chain probability is calculated through success probability of

the individual schemas in the chain. Whereas, Dev-PSchema calculates chain probability through a weighted combination of schema probabilities and the similarity between first schema preconditions in the chain and current state of the environment. This leads the agent to find more relevant and successful solutions for the given problem.

We would like to acknowledge that parts of this chapter have been included in a paper, given below, which is currently under internal review.

- Kumar S., Shaw P., Giagkos A., Braud R., Lee M.H., Shen Q. Learning affordances with action sequences and shaping knowledge through problem solving.

7.2 Experiments & Results

We performed two different experiments to test chain building and learning associations for tool use through exploratory play. For the experiments, we interfaced Dev-PSchema with the iCub humanoid robot [122] as in experiment 2 from Chapter 6.iCub vision and motor control systems are discussed in details in Chapter 6. The two experiments demonstrate the learning capability of the agent through exploration by solving user defined problems. Figure 7.1 shows the robot, iCub, in the lab environment, used in the experiments, with some objects.

To encourage the agent to demonstrate exploratory behaviours with the provided objects, we set the excitation parameters ω_3 and ω_4 to 0.3 and 0.7 respectively. As the experiment is related to the actions of the agent rather than the objects on which the actions are performed, we kept ω_1 and ω_2 as 0.5 for each. This enables the excitation to be more dependent on the actions performed rather than the perceived objects to explore them. Although the agent is able to develop knowledge through exploratory play, we simplified the scenarios to demonstrate the concept of shaping knowledge and problem solving in Dev-PSchema.

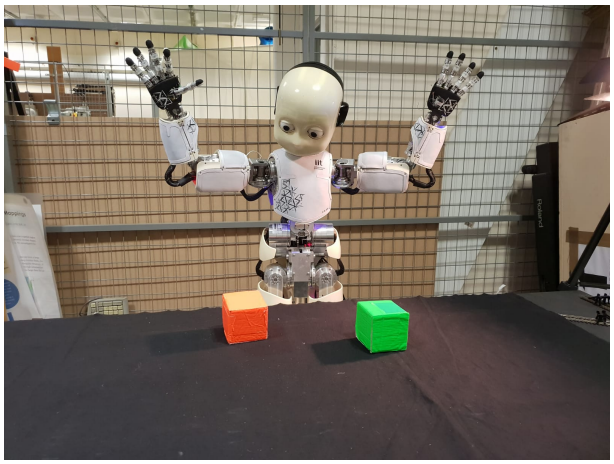
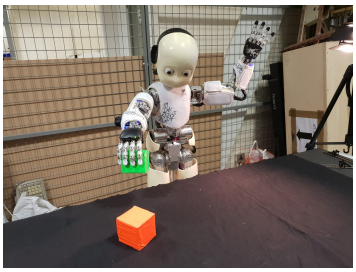


Fig. 7.1 The robot in the lab environment with some objects.

7.2.1 Experiment 1: Problem Solving with Experiences

This experiment is divided into four parts. Each part is related to learning a particular schema in the environment and developing a chain as a high-level action related to it. Each part of this experiment consists of two stages. In the first stage, iCub performs play behaviour in the environment. In the second stage, we test the learning by defining an objective state to achieve using learnt high-level actions. As the agent is not equipped with a natural language processing system (which is considered to be irrelevant here), the user defined objective is provided as a sensory state, similar to that used in schemas, as shown in Figure 7.2.



Sensor= proprio grip= 0.5 hand= 4 gaze_x= -44.54 gaze_y= -50.66

Sensor= colour feature_id= 13 proto_id= 2 gaze_x= -44.54 gaze_y= -50.66
Sensor= brightness feature_id= 45 proto_id= 2 gaze_x= -44.54 gaze_y= -50.66
Sensor= brightness feature_id= 1 proto_id= 2 gaze_x= -44.54 gaze_y= -50.66

Sensor= colour feature_id= 32 proto_id= 5 gaze_x= -15.41 gaze_y= -55.87
Sensor= brightness feature_id= 9 proto_id= 5 gaze_x= -15.41 gaze_y= -55.87
Sensor= colour feature_id= 12 proto_id= 5 gaze_x= -15.41 gaze_y= -55.87

Fig. 7.2 (Left) iCub, in the lab environment, performing “Reach” action and some objects in the environment. (Right) An instance of sensory information of a perceived environment passed by SMC towards Dev-PSchema.

In this 4 part experiment, we are demonstrating the scaffolding of knowledge and finding alternative solutions for user-defined problems. Each part of this experiment is related to a skill, learnt through exploration and play. Each such skill is described below:

Part I. Hold action

We refer to a chain of reach and grasp action schemas as a *Hold* action. This action is learnt by combining the reach and grasp actions having perceptual similarity between the preconditions and postconditions in the two different schemas respectively.

Part II. Transport action

We call a chain of Reach, Grasp and Reach actions as the *Transport* action. This action represents moving the hand towards another position while holding an object. This action is learnt by adding a reach action to the previously learnt *Hold* action. However, the second reach in the chain is towards a different position.

Part III. Move action

A *Move* action in this experiment involves a chain of reach, grasp, reach (different positions) and release actions. This action represents transporting an object from one position to another. The action is developed by the addition of a release action to the previously learnt *Transport* action.

Part IV. Displace action

We refer to a schema chain of reach and push actions as a *Displace* action. This action represents displacing an object from one position to another.

The agent, iCub, is pre-loaded with the bootstrap schemas. Bootstrap schemas and the mechanism to develop higher level schemas from the bootstrap schemas is described

in [95, 94, 93] and Chapter 3.

7.2.2 Results: Experiment 1

At each part of the experiment, we provided the agent with the relevant bootstrap schema(s) to develop the target skill i.e., chain. We let the agent play with two different objects, labelled *A* and *B*, to learn generalised skills, which can be applied to novel objects with a perceptual similarity. Once the agent develops the generalised schemas, we introduce a different object, labelled *C*, in the environment to test the learnt skill. We provide a problem related to the learnt skill for the agent to solve during the application of the generalised schema to the novel object. The agent responds with the possible solutions for the user-defined problem and executes the solution selected by the user. It should be noted that the "encouraged chains" flag is to make chains compete for excitation with other schemas in the memory. We discuss each part of the experiment that leads to the development of a skill and its test condition below:

7.2.2.1 Hold action

To develop this skill we let the agent play with two different objects *A* and *B* to learn generalised reach and grasp schemas, which are then combined together to form a generalised reach and grasp chain. Once the iCub reaches and grasps the first introduced object, we remove it and introduce the other object to reach and grasp. It should be noted that the position of the two objects may be different, leading the generalised positions, see Chapter 4 for details. As iCub reaches and grasps the second object the generalised mechanism activates, as the generalisation threshold is met (see Chapter 4), leading to the development of generalised reach and grasp schemas. With the generalised reach and grasp, the Part 1*a* of the experiment ends the ingredients for the *Hold* action are in place. Figure 7.3 shows the actions performed and their excitations during Part 1*a* of this experiment.

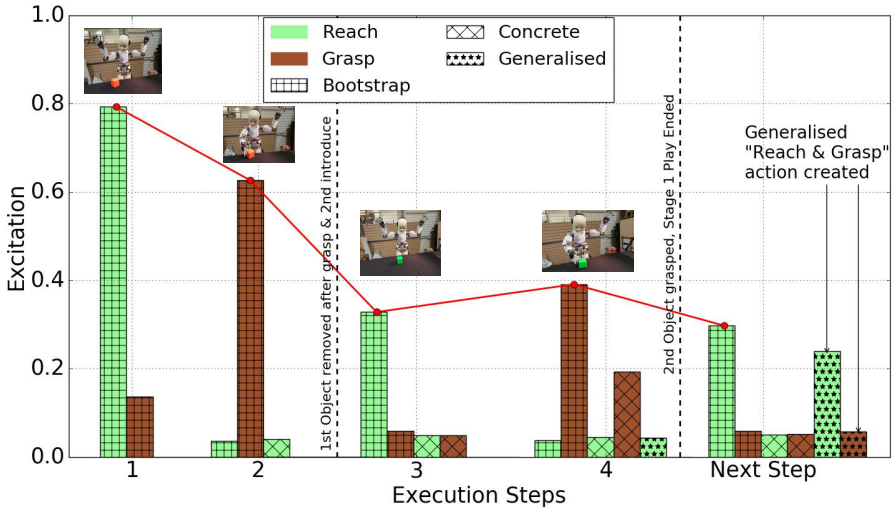


Fig. 7.3 Play behaviour to develop generalised "Reach $" and generalised "Grasp $".

Note, all the action types i.e., bootstrap, concrete and generalised, are combined together and the highest excitation of that action type is plotted in Figures 7.3, 7.5, 7.8 and 7.10. Also, fill patterns are used to represent the type of schemas and colours represent the actions in the figures. Either a schema or chain with the highest excitation is executed using the winner takes all algorithm. The "Next Step" provides the excitations following the last play action. The play is stopped at this point so the excitations and the step are ignored for execution.

It should be noted that every time the iCub finishes the desired action (set by the user but not defined to the iCub), its hands are reset to the home position and out of the visual field. Once the agent creates the generalised reach and grasp schemas, we introduce the object *C* in the environment. After updating the memory with perceptions of object *C*, an objective was provided to "Hold" the novel object. The high-level objective was described as the "Hand" at the novel object position and with the proprioceptive grip of 50%. Figure 7.4 shows the chains with their excitations suggested by the iCub to achieve the target objective state.

From Figure 7.4 it is evident that iCub suggested three different chains to achieve the objective state, "Hold" with object *C*. Each chain has its own excitation based on the schemas successes, similarity to the current environment and length of the chain (see Chapter 6 for a

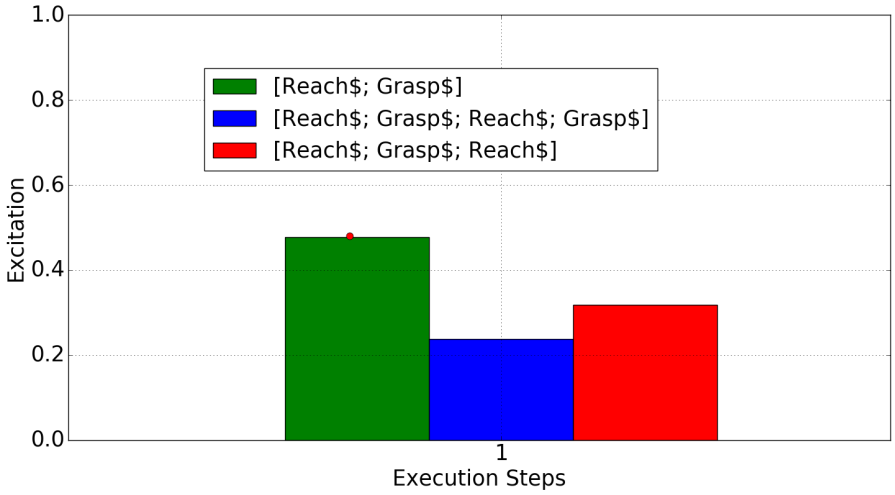


Fig. 7.4 Chains suggested by the agent and their excitations to grasp an object when the hand is not starting at the object position.

chain excitation). It should be noted that the second and third chains containing reach after the grasp action in the sequence are a reach towards the same position. iCub is able to reach for different positions, in this case, its second reach is towards the same position. This is due to over-generalisation in the reach and grasp schemas and greedy chaining algorithm to find longer chains up to the length limit. The highest excited chain to achieve the objective, "Hold" object *C*, contains just one set of the generalised reach and grasp actions. We selected the highest excited chain, which was successfully executed to grasp object *C* and leading the chain to be included in the memory.

7.2.2.2 Transport action

To learn this action, the agent needs to learn moving its hand towards a different position while holding an object. This needs an additional reach action to the "touch" as the agent has already learnt to reach and grasp objects. We introduced object *A* in the environment, which is reached and grasped. In the previous stage, the agent offered a "Reach, Grasp and Reach" chain as a solution to the "Hold" problem, where the final reach was to the same

position as the first object. The agent needs to learn a new generalised schema containing the object and the iCub hand position in the postconditions. The excitation mechanism allows reaching towards a random position when the excitation of all the schema gets below a certain level, in order to speed up the experiment we introduce the second object (object *B*) in the environment right after the first object is grasped. The reach action towards the new object is most excited, following which it learns a new reach schema that now includes holding an object. Figure 7.5 shows the play behaviour of the iCub to learn components of the *Transport* action.

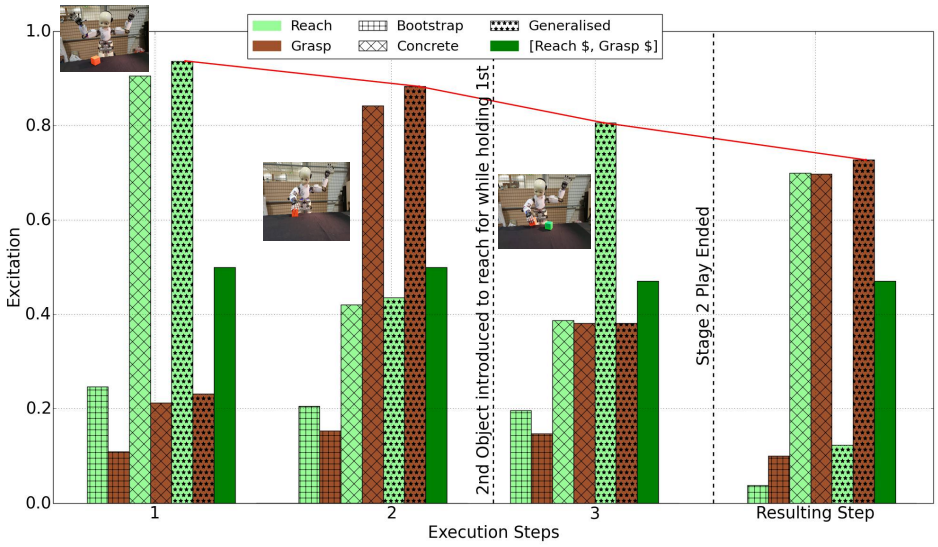


Fig. 7.5 Play behaviour to develop generalised "Reach $" while holding an object.

Once iCub reaches and grasps the new object, iCub sees two objects in the environment at the same position as its hand. It records a new concrete schema which shows the association of moving towards another object while holding an existing object. The iCub also learns a generalised reach schema at this point, combining all previously learnt reach schemas with the newly learnt schema. The new generalised reach schema will help to reach for the object with open hands as well as reach towards a different position while holding an object, as it contains the generalised "grip". Figure 7.6 shows the generalised reach schema developed at

the end of the play for Part 2*a*.

Preconditions	**Action**	**Postconditions**
<Observation=proprioceptive, hand=3.0, grip=$a, x=$c, y=$n> <Observation=colour, value=$g, size=$h, x=$u, y=$m> <Observation=brightness, value=$i, size=$q, x=$u, y=$m>	Reach ($u, $m)	<Observation=proprioceptive, hand=3.0, grip=$w, x=$u, y=$m> <Observation=colour, value=$g, size=$h, x=$u, y=$m> <Observation=brightness, value=$i, size=$q, x=$u, y=$m>
Associated Preconditions		**Associated Postconditions**
<Observation=colour, value=$d, size=$r, x=$c, y=$n> <Observation=edges, value=$s, size=$z, x=$c, y=$n> <Observation=brightness, value=$i, size=$t, x=$c, y=$n>		<Observation=colour, value=$d, size=$r, x=$u, y=$m> <Observation=edges, value=$s, size=$z, x=$u, y=$m> <Observation=brightness, value=$i, size=$t, x=$u, y=$m>

Fig. 7.6 Generalised reach schema generated from the play stage for the “Transport” action.

The schema in Figure 7.6 contains generalised preconditions and postconditions along with the associated observations. The associated observations i.e., associated preconditions and associated postconditions, contain the perceptions from the object in the iCub’s hand. This generalised reach schema is developed from combining all the previously developed reach schemas.

To test the skill developed in this play stage, we introduced object *C* in the environment. Once iCub perceives the object, we provide an objective state for the novel object to be at a different position. Since the system has so far only learnt to reach and grasp actions, it came up with solutions containing reach and grasp actions. Figure 7.7 shows the chains suggested by iCub to achieve the target objective.

From Figure 7.7 it is clear that iCub suggests two different chains to perform the *Transport* action on the novel object. The highest excited chain is the previously successful *Hold* action, however, the chain does not achieve the specified target. We let iCub execute the highest excited chain to perform the objective, *Transport* action. The action ends with grasping the object, rather than moving it to a different position as per the provided objective. In the second try, *Hold* action still the most excited action but is still unsuccessful with respect to the objective. On the third attempt, the excitation of the *Hold* action has now dropped below the desired *Transport* action. On the third run, the iCub executes the sequence of reach, grasp and reach schemas, which achieves the target state.

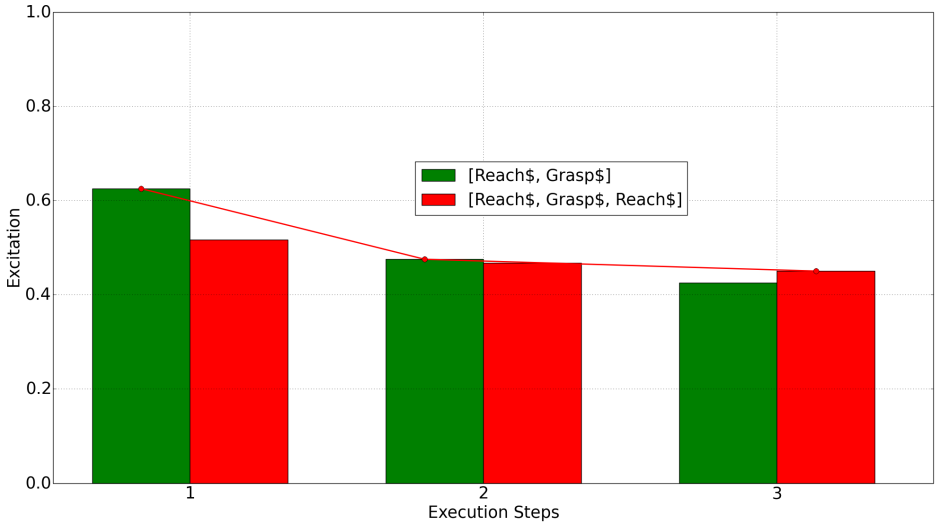


Fig. 7.7 Chains suggested by the agent and their excitations to transport the novel object to a different position while the hand is not at the object position.

7.2.2.3 Move action

We refer to a sequence of reach, grasp, reach (different positions) and release actions as the *Move* action. This action represents transporting an object from one position to another. In *Transport*, iCub has already learnt to move objects from one position to another. However, iCub has not yet learnt a schema to release a grasped object. We introduce a *release* action in the memory, providing an opportunity for the agent to scaffold its knowledge. This action makes iCub's hand open, if it is closed, and provides the grip as 0.0 in the proprioceptive response. We introduced an object in the environment, which the iCub then reaches and grasps. After the grasp action, the bootstrap release action becomes the most excited and it is executed. This sequence of actions, reach, grasp and release, is repeated with another object to get two examples of release actions, required for the generalisation. The new generalised release schema will help to release the novel held object. Figure 7.8 shows the play behaviour of the iCub to learn components of the *Move* action.

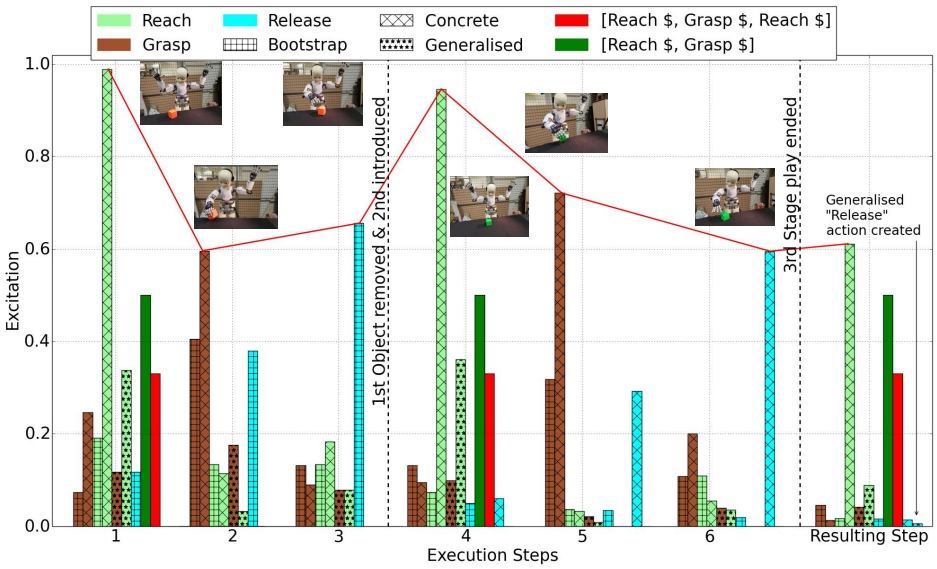


Fig. 7.8 Play behaviour to develop generalised "Release $" while holding and moving an object.

To test this skill, we introduced object *C* in the environment. Once iCub perceived the object we set an objective state, novel object to be at a new specified position, a state similar to the "move" action.

Figure 7.9 shows that the iCub suggests three different chains to perform the *Move* action with the novel object. By the generalisation mechanism, each of the suggested chains potentially achieves the target state. The highest excited chain is the *Move* action. A new chain, "Reach, grasp and release", action is also included in the suggestions. During the play behaviour with the release action when the grasped object was dropped, it bounced slightly and moved away from the hand position. Thus the generalised release schema ended up with the postconditions having the iCub hand and the dropped object at different positions. This caused a new suggested chain with reach, grasp and release actions, where the generalisation does not restrict the range for the coordinates in the postconditions.

As the user-defined objective was to move object *C* to a different (specified) position, all the suggested chains offer possible solutions. The chain excitation system finds the

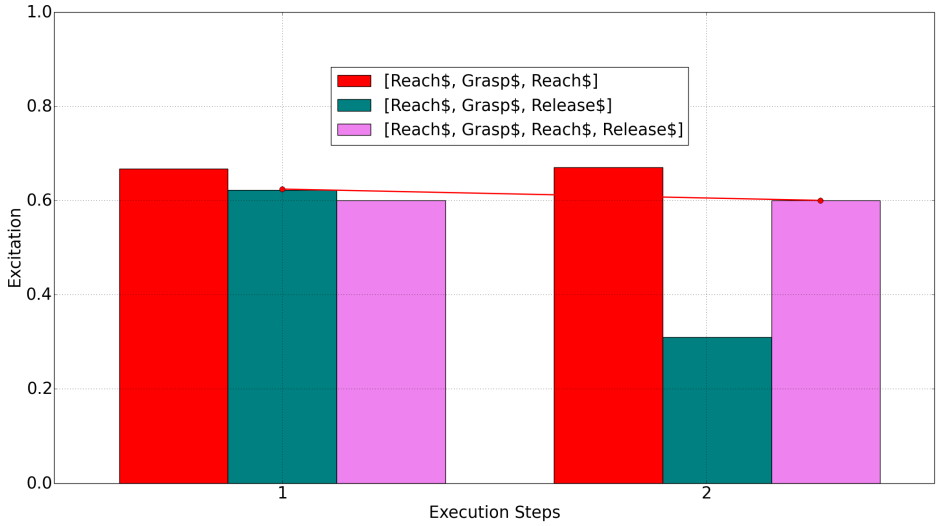


Fig. 7.9 Chains suggested by the agent and their excitations to transport the novel object to a different position while the hand is not at the object position.

Move action as the most excited for the objective. However, we instruct the iCub to run the second highest excited chain, reach, grasp and release, to encourage it to learn and adapt. As the newly defined position of the objective is at a distant position to the current position, releasing the simple object at the original position will not reach the target position causing the excitation mechanism to reduce the excitation of this chain in following executions.

Before the second attempt, we reset the environment and set the same objective. The agent returned the same suggested chains once more with updated excitation values, as seen in Figure 7.9 at execution step 2. In the second execution, we instruct the iCub to run the *Move* action, which it runs successfully and adds this chain into its memory.

7.2.2.4 Displace action

Instead of using reach-grasp, this action makes use of a new "Push" action to move an object from one position to another. A sequence of reach and push action is referred to as *Displace*

action in this experiment. To learn this action, the iCub only needs to learn the push action as it has already learnt the reach action. We introduced a bootstrap "Push" action in memory. This action makes iCub move its hand horizontally towards the body centreline and return to the same position. Thus any object at the hand position will be displaced in the direction of the hand motion.

To construct *Displace* action, iCub has to learn the generalised push schema. We let iCub reach and push objects *A* and *B* one after the other to create two concrete examples of push actions for the generalisation. Figure 7.10 shows the play behaviour of iCub to learn generalised push action. At this point, the agent has all the components for the *Displace* action. To test this skill, we introduced object *C* and provided an objective to make that object appear at a different position.

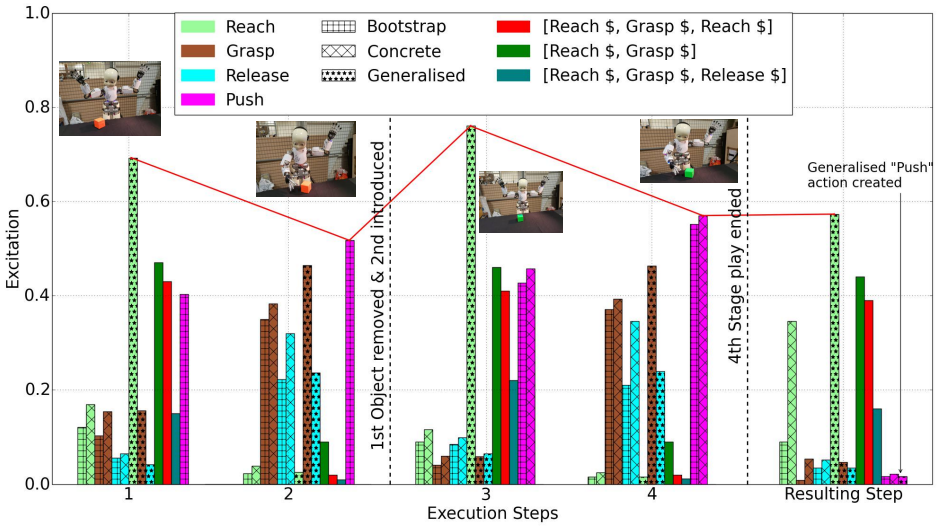


Fig. 7.10 Play behaviour to develop generalised "Push $".

Figure 7.11 shows that iCub suggests four different chains to displace the novel object. As the *Displace* action is new, and the shortest chain, it receives the highest excitation. We let iCub run this new chain which it runs successfully and adds this chain to its memory. It

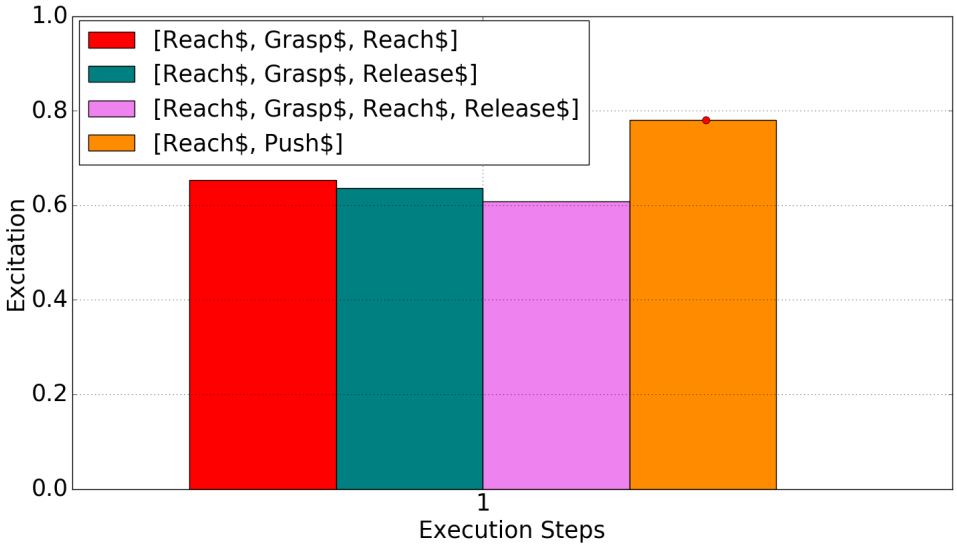


Fig. 7.11 Chains suggested by the system and their excitations to displace the novel object to a different position while the hand is not at the object position.

should be noted that all the possible solutions provided by iCub are relevant to the target state and provide higher probability to achieve it by considering previous experiences demonstrate the ability to find multiple alternative solutions to achieve the same objective state.

During the experiment, iCub created 29 new schemas including 14 concrete and 15 generalised schemas. The agent also added 5 different chains to memory. Figure 7.12 shows the total number of bootstrap, concrete, generalised schemas and chains in the memory at any point of this experiment.

During the experiment, the agent created different concrete and generalised schemas for each type of action. The concrete schemas were created when existing concrete and generalised schemas of the memory do not match with the perceived perceptions of the environment after the action execution. This was caused by variations in the obtained results and object perceptions such as lighting condition causing different perceptions for an object at a different time of the day, noise in the system and complexity of working in the real world. Table

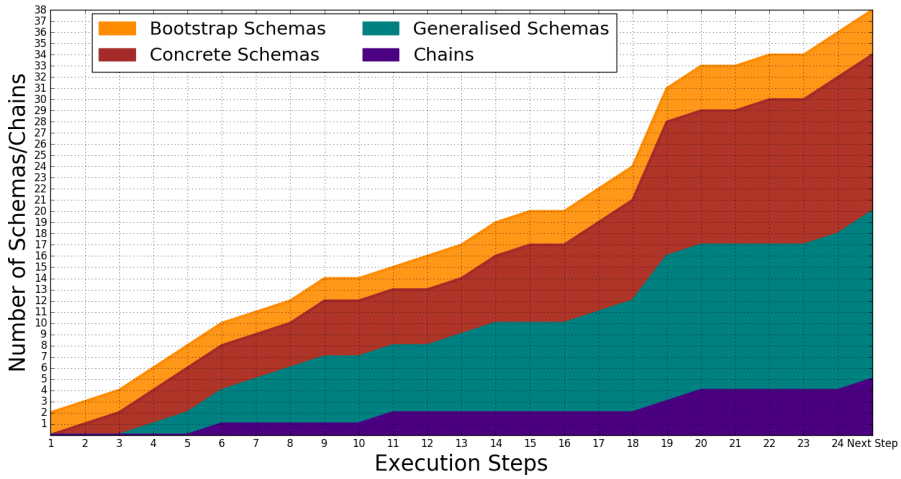


Fig. 7.12 Number of schemas/chains in the memory at each execution step. Lines are stacked up showing the cumulative total number of schemas and chains in the memory following each step.

7.1 shows the number of concrete and generalised schemas created for the each type of action.

Schema action	Concrete	Generalised
Reach	4	5
Grasp	5	6
Release	3	2
Push	2	2

Table 7.1 Total number of concrete and generalised schemas created for each type of action.

Figure 7.13 shows an overview of all the actions, including chains, performed during this experiment at each point of the execution.

7.2.3 Experiment 2: Learning associations - Towards tool-use

This experiment demonstrates the capability of the learning system to learn associations between action and object, and between the objects in the environment. To demonstrate this capability, we performed a tool-use like experiment. The experiment is designed to

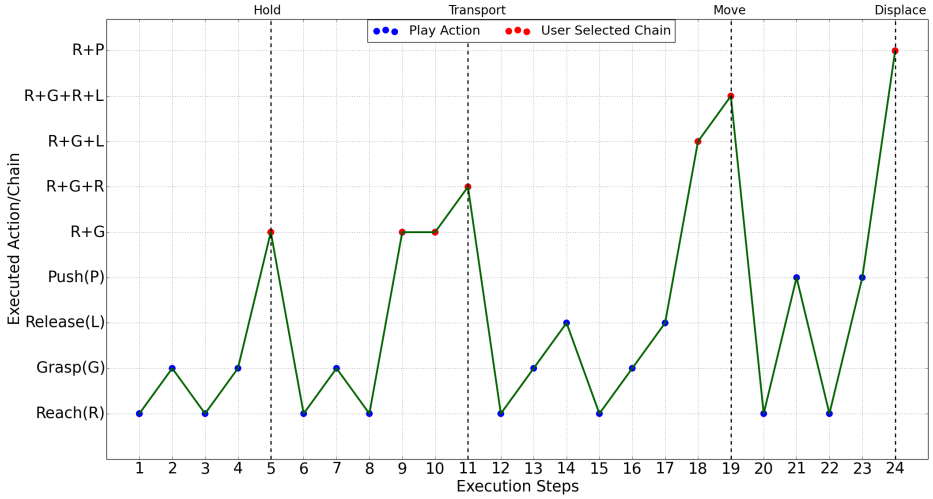


Fig. 7.13 All actions/chains executed during the play.

develop an understating of the associations between the objects effected through an action in the iCub. The agent learns an association when an action is performed on one object, causing a change in the environment other than the object upon which the action is was performed.

The agent is provided with reach and grasp actions only to speed up the experiment, thus providing a set of actions to choose from. We reset the agent with the memories from Part 7.2.2.2 (transport action) of experiment 1 described in Section 7.2.1. This set of memory enables the agent to reach for the target action with small number play actions, limiting the scope of actions for the purpose of this experiment. In the experiment, we introduce an object (a tool) in the environment and let the agent play with it. To learn the associations between the objects we set a trigger position in the environment, causing another object (a toy) to appear in the environment at a dedicated position. The trigger position is set by the user, hence the user decides when to present the toy object. The agent learns the association of moving the tool object towards the trigger position and the new object, the toy, appearing in the environment. Figure 7.14 shows the flow of the experiment for learning the associations and test condition.

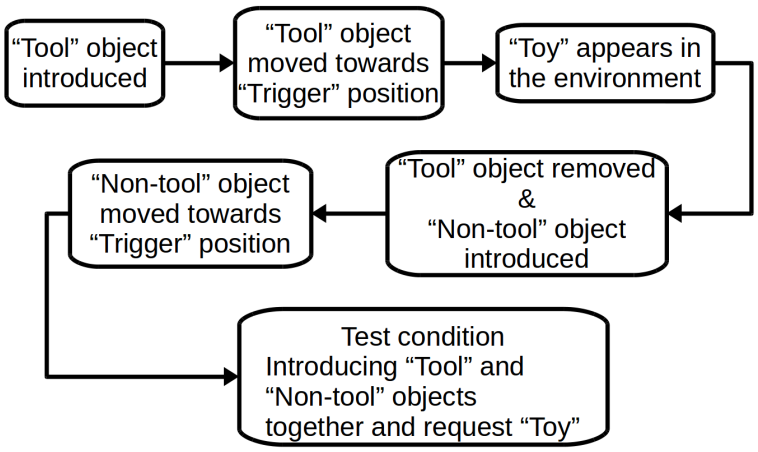


Fig. 7.14 Flow for the experiment to learn "object-object-action" associations.

Following the tool object, we introduced a non-tool object in the environment and let the agent play with it. We expected that the agent will re-use the learnt schema to re-introduce the toy in the environment. However, unlike the tool object, the non-tool object does not make the "toy" object appear in the environment. The agent records both experiences as a concrete schema for the tool object and a generalised schema for both tool and non-tool objects. This schema contains the common observations in the concrete pre and post conditions. All the unmatched observations, including the observations that represent the toy, are added into the associated pre and postconditions wherever necessary, see Section 3.3.2 regarding the associated observations. These schemas demonstrate an association between the objects through a trigger action. To test this learning, we present both objects (i.e., tool and non-tool) in the environment. Once the agent perceives both objects, we provide a target state to make the "toy" object appear in the environment at its dedicated position.

7.2.4 Results: Experiment 2

Although the agent is provided with a limited number of actions i.e., reach and grasp, it is still able to perform the same action repeatedly, until less excited. The noise in the system and variations in the perceived objects lead to lots of repetitions of the same actions. In the results here, we are only representing interesting events, reach to grasp and reach towards the

trigger position.

The experiment starts with the tool object in the environment. Figure 7.15 shows excitations of schemas during the play behaviour with both tool and non-tool objects.

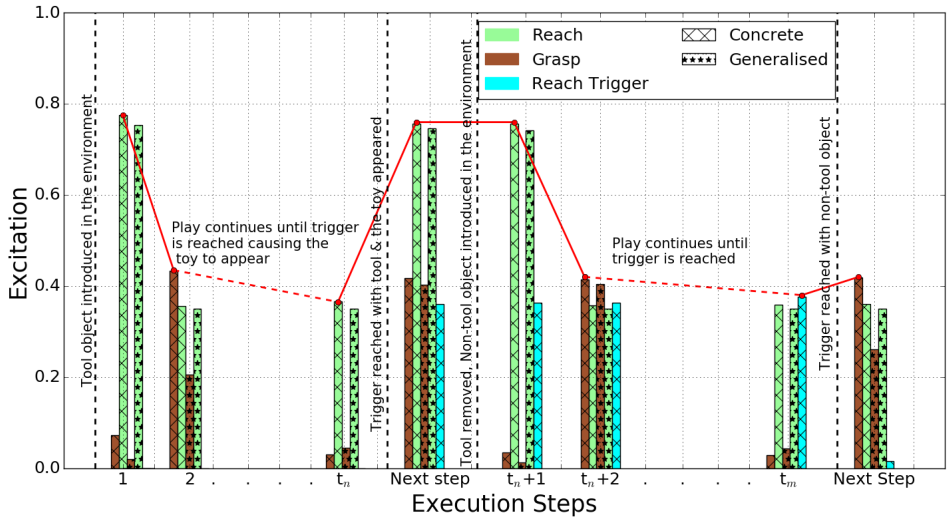


Fig. 7.15 Play behaviour to learn the association between the tool and the trigger position.

The agent starts with reaching for the tool object, as it is the only object present in the environment. Once the object is grasped, the agent plays with the object using different actions present in the memory. Later, at the execution t_n in Figure 7.15, the agent reaches for the trigger position, causing the toy object to appear in the environment. The action execution leads the agent to record the associations in a schema.

Following the experience with the tool object, we reset the environment and introduced the non-tool object. The agent reaches and grasps the non-tool object, at execution $t_n + 1$ and $t_n + 2$ respectively in Figure 7.15. Subsequently, the agent plays with the non-tool object and finally reaches for trigger position, using the previously generated schema shown at execution t_m. However, this action did not cause the toy object to appear in the environment. This leads the agent to develop a generalised schema through combining effects of the reach action

on the tool and non-tool objects towards the trigger position. The new generalised schema contains the observations in the concrete pre and post conditions that are common in both reach actions i.e., reach towards the trigger position while holding the tool or non-tool objects. Observations representing the "toy" object are added in the associated postconditions, as associated observations, of the newly created generalised schema.

To test the learning with both objects we introduce both, tool and non-tool, objects in the environment together. Once the agent perceives the environment, we ask it to bring the toy object in the environment by defining object features with its position where it was observed previously. Figure 7.16 shows the suggested schema chains and their excitations to achieve this objective.

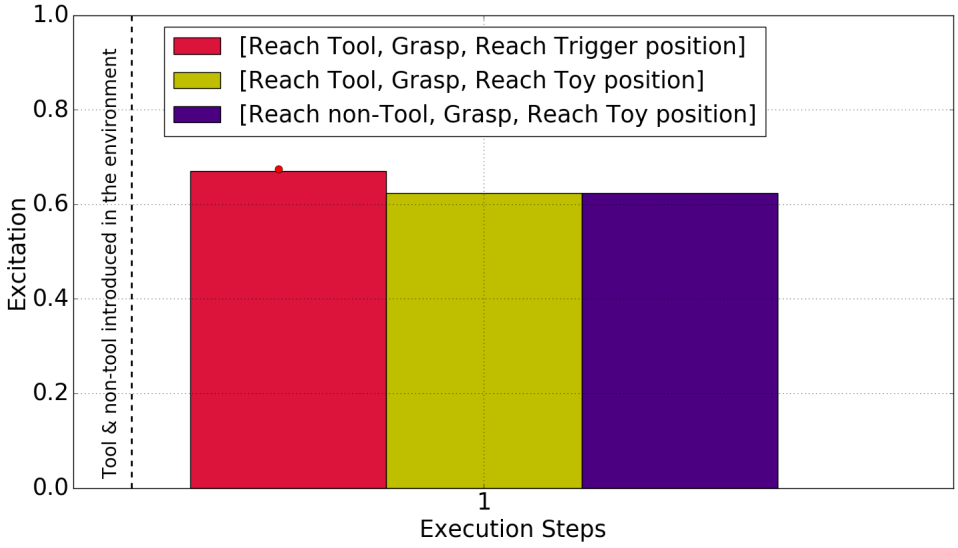


Fig. 7.16 Suggested chains when the requested to make the toy appear in the environment

The agent suggested three different chains to achieve the objective to get the toy object. The highest excited chain contains actions to reach and grasp the tool object, and reach towards the trigger position. This is the only chain which can achieve this objective state while the remaining two chains are "false positive" as those will not achieve the objective. While learning the association between the tool and the toy, the agent created concrete and

generalised schemas by combining previously generated concrete and generalised schemas. The agent used generalised schemas to create the "false positive" chains. Thus, the agent created over-generalised schemas and suggested chains based on these.

The highest excited chain consists of a combination of *concrete* and generalised schemas. It included the generalised reach and grasp schemas to obtain the tool object and the concrete schema to reach towards the trigger position. We selected the highest excited chain for execution which was successfully executed to make the toy appear.

Figure 7.17 shows the final step of the executed chain, where the object appeared in the environment.

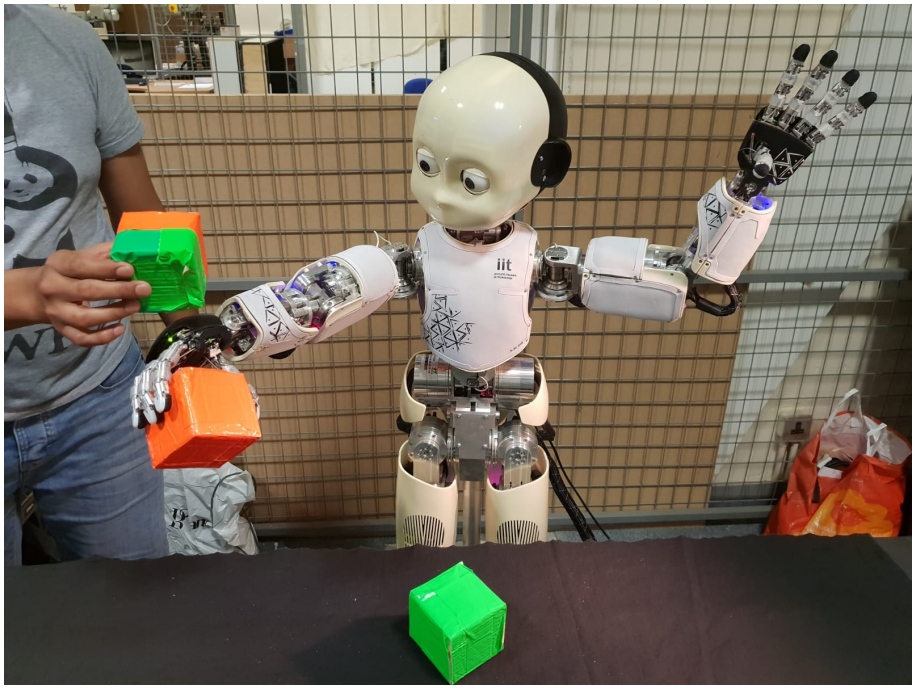


Fig. 7.17 Demonstration of the final step of the executed chain to bring the toy object in the environment.

As the agent reaches the trigger position, we (user) presented the toy object in the environment by holding it in the agent's visual space. To reduce noise, the object is presented in such

a way that only the toy object appears in the agent's visual space, as shown in Figure 7.17.

7.3 Discussions & Conclusions

In Chapter 6 we the demonstrated development of high-level actions through schema chaining to achieve a distant objective (in perception) selected through the excitation mechanism. In extension to this, in this chapter, we demonstrated the use of schema chaining in scaffolding the learning through solving a problem provided by an external agent. In the experiments, the agent (iCub) develops knowledge, through playful exploration, about the outcome of different actions on the given objects. Following that, the agent is given a target sensory state as an objective by the external agent (user). The agent finds potential solutions to achieve the state by matching it with the outcomes of action schemas i.e., postconditions, and linking it with the intermediary steps (schemas) to reach that point, i.e., objective schema preconditions, from the current world state.

In Experiment 1 the agent develops complex skills through exploratory play behaviours. The agent demonstrates usage of its learning to solve a problem related to the skills developed through play. In developmental psychology, scaffolding involves developing a physical or mental capability in infants through providing a restricted environment related to the ability [25]. Thus, this experiment demonstrates scaffolding of the skills and knowledge in a Dev-PSchema enabled agent. The agent demonstrated its skills in solving the problems provided as objectives by the external agent. As the problems are represented in sensory perceptions (state), thus the problem solving mechanism ends up with several solutions that are believed to be solving a problem, as shown in Figures 7.4, 7.7, 7.9 and 7.11. However, not every solution will actually achieve the objective state, as demonstrated in Figures 7.7 and 7.9. This provides a learning opportunity for the agent to shape its knowledge and therefore less likely to consider unsuccessful solutions in the future. Furthermore, this leads the user

as an external agent to shape the agent's knowledge through utilising its skills.

In the second experiment, the agent learns how an action on an object affects the perception for another object at different location i.e., making it appear. This capability can be compared with tool-use, where the agents use one object to act or bring an effect on another object. Although this experiment may not represent a demonstration of tool-use, it can be considered as a demonstration of the ability of the system to learn tool-use through action sequencing. Thus the capability demonstrates learning the associations between the two objects through an action on one of them.

In this experiment, the agent retrieves the object i.e., toy, by moving the tool object towards the trigger position. There are several examples in developmental psychology regarding infants using one object to retrieve another [118, 66]. Goubet et al. [66] found that the 14-18 month old subjects performed better in the toy retrieving tasks with more difficulty. The toy retrieving task consisted of 3 to 11 steps in total, varying in difficulty. However, 9 months old infants performed better than the 14-18 months old in the tasks with three steps, reach the mat, pull and retrieve the target object. The authors believed that older infants may be inclined to use more newly developed skills than younger infants, causing them to fail to complete the task. Furthermore, the results demonstrated that the older infants asked for the experimenter's help for a small number of times and used the tool more efficiently as compared to the younger infants. The infants, in this study, not only demonstrated the capability to build a plan to obtain the target object but the capability of tool-use.

The goal of the experiments was to demonstrate the capacity in Dev-PSchema for shaping the knowledge, gained during the exploratory play, through utilising the skills developed through experiences. The agent has successfully demonstrated to scaffold its knowledge and skills through play and utilise the skills to re-achieve a state in the environment, provided as a challenge to solve.

Chapter 8

Conclusion and Future Developments

In this thesis, we have presented a schema-based learning system with underlying mechanisms for excitation, generalisation and chaining. We have presented a schema-based play generator for artificial agents, termed as Dev-PSchema, inspired from Piaget's cognitive developmental theory. With the help of experiments in both, a simulated environment and with the real iCub robot, we have demonstrated the ability of the system to create schemas of sensorimotor experiences from playful interaction with objects in a given environment.

8.1 Conclusion on Contributions

In Chapter 1 we briefly introduced the contributions of this thesis. Here, we summarise and provide conclusions on those contributions.

8.1.1 Open-ended Learning and Adaptability

The abstract representation of the observations, i.e., sensory information, and the actions enable Dev-PSchema to be an open-ended learning system that is capable of processing new sensory information and actions as captured during play. Although, the experiments presented in this thesis were designed to demonstrate the underlying components of Dev-PSchema, the experiments at some level demonstrate open-ended learning. The experiment,

presented in Section 6.2.3, demonstrates the exploratory behaviours of the agents through the decision making process. Although, the experiments ended when the agent, achieves the goal, which the agent was unaware of, set by the experimenter, the experiment demonstrates the decision making process and development of higher level actions through combinations of schema, which in turn will enable the agent to explore further. Furthermore, due to abstraction in observation and action, the experiment is performed with two totally different platforms, a simple Sandbox simulator and a real humanoid robot, without any changes in the Dev-PSchema system. This provides evidence to show the adaptability of Dev-PSchema to different platforms, without any changes within the system itself.

Furthermore, in Chapter 7 the agents demonstrated learning in different ways, through exploration. The agents learnt different schemas through performing actions on the objects in the environment. The learnt schemas were used to suggest different solutions for a given problem. The suggested solutions were developed through a combination of learnt concrete and generalised schemas based on experiences. The suggested solutions were obtained through predictions made with the generalised schema. However, all the suggested solutions do not necessarily lead to achieving the target state, as demonstrated in Section 7.2.2. The unsuccessful in attempts to solve the problems provide the agent further opportunities to learn through either developing new schemas or reducing the likelihood of suggesting it in the future. The experiment demonstrates the open-ended learning in the system, developing skills to solve problems and adapting learning to different situations.

8.1.2 Generalising from Experiences

The generalisation enables the agent to build a general concept about effects through actions on the perceived object by finding contextual similarity in similar experiences. The generalisation mechanism helps to develop a general concept about action-object pair, through inductive inference for generalisation, and a generalised representation (schema) helps to predict action outcomes. The generalisation mechanism uses schemas with similar actions

to build a generalised schema through inductive inference. It generalises the variations in object features between similar schemas that share the same types of perception in response to the action. The mechanism focuses on the common features (object properties) present in schemas for generalisation to develop a common representation of all the schemas with the same type of action. Thus the mechanism helps the agent to extend its knowledge to novel situations and environments, based on perceptual similarity to previously experienced environments, without needing to build a separate memory for each individual instance. This capability of generalisation has been demonstrated through the two experiments presented in Chapter 4. The first experiment, presented in Section 4.2.1, demonstrates developing an understanding of non-visual properties of objects linked to visual properties. The second experiment, presented in Section 4.2.3, demonstrates how generalisation can be used to predict object properties with numerical values through functional generalisation. In conclusion, the experiments provide evidence to develop a general understanding about objects and the effects of actions on them, predicting object features through generalised schemas developed through playful exploration.

8.1.3 Simulating Individual Variations

The excitation mechanism of the system drives the play behaviour based on the perceived environment and the agent's previous experiences. The excitation mechanism, that is modelled on intrinsic motivations and habituation paradigm in developmental psychology, generates play behaviours that demonstrate different preferences in the environment. The agents demonstrated their interests toward either familiar or novel objects in the environment depending upon the tuning parameters of the excitation system i.e., ω_1 and ω_2 (see Chapter 5). This mechanism enables the system to simulate several agents, demonstrating different behaviours in the same environment, whilst having the same state of knowledge and experiences. As each individual in the environment develops with the different experiences, therefore variations between their preferences are more obvious. Furthermore, the mechanism enables the agents to demonstrate either exploratory behaviours or exploitation of existing actions

through tuning the other set of excitation parameters i.e., ω_3 and ω_4. This play behaviour helps the agent in the decision-making progress for selecting a suitable action for a perceived object and enables it to extend its knowledge through playful explorations. In conclusion, the excitation mechanism provides the capability to explore the environment, learning and applying the skills and understanding gained. The agent can demonstrate different preferences in decision making process, through tuning the excitation parameters as demonstrated through experiments in Chapter 5.

8.1.4 Forming Higher Level Actions

We extended the chaining mechanism of the system, originally developed for PSchema [172], enabling the agent to develop and use chains in the decision making process through the excitation mechanism. Previously, in PSchema, the chaining mechanism was only used for solving a problem provided by an external agent. In the developed excitation system, the agent finds the possible sequences of actions to achieve the postconditions of a schema which are not possible to achieve directly. Such action sequences can be considered as high-level actions, represented as schema chains in the system. The chaining mechanism demonstrates the learning of high-level actions, modelled on the planning behaviour observed in early infancy [146]. Thus the mechanism demonstrates the developmental progression in knowledge shaped from a set of single action schemas to schema chains representing high-level actions. The mechanism is further extended by considering the schema chains as either reflexive chains or motor programs, depending upon the repetitive use of the chains as demonstrated in the experiment presented in Section 6.2. This feature is also modelled on the infants' behaviours observed in developmental psychology [160, 98, 41]. A motor program represents a schema chain, that has been repeatedly used successfully executed as a singular unit, without considering feedback at each individual step in the sequence.

Dev-PSchema provides an opportunity for an external agent to scaffold the agent's knowledge using the chaining mechanism by providing a target state to achieve. The mechanism

provides two capabilities for the agent; i) scaffolding, and, ii) shaping knowledge. The agent uses the chaining mechanism to find different high-level actions that achieve the target state provided by the external agent and executes the solution to achieve the state. The agent scaffolds its knowledge and skills through experiences increasing the skills' complexity, leading the agent to develop different solutions that achieve the same solution. If the agent finds alternative solutions for a problem, it interacts with the user i.e., external agent, to select a solution from the list of alternative solutions. This provides an opportunity for the external agent to select a suitable solution that will lead the agent to develop further and shape its learning. During the problem solving the agent executes a user selected solution, however, if the chain is not successful, its excitation will be reduced, decreasing the likelihood of it being selected again next time.

The chaining mechanism, in conclusion, provides the capability for the agent to construct schema chains, leading it to learn high-level actions through combinations of basic actions. As these chains are successfully reused, they can later be considered and executed as if they were a single action.

By combining all the contributions, this model will lead a Dev-PSchema enabled agent to develop its knowledge incrementally through experiences, develop complex behaviours and utilise those to interact with novel objects and predict the outcomes. In a hypothetical scenario where an agent with weights leaned toward exploratory behaviours that will lead the agent to develop more complete generalised schemas through interactions with different objects using different actions. This will also lead the agent to develop complex actions (schema chains) using generalised schemas, which can further be utilised in novel situations. An agent with weights leaned toward the exploitative behaviours will develop more partially generalised schemas through interactions with preferred objects using repetitive behaviours. This will lead the agent to develop complex actions using concrete and partially generalised schemas, which can be utilised further in similar situations.

8.2 Research Question and Objectives: A Revisit

In Chapter 1, we set research questions given below:

i. Can a schema based model offer open-ended learning through exploratory play and be able to incorporate new information without any predefined template?

ii. Can a schema system develop generalisation structures, as observed in infants, through play behaviours and find a functional relationship in the generalisation?

iii. Can a schema system simulate infants with different preferences for actions in a given situation of the environment as infants do?

iv. Can a schema based system develop action sequences using basic schemas, through exploratory play, and utilise the sequences as high-level actions, as observed in humans?

v. Can an external user help to scaffold and shape the schema knowledge developed through play?

Through the series of experiments, presented in Chapters 4 to 7, we demonstrated the capabilities of Dev-PSchema extended through its sub-components. We demonstrated that Dev-PSchema is capable of developing general concepts about objects and predict the effects of different behaviours on them through generalisation, as seen in Chapter 4.

In Chapters 6 and 7 we demonstrated Dev-PSchema building the high-level actions in the shape of schema sequences. The chains represent high-level actions, considered as the skills, developed through exploration and later used to achieve the objectives that are not possible to achieve by any single schema. The acquired skills were applied in novel situations for achieving an objective state, provided by either the internal decision making process in play mode or the external agent in problem solving mode. This is also seen as an opportunity to scaffold and shape the agent's knowledge.

Furthermore, Dev-PSchema demonstrated the capability to simulate variations in behaviours by adjusting preferences. The decision making process of Dev-PSchema enables the agents to follow different learning paths resulting in the simulation of different individuals. Thus Dev-PSchema provides opportunities for the agents to demonstrate playful behaviours in the environment in order to learn different skills and build object-action concepts. In all these experiments provide an evidence that Dev-PSchema offers open-ended learning. This is further supported by smooth switching of the agent, from a simple simulator to a complex humanoid robot, without any major changes in Dev-PSchema.

8.3 Future Work

In this section, we propose some future developments for Dev-PSchema to enhance its capabilities.

Dev-PSchema builds schemas with pre and post conditions along with associated conditions, see Chapter 3. These associated conditions once added into a schema remain unaltered in the memory, except when these are obtained repeatedly through reuse of the same action and added to the concrete pre/post-conditions. Such associations may be part of the noise included in perceptions acquired following an action. Dev-PSchema can be further developed to add tolerance for the associated observations following the repeated experiences. The capability can be expanded further to remove such associations as a part of the behavioural effect in a given situation. The mechanism can be developed to remove the associations through repeated experiences that fail to obtain the associations.

Dev-PSchema has been observed to develop over-generalised schemas, as demonstrated in Chapters 4 and 7. The possible solution is to amend the generalised schemas through deductive reasoning, a method of reasoning from very generalised to specific examples based on the evidences. The generalisation algorithm can be extended to de-generalise schemas

through experiences. This solution will not increase the number of schemas in the memory, however this will change the contents i.e., object properties, of generalised schemas from generalised to specific values. Furthermore, the generalisation mechanism can be further extended to learn bounds and limitations for property values present in generalised schemas. For example, if a generalised reach schema contains generalised coordinates, then theoretically the agent assumes it can reach any point in the visual field. However, it is not possible without locomotion which is not yet modelled. To solve this, the agent should be able to learn possible extreme values for each generalised variable through its experiences.

The functional generalisation in Dev-PSchema is limited to the additive function only, hence finds a linear relationship between the properties present in the schema pre and post conditions.The functional generalisation can be further extended to include high-level functional labelling such as less/more, smaller/larger, closer/farther etc. This capability can be extended into Dev-PSchema and labelling through interaction when numerical functional generalisation is built through concrete schemas having different numerical values for generalised property.

The chaining mechanism is limited to develop chains from the schemas. This can be expended to develop chains of the chains. Thus building more complex actions by combining the complex actions. However, longer chains will be less likely to be successful in a real environment due to unpredictable noise.

The excitation calculator in Dev-PSchema weights all the perceived properties equally. However, the mechanism should be able to learn prioritisation of the properties for play behaviours. For example, in the push action the shape of the object may be prioritised higher than its colour as the shape of the object affects the distance travelled when pushed, not the colour. Furthermore, the mechanism enables the agents to explore and play with novel objects in the environment. Thus while playing with an object, the agent may drop the object and interact with other perceived (novel) objects. This preference can be changed manually

by varying the excitation parameters i.e., ω_1 and ω_2. Dev-PSchema may be extended to develop the object preferences through the agent's behaviours and acquired outcomes that lead to dynamic changes in the preferences. For example, a hungry infant will be interested in food rather than play. The infant's behaviour may switch to play once the need is satisfied.

Montesano et al. [125] proposed that a robotic model inspired from developmental psychology should be able to demonstrate learning through active experiences and imitations. As this thesis is mainly focused on developing object knowledge using experiences gained through active exploratory play, learning through observation and imitation is not incorporated in Dev-PSchema. Dev-PSchema may be extended to develop schemas through observations. This could be implemented through simulating observed behaviours as active behaviours, as if the agent is performing the action itself. This idea can be supported from developmental psychology and neuroscience where mirror neurons in the human brain have been observed to be activated when a subject performs an action him/her-self or observes the same action being performed by the other subject [156, 50, 49]. This will extend the capabilities of Dev-PSchema system, leading it to learn through passive experiences. Furthermore, this capability will provide the opportunity to expand the agent's learning through social interactions with the other agents.

References

[1] Adams, M. J. and Collins, A. (1977). A schema-theoretic view of reading. *BBN report; no. 3548*.

[2] Adler, S. A. and Haith, M. M. (2003). The nature of infants' visual expectations for event content. *Infancy*, 4(3):389–421.

[3] Adolph, K. E., Bertenthal, B. I., Boker, S. M., Goldfield, E. C., and Gibson, E. J. (1997). Learning in the development of infant locomotion. *Monographs of the society for research in child development*, pages i–162.

[4] Agrawal, P., Nair, A. V., Abbeel, P., Malik, J., and Levine, S. (2016). Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082.

[5] Aguilar, W. and Pérez y Pérez, R. (2017). Emergence of eye–hand coordination as a creative process in an artificial developmental agent. *Adaptive Behavior*, 25(6):289–314.

[6] Aguilar, W. and y Pérez, R. P. (2015). Dev er: A computational model of early cognitive development as a creative process. *Cognitive Systems Research*, 33:17–41.

[7] Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., et al. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557.

[8] Asada, M. (2011). *Neuromorphic and Brain-Based Robots*, chapter Can cognitive developmental robotics cause a paradigm shift?, pages 251–273. Cambridge University Press.

[9] Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., and Yoshida, C. (2009). Cognitive developmental robotics: A survey. *IEEE Transactions on Autonomous Mental Development*, 1(1):12–34.

[10] Asada, M., MacDorman, K. F., Ishiguro, H., and Kuniyoshi, Y. (2001). Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37(2):185–193.

[11] Asimov, I. (1933). *I, Robot*. Facwcett Crest.

[12] Baldassarre, G. (2011). What are intrinsic motivations? a biological perspective. In *Development and learning (icdl), 2011 ieee international conference on*, volume 2, pages 1–8. IEEE.

[13] Baldwin, D. A., Markman, E. M., and Melartin, R. L. (1993). Infants' ability to draw inferences about nonobvious object properties: Evidence from exploratory play. *Child development*, 64(3):711–728.

[14] Barto, A. G., Singh, S., and Chentanez, N. (2004). Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 112–19. Citeseer.

[15] Basheer, I. and Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43:3–31.

[16] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, 24(4):509–522.

[17] Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J.-M., Alvarez-Icaza, R., Arthur, J. V., Merolla, P. A., and Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716.

[18] Berlyne, D. E. (1950). Novelty and curiosity as determinants of exploratory behaviour. *British Journal of Psychology*, 41(1-2):68–80.

[19] Berlyne, D. E. (1954). A theory of human curiosity. *British Journal of Psychology. General Section*, 45(3):180–191.

[20] Berlyne, D. E. (1965). Structure and direction in thinking.

[21] Bogdan, P. A., Rowley, A. G. D., Rhodes, O., and Furber, S. B. (2018). Structural plasticity on the spinnaker many-core neuromorphic system. *Frontiers in Neuroscience*, 12:434–453.

[22] Bomba, P. C. and Siqueland, E. R. (1983). The nature and structure of infant form categories. *Journal of Experimental Child Psychology*, 35(2):294–328.

[23] Bornstein, M. H. and Lamb, M. E. (2002). *Development in infancy: An introduction*. Psychology Press.

[24] Boyke, J., Driemeyer, J., Gaser, C., Büchel, C., and May, A. (2008). Training-induced brain structure changes in the elderly. *Journal of Neuroscience*, 28(28):7031–7035.

[25] Breazeal, C. (1998). Learning by scaffolding. *Thesis Proposal*, pages 1–106.

[26] Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23.

[27] Brooks, R. A., Breazeal, C., Marjanovic, M., Scassellati, B., and Williamson, M. M. (1999). The cog project: Building a humanoid robot. *Lecture Notes in Computer Science*, pages 52–87.

[28] Brooks, R. A., Stein, L. A., et al. (1994). Building brains for bodies. *Autonomous Robots*, 1(1):7–25.

[29] Brugger, A., Lariviere, L. A., Mumme, D. L., and Bushnell, E. W. (2007). Doing the right thing: Infants' selection of actions to imitate from observed event sequences. *Child development*, 78(3):806–824.

[30] Canfield, R. L. and Haith, M. M. (1991). Young infants' visual expectations for symmetric and asymmetric stimulus sequences. *Developmental Psychology*, 27(2):198–208.

[31] Cangelosi, A., Schlesinger, M., and Smith, L. B. (2015). *Developmental robotics: From babies to robots*. MIT Press.

[32] Čapek, K. (1920). Rur-rossum's universal robots.

[33] Cavanaugh, J. C. and Blanchard-Fields, F. (2018). *Adult development and aging*. Cengage Learning.

[34] Chao, F., Lee, M. H., Jiang, M., and Zhou, C. (2014). An infant development-inspired approach to robot hand-eye coordination. *International Journal of Advanced Robotic Systems*, 11(2):1–14.

[35] Chaput, H. H. (2004). *The constructivist learning architecture: A model of cognitive development for robust autonomous robots*. PhD thesis.

[36] Chernova, S. and Veloso, M. (2004). An evolutionary approach to gait learning for four-legged robots. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2562–2567. IEEE.

[37] Clark, J. E. (2005). From the beginning: a developmental perspective on movement and mobility. *Quest*, 57(1):37–45.

[38] Colombo, J. (2001). The development of visual attention in infancy. *Annual review of psychology*, 52(1):337–367.

[39] Colombo, J., Shaddy, D. J., Richman, W. A., Maikranz, J. M., and Blaga, O. M. (2004). The developmental course of habituation in infancy and preschool outcome. *Infancy*, 5(1):1–38.

[40] De Garis, H., Shuo, C., Goertzel, B., and Ruiting, L. (2010). A world survey of artificial brain projects, part i: Large-scale brain simulations. *Neurocomputing*, 74(1):3–29.

[41] Defeyter, M. A. and German, T. P. (2003). Acquiring an understanding of design: evidence from children's insight problem solving. *Cognition*, 89(2):133–155.

[42] Dharmendra S., M. Introducing a brain-inspired computer. http://www.research.ibm.com/articles/brain-chip.shtml, accessed on January 10, 2018.

[43] Dixon, P., McAnsh, S., and Read, L. (2012). Repetition effects in grasping. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 66(1):1–17.

[44] Dobelle, W. H. (2000). Artificial vision for the blind by connecting a television camera to the visual cortex. *ASAIO journal*, 46(1):3–9.

[45] Drescher, G. L. (1987). A mechanism for early piagetian learning. In *AAAI*, pages 290–294.

[46] Drescher, G. L. (1991). *Made-up minds: a constructivist approach to artificial intelligence*. MIT press.

[47] Earland, K., Lee, M., Shaw, P., and Law, J. (2014). Overlapping structures in sensory-motor mappings. *PloS one*, 9(1):1–16, e84240.

[48] Edsinger, A. and Kemp, C. C. (2006). What can i control? a framework for robot self-discovery. In *6th International Conference on Epigenetic Robotics*, pages 1–8. Citeseer.

[49] Fabbri-Destro, M. and Rizzolatti, G. (2008). Mirror neurons and mirror systems in monkeys and humans. *Physiology*, 23(3):171–179.

[50] Filimon, F., Nelson, J. D., Hagler, D. J., and Sereno, M. I. (2007). Human cortical representations for reaching: mirror neurons for execution, observation, and imagery. *Neuroimage*, 37(4):1315–1328.

[51] Finn, C. and Levine, S. (2017). Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE.

[52] Fischer, K. W. (1980). A theory of cognitive development: The control and construction of hierarchies of skills. *Psychological review*, 87(6):477–531.

[53] Flunkert, B. (2009). The role of the contingent negative variation in chunking. evidence from a go/nogo discrete sequence production task. B.S. thesis, University of Twente.

[54] Fromberg, D. P. and Bergen, D. (2012). *Play from birth to twelve: Contexts, perspectives, and meanings*. Routledge.

[55] Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of neural engineering*, 13(5):051001.

[56] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467.

[57] Furby, L. and Wilke, M. (1982). Some characteristics of infants' preferred toys. *The Journal of genetic psychology*, 140(2):207–219.

[58] Gelman, R. (1972). Logical capacity of very young children: Number invariance rules. *Child Development*, pages 75–90.

[59] Gelman, S. A. and Davidson, N. S. (2016). Young children's preference for unique owned objects. *Cognition*, 155:146–154.

[60] Gerber, R. J., Wilks, T., and Erdie-Lalena, C. (2010). Developmental milestones: motor development. *Pediatrics in Review*, 31(7):267–277.

[61] Gershkoff-Stowe, L., Connell, B., and Smith, L. (2006). Priming overgeneralizations in two-and four-year-old children. *Journal of Child Language*, 33(3):461–486.

[62] Giagkos, A., Lewkowicz, D., Shaw, P., Kumar, S., Lee, M., and Shen, Q. (2017). Perception of localized features during robotic sensorimotor development. *IEEE Transactions on Cognitive and Developmental Systems*, 9(2):127–140.

[63] Goertzel, B., Lian, R., Arel, I., De Garis, H., and Chen, S. (2010). A world survey of artificial brain projects, part ii: Biologically inspired cognitive architectures. *Neurocomputing*, 74(1):30–49.

[64] Gómez, R. L. and Gerken, L. (2000). Infant artificial language learning and language acquisition. *Trends in cognitive sciences*, 4(5):178–186.

[65] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[66] Goubet, N., Rochat, P., Maire-Leblond, C., and Poss, S. (2006). Learning from others in 9-18-month-old infants. *Infant and Child Development: An International Journal of Research and Practice*, 15(2):161–177.

[67] Graham, S. A. and Poulin-Dubois, D. (1999). Infants' reliance on shape to generalize novel labels to animate and inanimate objects. *Journal of child language*, 26(2):295–320.

[68] Guerin, F. (2010). Advancing knowledge of cognitive development sequences in infancy. *AI inspired Biology*, page 29.

[69] Guerin, F. (2011). Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review*, 26(2):209–236.

[70] Guerin, F. and McKenzie, D. (2008). A piagetian model of early sensorimotor development. In *Eighth International Conference on Epigenetic Robotics, University of Sussex*.

[71] Guizzo, E. (2010). Geminoid f gets job as robot actress. *IEEE Spectrum Online*.

[72] Haith, M. M., Hazan, C., and Goodman, G. S. (1988). Expectation and anticipation of dynamic visual events by 3.5-month-old babies. *Child development*, pages 467–479.

[73] Hirsh-Pasek, K. and Golinkoff, R. M. (2008). Why play= learning. *Encyclopedia on early childhood development*, pages 1–7.

[74] Honavar, V. (2006). Artificial intelligence: An overview. 2007-12-1]. http://www. cs. i-astate. edu/-cs572/handoutl. pdf.

[75] Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., and Tanie, K. (2001). Planning walking patterns for a biped robot. *IEEE Transactions on robotics and automation*, 17(3):280–289.

[76] Hughes, F. P. (2009). *Children, play, and development*. Sage.

[77] Hunter, M. A., Ames, E. W., and Koopman, R. (1983). Effects of stimulus complexity and familiarization time on infant preferences for novel and familiar stimuli. *Developmental Psychology*, 19(3):338–352.

[78] Jain, A. K., Mao, J., and Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, 29(3):31–44.

[79] Jax, S. A. and Rosenbaum, D. A. (2007). Hand path priming in manual obstacle avoidance: evidence that the dorsal stream does not only control visually guided actions in real time. *Journal of Experimental Psychology: Human Perception and Performance*, 33(2):425–441.

[80] Jensen, E. (1998). *Teaching with the brain in mind*. Association for Supervision and Curriculum Development.

[81] JONSSON, C.-O., REIMBLADH-TAUBE, G., and Sjöswärd, E. (1993). Forms, uses and functions of children's favourite objects. *Scandinavian journal of psychology*, 34(1):86–93.

[82] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[83] Kansky, K., Silver, T., Mély, D. A., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *International Conference on Machine Learning*, pages 1809–1818.

[84] Kaplan, F. and Oudeyer, P.-Y. (2006). Curiosity-driven development. In *Proceedings of the International Workshop on Synergistic Intelligence Dynamics*, pages 1–8. Citeseer.

[85] Keele, S. W. (1968). Movement control in skilled motor performance. *Psychological bulletin*, 70(6p1):387–403.

[86] Kellman, P. J. and Spelke, E. S. (1983). Perception of partly occluded objects in infancy. *Cognitive psychology*, 15(4):483–524.

[87] Kent, S. W., Wilson, A. D., Plumb, M. S., Williams, J. H., and Mon-Williams, M. (2009). Immediate movement history influences reach-to-grasp action selection in children and adults. *Journal of motor behavior*, 41(1):10–15.

[88] Khodagholy, D., N Gelinas, J., Thesen, T., Doyle, W., Devinsky, O., G Malliaras, G., and Buzsáki, G. (2014). Neurogrid: Recording action potentials from the surface of the brain. *Nature neuroscience*, 18.

[89] Kirkham, N. Z., Slemmer, J. A., and Johnson, S. P. (2002). Visual statistical learning in infancy: Evidence for a domain general learning mechanism. *Cognition*, 83(2):B35–B42.

[90] Kober, J. and Peters, J. (2012). Reinforcement learning in robotics: A survey. In *Reinforcement Learning*, pages 579–610. Springer.

[91] Koulaguina, E. and Shi, R. (2010). The mechanism underlying the learning of rules and exceptions in 14-month-old infants. In *A supplement to the Proceedings of the 34th Boston University Conference on Language Development [BUCLD34]*, pages 1–13. Cascadilla Press Somerville, MA.

[92] Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., Ude, A., Asfour, T., Kraft, D., Omrčen, D., et al. (2011). Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 59(10):740–757.

[93] Kumar, S., Shaw, P., Giagkos, A., Braud, R., Lee, M. H., and Shen, Q. (2018). Developing hierarchical schemas and building schema chains through practice play behaviour. *Frontiers in neurorobotics*, 12:1–33.

[94] Kumar, S., Shaw, P., Lewkowicz, D., Giagkos, A., Lee, M., and Shen, Q. (2016a). Generalising predictable object movements through experience using schemas. In *International Conference on Simulation of Adaptive Behavior*, pages 329–339. Springer.

[95] Kumar, S., Shaw, P., Lewkowicz, D., Giagkos, A., Shen, Q., and Lee, M. (2016b). Developing object understanding through schema generalisation. In *Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2016 Joint IEEE International Conference on*, pages 33–38. IEEE.

[96] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64.

[97] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

[98] Lashley, K. S. (1951). *The problem of serial order in behavior*, volume 21. Bobbs-Merrill.

[99] Laud, A. and DeJong, G. (2002). Reinforcement learning and shaping: Encouraging intended behaviors. In *ICML*, pages 355–362.

[100] Law, J., Lee, M., Hülse, M., and Tomassetti, A. (2011). The infant development timeline and its application to robot shaping. *Adaptive Behavior*, 19(5):335–358.

[101] Law, J., Shaw, P., Earland, K., Sheldon, M., and Lee, M. H. (2014a). A psychology based approach for longitudinal development in cognitive robotics. *Frontiers in neurorobotics*, 8:1–19.

[102] Law, J., Shaw, P., and Lee, M. (2013). A biologically constrained architecture for developmental learning of eye–head gaze control on a humanoid robot. *Autonomous Robots*, 35(1):77–92.

[103] Law, J., Shaw, P., Lee, M., and Sheldon, M. (2014b). From saccades to grasping: A model of coordinated reaching through simulated development on a humanoid robot. *IEEE Transactions on Autonomous Mental Development*, 6(2):93–109.

[104] LeCun, Y. (2018). The power and limits of deep learning: In his iri medal address, yann lecun maps the development of machine learning techniques and suggests what the future may hold. *Research-Technology Management*, 61(6):22–27.

[105] Lee, M. H. (2011). Intrinsic activitity: from motor babbling to play. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, pages 1–6. IEEE.

[106] Legare, C. H. (2011). Exploring explanation: Explaining inconsistent evidence informs exploratory, hypothesis-testing behavior in young children. *Child Development*, 83(1):173–185.

[107] Lewkowicz, D., Giagkos, A., Shaw, P., Kumar, S., Lee, M., and Shen, Q. (2016). Towards learning strategies and exploration patterns for feature perception. In *Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2016 Joint IEEE International Conference on*, pages 278–283. IEEE.

[108] Loewenstein, G. (1994). The psychology of curiosity: A review and reinterpretation. *Psychological Bulletin*, 116(75-98).

[109] Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connection Science*, 15(4):151–190.

[110] Mandler, J. M. (1988). How to build a baby: On the development of an accessible representational system. *Cognitive Development*, 3(2):113–136.

[111] Mandler, J. M. and McDonough, L. (1998). Studies in inductive inference in infancy. *Cognitive psychology*, 37(1):60–96.

[112] Mandler, J. M. and McDonough, L. (2000). Advancing downward to the basic level. *Journal of Cognition and Development*, 1(4):379–403.

[113] Mar, T., Tikhanoff, V., Metta, G., and Natale, L. (2015). Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3200–3206. IEEE.

[114] Markram, H. (2006). The blue brain project. *Nature Reviews Neuroscience*, 7(2):153–160.

[115] Markram, H. (2012). The human brain project. *Scientific American*, 306(6):50–55.

[116] Mather, E. (2013). Novelty, attention, and challenges for developmental psychology. *Frontiers in psychology*, 4:491–494.

[117] McCall, R. B. (1974). Exploratory manipulation and play in the human infant. *Monographs of the Society for Research in Child Development*, pages 1–88.

[118] McCarty, M. E., Clifton, R. K., and Collard, R. R. (1999). Problem solving in infancy: the emergence of an action plan. *Developmental psychology*, 35(4):1091–1101.

[119] McLeod, S. (2009). Jean piaget. Simply psychology, https://www.simplypsychology.org/piaget.html, accessed on December, 2017.

[120] Meeden, L. A. and Blank, D. S. (2006). Introduction to developmental robotics.

[121] Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673.

[122] Metta, G., Sandini, G., Vernon, D., Natale, L., and Nori, F. (2008). The icub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*, pages 50–56. ACM.

[123] Meyer, J.-A., Husbands, P., and Harvey, I. (1998). Evolutionary robotics: A survey of applications and problems. In *European Workshop on Evolutionary Robotics*, pages 1–21. Springer.

[124] Mikaitis, M., Lester, D. R., Shang, D., Furber, S., Liu, G., Garside, J., Scholze, S., Höppner, S., and Dixius, A. (2018). Approximate fixed-point elementary function accelerator for the spinnaker-2 neuromorphic chip. In *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*, pages 37–44. IEEE.

[125] Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: from sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26.

[126] Morales, A., Asfour, T., Azad, P., Knoop, S., and Dillmann, R. (2006). Integrated grasp planning and visual object localization for a humanoid robot with five-fingered hands. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5663–5668. IEEE.

[127] Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370.

[128] Newcombe, N. and Huttenlocher, J. (1992). Children's early ability to solve perspective-taking problems. *Developmental psychology*, 28(4):635–643.

[129] Newman, C. D. (2001). *The planning and control of action in normal infants and children with Williams syndrome*. PhD thesis, University of London.

[130] Newport, E. L. (1990). Maturational constraints on language learning. *Cognitive science*, 14(1):11–28.

[131] Nicholson, K. G. and Humphrey, G. K. (2004). The effect of colour congruency on shape discriminations of novel objects. *Perception*, 33(3):339–353.

[132] Nicolopoulou, A. (2010). The alarming disappearance of play from early childhood education. *Human development*, 53(1):1–4.

[133] Nilsson, N. J. (1998). *Artificial intelligence: a new synthesis*. Elsevier.

[134] Oudeyer, P.-Y. (2004). Intelligent adaptive curiosity: a source of self-development. In *Proceedings of the Fourth International Workshop on Epigenetic Robotics*, pages 127–130.

[135] Oudeyer, P.-Y. and Kaplan, F. (2008). How can we define intrinsic motivation? In *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies, Lund: LUCS, Brighton*, pages 1–9. Lund University Cognitive Studies, Lund: LUCS, Brighton.

[136] Oudeyer, P.-Y., Kaplan, F., Hafner, V. V., and Whyte, A. (2005). The playground experiment: Task-independent development of a curious robot. In *Proceedings of the AAAI Spring Symposium on Developmental Robotics*, pages 42–47. Stanford, California.

[137] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17.

[138] Paulus, M., Hunnius, S., van Wijngaarden, C., Vrins, S., van Rooij, I., and Bekkering, H. (2011). The role of frequency information and teleological reasoning in infants' and adults' action prediction. *Developmental psychology*, 47(4):976–983.

[139] Pellegrini, A. D. and Smith, P. K. (1998). The development of play during childhood: forms and possible functions. *Child Psychology and Psychiatry Review*, 3(2):51–57.

[140] Perotto, F. S. (2013). A computational constructivist model as an anticipatory learning mechanism for coupled agent-environment systems. *Constructivist Foundations*, 9(1):46–56.

[141] Petit, M., Pointeau, G., and Dominey, P. F. (2016). Reasoning based on consolidated real world experience acquired by a humanoid robot. *Interaction Studies*, 17(2):248–278.

[142] Pezzulo, G. (2008). Coordinating with the future: the anticipatory nature of representation. *Minds and Machines*, 18(2):179–225.

[143] Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *science*, 318(5853):1088–1093.

[144] Piaget, J. (1952). *Play, dreams and imitation in childhood.* WW Norton & Co.

[145] Piaget, J. (1954). The construction of reality in the child.

[146] Piaget, J. (1965). The stages of the intellectual development of the child. *Educational psychology in context: Readings for future teachers*, pages 98–106.

[147] Piaget, J. and Cook, M. (1952). *The origins of intelligence in children*, volume 8. International Universities Press New York.

[148] Piaget, J. and Inhelder, B. (1969). *The Psychology of the Child, translated from the French by Helen Weaver*. New York: Basic Books.

[149] Piek, J. P. (2006). *Infant motor development*, volume 10. Human Kinetics.

[150] Pierce, D., Munier, V., and Myers, C. T. (2009). Informing early intervention through an occupational science description of infant–toddler interactions with home space. *American Journal of Occupational Therapy*, 63(3):273–287.

[151] Pinto, L., Gandhi, D., Han, Y., Park, Y.-L., and Gupta, A. (2016). The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pages 3–18. Springer.

[152] Pounds, P., Mahony, R., and Corke, P. (2010). Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699.

[153] Pramling Samuelsson, I. and Johansson, E. (2006). Play and learning—inseparable dimensions in preschool practice. *Early Child Development and Care*, 176(1):47–65.

[154] Rakison, D. H. (2005). Developing knowledge of objects' motion properties in infancy. *Cognition*, 96(3):183–214.

[155] Rankin, C. H., Abrams, T., Barry, R. J., Bhatnagar, S., Clayton, D. F., Colombo, J., Coppola, G., Geyer, M. A., Glanzman, D. L., Marsland, S., et al. (2009). Habituation revisited: an updated and revised description of the behavioral characteristics of habituation. *Neurobiology of learning and memory*, 92(2):135–138.

[156] Rizzolatti, G. and Craighero, L. (2004). The mirror-neuron system. *Annu. Rev. Neurosci.*, 27:169–192.

[157] Rochat, P. (1992). Self-sitting and reaching in 5- to 8-month-old infants: The impact of posture and its development on early eye-hand coordination. *Journal of Motor Behavior*, 24(2):210–220.

[158] Rosander, K. and von Hofsten, C. (2004). Infants' emerging ability to represent occluded object motion. *Cognition*, 91(1):1–22.

[159] Rose, S. A., Gottfried, A. W., Melloy-Carminar, P., and Bridger, W. H. (1982). Familiarity and novelty preferences in infant recognition memory: Implications for information processing. *Developmental Psychology*, 18(5):704–713.

[160] Rosenbaum, D. A., Cohen, R. G., Jax, S. A., Weiss, D. J., and Van Der Wel, R. (2007). The problem of serial order in behavior: Lashley's legacy. *Human movement science*, 26(4):525–554.

[161] Ruff, H. A. (1986). Components of attention during infants' manipulative exploration. *Child development*, pages 105–114.

[162] Ryan, R. M. and Deci, E. L. (2000). When rewards compete with nature: The undermining. *Intrinsic and extrinsic motivation: The search for optimal motivation and performance*, pages 13–17.

[163] Sann, C. and Streri, A. (2007). Perception of object shape and texture in human newborns: evidence from cross-modal transfer tasks. *Developmental Science*, 10(3):399–410.

[164] Scassellati, B. (2002). Theory of mind for a humanoid robot. *Autonomous Robots*, 12(1):13–24.

[165] Schillaci, G. and Hafner, V. V. (2011). Random movement strategies in self-exploration for a humanoid robot. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 245–246. ACM.

[166] Schlesinger, M. (2003). A lesson from robotics: Modeling infants as autonomous agents. *Adaptive Behavior*, 11(2):97–107.

[167] Schlesinger, M. and Casey, P. (2003). Visual expectations in infants: Evaluating the gaze-direction model. pages 115–122.

[168] Schmuckler, M. A., Collimore, L. M., and Dannemiller, J. L. (2007). Infants' reactions to object collision on hit and miss trajectories. *Infancy*, 12(1):105–118.

[169] Sharma, V., Rai, S., and Dev, A. (2012). A comprehensive study of artificial neural networks. *International Journal of Advanced research in computer science and software engineering*, 2(10).

[170] Shaw, P., Law, J., and Lee, M. (2012). An evaluation of environmental constraints for biologically constrained development of gaze control on an icub robot. *Paladyn*, 3(3):147–155.

[171] Shaw, P., Lewkowicz, D., Giagkos, A., Law, J., Kumar, S., Lee, M., Shen, Q., and d'Autume, C. D. M. (2015). Babybot challenge: Motor skills. In *Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2015 Joint IEEE International Conference on*, pages 47–54. IEEE.

[172] Sheldon, M. (2013). *Intrinsically motivated developmental learning of communication in robotic agents*. PhD thesis, Aberystwyth University.

[173] Sheldon, M. and Lee, M. (2011). Pschema: A developmental schema learning framework for embodied agents. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, pages 1–7. IEEE.

[174] Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323.

[175] Sigman, M. (1976). Early development of preterm and full-term infants: Exploratory behavior in eight-month-olds. *Child Development*, pages 606–612.

[176] Slater, A. (1989). Visual memory and perception in early infancy. *Infant development*, pages 43–71.

[177] Slater, A. and Lewis, M. (2007). *Introduction to infant development*. Oxford University Press.

[178] Spelke, E. (1994). Initial knowledge: Six suggestions. *Cognition*, 50(1):431–445.

[179] Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental science*, 10(1):89–96.

[180] Spelke, E. S., Phillips, A., and Woodward, A. L. (1995). Infants' knowledge of object motion and human action. pages 44–78.

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

[182] Steele, D. and Pederson, D. R. (1977). Stimulus variables which affect the concordance of visual and manipulative exploration in six-month-old infants. *Child Development*, pages 104–111.

[183] Stojanov, G., Kulakov, A., and Clauzel, D. (2006). On curiosity in intelligent robotic systems. In *Proceedings. of the AAAI Fall Symposium on Interaction and Emergent Phenomena in Societies of Agents*, pages 44–51.

[184] Sugihara, K. and Smith, J. (1997). Genetic algorithms for adaptive motion planning of an autonomous mobile robot. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 138–143. IEEE.

[185] Summers, J. J. and Anson, J. G. (2009). Current status of the motor program: Revisited. *Human Movement Science*, 28(5):566–577.

[186] Thelen, E., Corbetta, D., Kamm, K., Spencer, J. P., Schneider, K., and Zernicke, R. F. (1993). The transition to reaching: Mapping intention and intrinsic dynamics. *Child development*, 64(4):1058–1098.

[187] Tremoulet, P. D., Leslie, A. M., and Hall, D. G. (2000). Infant individuation and identification of objects. *Cognitive Development*, 15(4):499–522.

[188] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

[189] Vadakkepat, P., Tan, K. C., and Ming-Liang, W. (2000). Evolutionary artificial potential fields and their application in real time robot path planning. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 256–263. IEEE.

[190] van der Wel, R. P., Fleckenstein, R. M., Jax, S. A., and Rosenbaum, D. A. (2007). Hand path priming in manual obstacle avoidance: evidence for abstract spatiotemporal forms in human motor control. *Journal of Experimental Psychology: Human Perception and Performance*, 33(5):1117–1126.

[191] Vernon, D. (2014). *Artificial cognitive systems: A primer*. MIT Press.

[192] Vernon, D., Von Hofsten, C., and Fadiga, L. (2011). *A roadmap for cognitive development in humanoid robots*, volume 11. Springer Science & Business Media.

[193] Verwey, W. B. (2001). Concatenating familiar movement sequences: The versatile cognitive processor. *Acta psychologica*, 106(1-2):69–95.

[194] von Hofsten, C. (1984). Developmental changes in the organization of prereaching movements. *Developmental Psychology*, 20(3):378–388.

[195] Von Hofsten, C. (2007). Action in development. *Developmental science*, 10(1):54–60.

[196] von Hofsten, C., Vishton, P., Spelke, E. S., Feng, Q., and Rosander, K. (1998). Predictive action in infancy: tracking and reaching for moving objects. *Cognition*, 67(3):255–285.

[197] Vygotsky, L. (1934). Thought and language. *Trans. E. Hanfmann and G. Vakar. Cambridge: MIT Press*.

[198] Vygotsky, L. S. (1967). Play and its role in the mental development of the child. *Soviet psychology*, 5(3):6–18.

[199] Weigelt, M. and Schack, T. (2010). The development of end-state comfort planning in preschool children. *Experimental Psychology*, pages 476–482.

[200] Weisler, A. and McCall, R. R. (1976). Exploration and play: Resume and redirection. *American Psychologist*, 31(7):492–508.

[201] Welder, A. N. and Graham, S. A. (2001). The influence of shape similarity and shared labels on infants' inductive inferences about nonobvious object properties. *Child development*, 72(6):1653–1673.

[202] Weng, J. (2004). Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, 1(02):199–236.

[203] Wilcox, T. (1999). Object individuation: Infants' use of shape, size, pattern, and color. *Cognition*, 72(2):125–166.

[204] Wörgötter, F., Agostini, A., Krüger, N., Shylo, N., and Porr, B. (2009). Cognitive agents—a procedural perspective relying on the predictability of object-action-complexes (oacs). *Robotics and Autonomous Systems*, 57(4):420–432.

[205] Yuille, A. L. and Liu, C. (2018). Deep nets: What have they ever done for vision? *arXiv preprint arXiv:1805.04025*.

[206] Zhang, G., Patuwo, B. E., and Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62.

Appendix A

Low-Level System Architecture

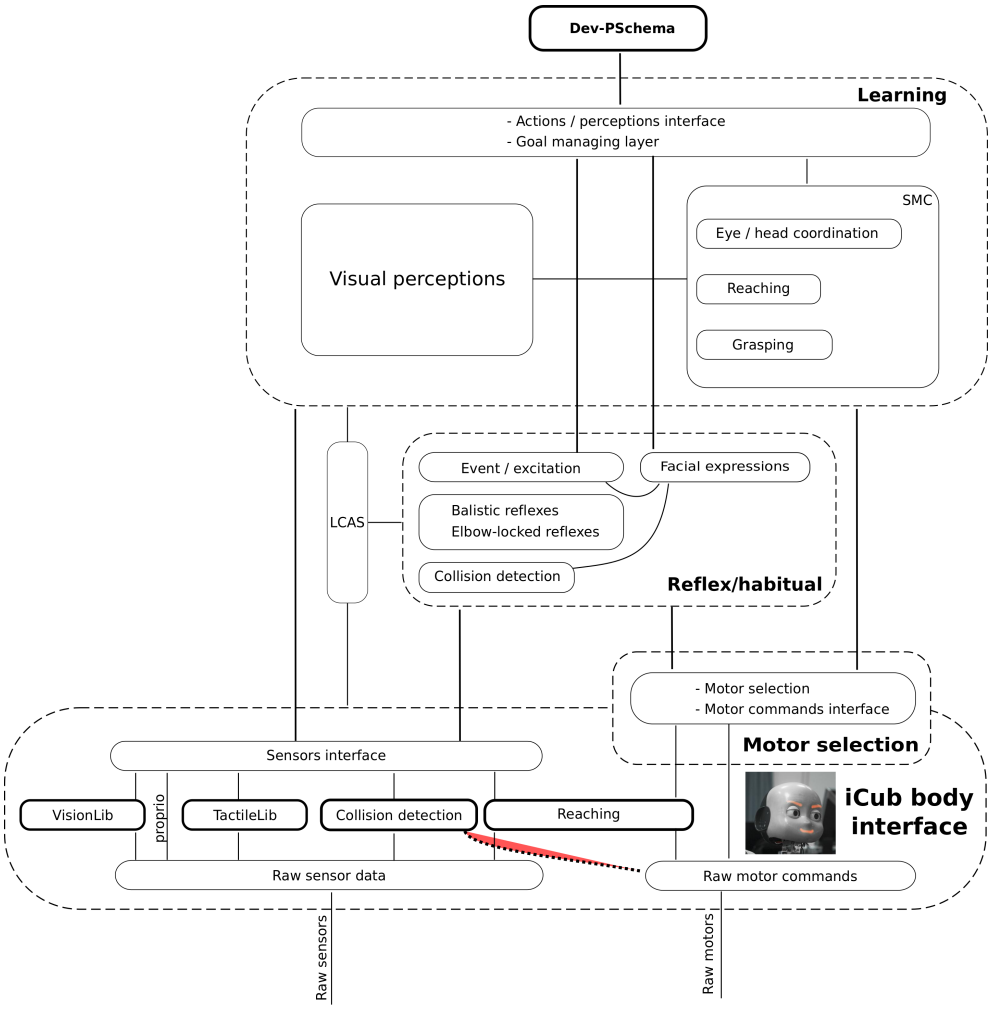


Fig. A.1 Low-level system architecture and the connection with Dev-PSchema.