UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

NATAN DE ALMEIDA JUNGES

LEARNING PRAGMATIC FRAMES IN A COMPUTATIONAL ARCHITECTURE

CURITIBA

2022

NATAN DE ALMEIDA JUNGES

LEARNING PRAGMATIC FRAMES IN A COMPUTATIONAL ARCHITECTURE

Aprendendo Pragmatic Frames em uma Arquitetura Computacional

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Gustavo Alberto Giménez Lugo

CURITIBA

2022

NATAN DE ALMEIDA JUNGES

LEARNING PRAGMATIC FRAMES IN A COMPUTATIONAL ARCHITECTURE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 14/dezembro/2022

Cédric Colas
Doutor
Massachusetts Institute of Technology

Christian Daniel Von Lucken Martinez
Doutor
Universidad Nacional de Asunción

Cesar Tacla
Doutor
Universidade Tecnológica Federal do Paraná

João Fabro
Doutor
Universidade Tecnológica Federal do Paraná

CURITIBA

2022

ACKNOWLEDGEMENTS

Gostaria de agradecer a todas as pessoas que estiveram comigo nessa jornada, mostrando o caminho e dando apoio sempre que necessário. Aos meus pais pelo incentivo e suporte constantes. À toda minha família por estar ao meu lado nos momentos difíceis. Ao meu orientador, Prof. Dr. Gustavo Alberto Giménez Lugo, pelas perguntas instigantes. E a todos os professores com quem tive o prazer de aprender.

RESUMO

Ainda permanece um mistério como as interações sociais podem ser aprendidas tão cedo no processo de desenvolvimento humano. Procurando entender melhor como as interações sociais funcionam e como elas emergem na infância, Bruner (1983) apresentou o conceito de Pragmatic Frame (PF) (*enquadramento pragmático*, em tradução livre), uma forma de descrever e estudar as interações sociais. Este trabalho aborda parte do mistério de aprender um PF, propondo uma taxonomia e uma representação computacional de PFs, que são usadas em uma solução para um agente artificial aprender uma parte específica de um PF através de FAMA (Aineto, Jiménez Celorrio e Onaindia (2019), Aineto, Jiménez e Onaindia (2019)), uma abordagem de aprendizado de modelo de ação. Esta solução é implementada e testada com experimentos que tentam verificar sua viabilidade.

Palavras-chave: pragmatic frames; representação de pragmatic frame; aprendizado de pragmatic frame; aprendizado de modelo de ação.

ABSTRACT

It still remains a mystery how social interactions can be learned so early in the human development process. Trying to understand better how social interactions work and how they emerge in infancy, Bruner (1983) presented the concept of a Pragmatic Frame (PF), a way to describe and study social interactions. This work addresses part of the mystery of learning a PF, proposing a taxonomy and a computational representation of PFs, which are used in a solution for an artificial agent to learn a specific part of a PF through FAMA (Aineto, Jiménez Celorrio e Onaindia (2019), Aineto, Jiménez e Onaindia (2019)), an action model learning approach. This solution is implemented and tested with experiments that try to assess its viability.

Keywords: pragmatic frames; pragmatic frame representation; pragmatic frame learning; action model learning.

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF GRAPHS

LIST OF BOARDS

LIST OF ABBREVIATIONS AND ACRONYMS

Pseudo-Acronyms

CI1	Confirm Induction 1
CI2	Confirm Induction 2
CIR	Confirm Induced Referent
CR	Confirm Referent
CW1	Confirm Word 1
CW2	Confirm Word 2
EEG	Electroencephalogram
EI	Emotional and Inner state
FO	Fully-Observable
FSD	First Simple Detour
LD-DFS	Limited-Depth Depth-First-Search
NO	Non-Observable
PDDL	Planning Domain Definition Language
PF	Pragmatic Frame
PO	Partially-Observable
PO*	Partially-Observable*
PS	Philosophical/Spiritual
PSTQ	Physical, Spatio-Temporal and Quantitative
RIC1	Request Induction Confirmation 1
RIC2	Request Induction Confirmation 2
RIRC1	Request Induced Referent Confirmation 1
RIRC2	Request Induced Referent Confirmation 2

RRC1	Request Referent Confirmation 1
RRC2	Request Referent Confirmation 2
RWC1	Request Word Confirmation 1
RWC2	Request Word Confirmation 2
SS	Self and Sociality
SSD	Second Simple Detour
STRIPS	STanford Research Institute Problem Solver
TIR	Transmit Induced Referent
TR	Transmit Referent
TW	Transmit Word
WAT	Words As social Tools

CONTENTS

1 INTRODUCTION

Human social skills are very complex, but also very useful. They can be used for learning, for practicing skills, for collaborative work, and are an end by themselves. Language and culture could not be possible without them, and most human technologies are a product of social collaboration. That's not a surprise that humans develop social skills very early on in their development process, as they are distinctive features of our species, although it still remains a mystery how such complex skills can be learned at such early age. Trying to understand better how social interactions work and how they emerge in infancy, Bruner (1983) presented the concept of a Pragmatic Frame (PF), a way to describe and study social interactions between infants and their caretakers. This concept is so useful that it has since been generalized to describe all types of social interactions, between all types of agents.

Although the theory of PFs is very useful to study social interactions, it lacks a more formal representation, necessary for a more precise analysis of the dynamics of social interactions. Vollmer *et al.* (2016) presents a taxonomy for them, which goes some length to address that, but it is limited in what types of PFs it can classify and which aspects it can represent. Trying to address the mystery of how humans can learn PFs, this work proposes another taxonomy, complementary to the one presented in Vollmer *et al.* (2016), and a computational representation of PFs, which is used to show a way that some parts of a PF could be learned.

1.1 Motivation

Of all human cognitive skills, learning is certainly one of the most important, second to social skills. Contemporary research on motor learning in infants and early toddlers has shed a light on how unsupervised learning can take place in an open environment, with computer simulations mimicking several key aspects of this type of learning.

On social learning, which becomes the main type of learning in late infancy/toddlerhood, there has also been quite an advancement, mainly on learning in a social environment. Psychological theories such as *PFs* (Bruner (1983), Rohlfing *et al.* (2016)) and, more recently, Words As social Tools (WAT) (BORGHI *et al.*, 2019), have enabled the understanding of how abilities like simple language (with concrete words and unambiguous syntax) and solo actions, or even more complex ones, like abstract words and joint actions and goals, can be learned in a social context.

But little has been done to understand how this social context is formed and how social interactions and skills are learned. As these interactions and skills form the background against which all social learning takes place, it is *critical* to thoroughly understand their learning.

1.2 Goals

1.2.1 Main goal

The main goal of this work is to provide computational representations of social interactions described by the theory of PFs, and use them to show how a specific part of a PF can be learned. It is hoped that the computational representations and the method presented to learn PFs could be valuable to other areas of Cognitive Sciences, helping to further the understanding of the internal representations of PFs in the mind and how humans learn social interactions.

1.2.2 Specific goals

- To understand the central elements that make up a PF, and how such elements could be learned;
- To understand the relevant dimensions for classifying PFs and how they are learned;
- To propose a method for learning specific elements of a PF;
- To evaluate the computational performance of the proposed method, making use of an experimental scenario that involves more than one PF of the same type and comparing their learning rates and transfer-learning capabilities.

1.3 Approached questions

Some of the questions approached, discussed, or at least related to this work are: How PFs are learned? What are the relevant dimensions to classify how a PF is learned? How can those dimensions affect learning a PF? Can a PF be computationally modeled? Can a robot learn a PF? Is the learning approach used by the robot consistent to how a human learns PFs? What is the performance of this learning approach?

1.4 Contributions

The most important contribution of this work is the computational modeling of PFs, which is essential to the study of PFs in Computing and Robotics contexts, but is also important to other areas of Cognitive Sciences, since there aren't many representations of PFs in those areas either. A taxonomy for PFs, especially one that focuses on the conditions in which a PF is learned, is also a big contribution, shedding a light on PFs and their internals that will enable further investigation in several areas of Cognitive Sciences. Last but not least, an approach that allows learning parts of a PF is a proof-of-concept contribution that can be developed further, with a

big potential to not just improve the autonomy and adaptability of robots, but also to understand better how humans learn and develop social interactions.

2 THEORETICAL CONCEPTS

Some concepts touched by this work come from areas outside Computing, and some come from more specific areas of Computing, all of which a non-specialist researcher or professional might not be familiar with. To enable the understanding of this work, they are briefly covered here, but for a deeper dive into them, further reading of the bibliography is encouraged.

2.1 Agent

> An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**. (RUSSELL; NORVIG, 2020, p. 36)

According to Russell e Norvig (2020, p. 40), the agent will always try to select actions that are expected to yield the best result in some internal or external performance measure (its motivation), based on sensory information and its prior knowledge.

2.2 Environment

From the definition of an agent, it can already be understood the concept of an environment, and the nature of their connection:

> [. . .] the part that affects what the agent perceives and that is affected by the agent's actions. (RUSSELL; NORVIG, 2020, p. 36)

An environment provides the means for **learning** (as a search/exploration space), **communication** (determining how the information is encoded and being the channel where it travels through) and **social interaction**.

2.3 Goal

The notion of goal adopted by Castelfranchi (2014) is that it is:

- A **mental representation**;
- An **alternative description** of the world, not necessarily reflecting what it currently is;
- **Multimodal**, encompassing different levels of abstraction and different cognitive functions;
- Used as **reference** by the agent when modifying the world.

On that notion, a goal is not always directly pursued. It can be dropped in favor of other goals, or even be unpursuable, due to lack of preconditions, means or skills; it can just be used as a reference to evaluate how close/far the world is from it, producing no action to actually modify the world; it can be pursued by doing nothing, except waiting, wishing or hoping; or it can already have been achieved.

2.4 Planning

> Planning is the art and practice of thinking before acting (HASLUM, 2006, p. 1)

According to Haslum (2006, p. 1), planning involves determining the applicable actions in a certain state, being able to predict their outcomes, and using this information to find an optimal action plan to achieve specific goals.

Following the definitions presented by Aineto, Jiménez Celorrio e Onaindia (2019), a **planning frame**, which determines the exploration space of a planning task, consists of a set of logical **fluents** (or *propositional variables*) F, used to express the agent's beliefs about the environment (i.e. its **state**), and a set of **actions** A, that limits the possible actions the agent can apply to the environment. The fluents of F are instantiated from **predicates** about the **objects** present in the environment. The actions of A are instantiated from **action schemas** that, much like predicates, receive the objects present in the environment as parameters.

The actions of A have **preconditions** and **effects** defined as partial assignments of values to the fluents of F. Therefore, an action can only be applied if all of its preconditions are met in the current state, and when it is applied, its effects are applied to the current state.

When combined with an **initial state** I and a **goal condition** G, a planning frame becomes a **planning problem**, for which a **plan** (a sequence of actions from A) can actually be computed. Such plan solves this problem when G is satisfied at its end.

2.4.1 Action model learning

Most problems in Computing consist of *input data* and a *program* as inputs, for which an *output* must be found. But action model learning, much like any other machine learning problem, consists of *input data* (the initial state and the goal condition) and its respective *output* (the plan, or rather a *plan trace*) as inputs, for which a *program* (the actions, their preconditions and effects) must be found.

According to Aineto, Jiménez Celorrio e Onaindia (2019), a **plan trace** is an *observation* of a *plan execution*, containing a sequence of applied actions and its state trajectory. Both action sequence and state trajectory can have different degrees of **observability** in a plan trace, for which different approaches might be required. The action sequence can be:

- **Fully-Observable (FO)**, when *all* the actions applied in the plan execution are observable in the plan trace.
- **Partially-Observable (PO)**, when *some* (but not all) of the actions applied in the plan execution are observable in the plan trace.
- **Non-Observable (NO)**, when *none* of the actions applied in the plan execution are observable in the plan trace.

The state trajectory can be:

- **Fully-Observable (FO)**, when *all* of the states reached in the plan execution are fully-observable (i.e. all are full assignments of values to the fluents of F) in the plan trace.
- **Partially-Observable (PO)**, when *some* (possibly none) of the states reached in the plan execution are fully-observable (i.e. some are partial assignments of values to the fluents of F) in the plan trace.
  - **Partially-Observable* (PO*)**, when *all* of the states reached in the plan execution are at least partially-observable (i.e. none are empty) in the plan trace.
  - **Non-Observable (NO)**, when *none* of the states reached in the plan execution are at least partially-observable (i.e. all are empty) in the plan trace.

A STanford Research Institute Problem Solver (STRIPS) **action model** is an action schema that contains the action name, its parameters, its preconditions ($pre(\xi)$), positive ($add(\xi)$) and negative ($del(\xi)$) effects. Such model must meet three requirements: a predicate can only become false if it originally was true ($del(\xi) \subseteq pre(\xi)$); a predicate cannot become both false and true ($del(\xi) \cap add(\xi) = \varnothing$); and a predicate can only become true if it originally was false ($pre(\xi) \cap add(\xi) = \varnothing$).

The set of action models learned by an agent constitutes a **domain model**. Combining an initial domain model $\mathcal{M}$ with a set of plan traces $\mathcal{T}$ results in a **learning task**, which produces a new domain model $\mathcal{M}'$, consistent with $\mathcal{M}$ and $\mathcal{T}$.

3 PRAGMATIC FRAMES AND ACTION MODEL LEARNING: TAXONOMIES AND COMPUTATIONAL MODELS

> A format is a standardized, initially microcosmic interaction pattern between an adult and an infant that contains demarcated roles that eventually become reversible. (BRUNER, 1983, p. 120)

Pragmatic Frames (PFs), also known as *interactional formats* or *frames* (BRUNER, 1983, pp. 121, 122), determine the communication between the adult and the infant, and how such communication provides information to parameterize their **actions**[1]. They have a script-like structure, containing the **sequence** of actions to take, the **selection** of which actions to take, and how/when actions should be **repeated**.

Much like a language, a PF has a **deep** structure, that is the invariant structure of the interaction, and a **surface** structure, that comprises all the different forms that the deep structure can be realized as. According to Rohlfing *et al.* (2016), a PF also consists of **syntax**, that accounts for its apparent structure, and **meaning**, that encompasses all the cognitive functions involved in it.

PFs are **emergent** by nature, requiring repetitive exposure to be learned. As the infant gets more familiar with them, they become more stable, only varying in their **learning content**, such that the infant can very quickly identify the learning content and extract it. Once learned, they can be used as **modules** to compose more complex social interactions, or **abstracted** and applied to a broader range of situations where they were never observed before, serving as powerful tools to scaffold an infant's development.

Due to their emergent nature, PFs provide powerful means to learn from social interactions, **framing** the learning content in a stable **practical** structure (hence the name). An infant can leverage this structure to **ground** their knowledge of **language**, constraining its use in a **familiar** syntactic structure and extracting its meaning from an also familiar semantic structure, all realized in a practical interaction experience.

A classical example of PF is the book reading frame, given by Bruner (1983). In such a frame, a parent reads a picture book to their child, teaching labels for the pictures. First, the parent points to a picture and says, "Look!", directing the child's attention to it. Then they say, "What's that?", indicating a label for the picture is expected. Depending on their maturity and engagement, the child might attempt an answer, but either way the parent will provide feedback of the respective label, by saying "Yes, it's a..." or "No, it's a...".

Although the definition of a PF was originally applied to the adult-infant interaction, due to its great potential of representing social interactions, it has since been generalized to interactions between any types of agents, be it adult-adult, adult-infant, infant-infant, human-human, human-robot or even robot-robot.

[1] Actions should not be interpreted only as physical, as it may encompass calculations, reasoning, imagination etc., all of which may well be internal to the agent's mind.

One example of a human-robot PF is a cooperative construction frame, given by Lallee *et al.* (2010) and studied by Vollmer *et al.* (2016). In such a frame, both human and robot have to build a table together, with the human screwing the parts together and the robot holding the structure and handing the required parts. First, the robot signals it is ready to start. Then the human tells it to start learning, provides a sequence of commands, and ends with a respective signal. For each command in the sequence, the robot asks confirmation, and the human can correct it if it's wrong. After a command is confirmed, the robot executes it and waits for the next. Once the sequence of commands is finished, the robot stores it so it can be reused when requested by the human.

To further the understanding of PFs, as well as the PFs proposed in this work, a taxonomy for PFs is presented here. Another taxonomy is also proposed in section 4.1. The action model learning approach chosen for this work is also presented here, covering the motivation of such choice and how it works.

3.1 Taxonomy of Vollmer *et al.* (2016)

This taxonomy is appropriate for this work for three reasons. First, it is intended for PFs for learning, which is the type of PFs this work is interested in. Second, it encompasses human-robot interactions, fundamental for a modern computational perspective of PFs. Third, although it does not provide dimensions to classify how learning a PF takes place, it was developed with that in mind.

The two main aspects that this taxonomy aims to categorize are the **interactional characteristics** of a PF for learning (THOMAZ; BREAZEAL, 2006) and its underlying **learning mechanisms** (CUAYÁHUITL, 2015).

Thomaz e Breazeal (2006) categorized the interactional characteristics from a human-robot interaction perspective, which fits well with this work but needs some adaptation for a proper human-human interaction analysis. They propose three dimensions (their names are adapted for this work): **training explicitness** (*"Is the system passively observing the performance of a human or is a human teacher teaching the robot?"*, Vollmer *et al.* (2016)), **interaction leading** (*Is the robot or the human who leads the interaction?*) and **exploration autonomy** (*Is there a human guiding the robot's exploration?*).

Cuayáhuitl (2015) proposes only one dimension (which in this work is called **learning modality**) with four possible values: **supervised learning**, **reinforcement learning**, **unsupervised learning** and **active learning**. Vollmer *et al.* (2016) regroups them differently, from a more human-robot interaction perspective:

- **Passive learning**, where the robot observes the human's behavior, without querying the human for information. It includes both supervised and unsupervised learning.

- **Exploration learning**, where the robot explores the environment to learn. It includes reinforcement learning, and can be divided into two overlapping subcategories:
  - **With initial tutor demonstration**, where the reward function is initialized from observation of the human's behavior.
  - **With tutor refinement**, where the rewards originate from the feedback provided by the human.
- **Active learning**, where the robot queries the human for information, trying to maximize its information gain.

To exemplify how this taxonomy is used and what its dimensions mean, two examples of PFs classified by this taxonomy were selected from Vollmer *et al.* (2016).

3.1.1 Classification of Akgun *et al.* (2012)

In this PF, a robot learns goal-oriented and means-oriented movements in an unsupervised manner, where the human tutor moves the robot's arm to show key positions of a desired movement. The tutor can start a new demonstration or review the previous one, navigating through the recorded key positions, modifying them, adding new ones and removing existing ones.

Respective to the interactional characteristics, this PF's training is explicit, as the human tutor is explicitly teaching the robot how to perform the desired movement; its interaction is led by the human, who starts, ends and makes all decisions in the interaction; the robot does not have any autonomy to explore, as it only learns from explicit tutor demonstrations. Respective to the learning mechanisms, this PF's learning modality is passive, as the learning is unsupervised and the robot neither explores the environment nor queries the human tutor to learn.

3.1.2 Classification of Grizou, Lopes e Oudeyer (2013), Grizou *et al.* (2014)

These works present two similar PFs, hence they are analyzed together. In the first one, the robot learns to stack blocks. In the second, the robot learns how to move in a grid world based on Electroencephalogram (EEG) signals. In both cases, a sequential task and a feedback-to-meaning mapping is learned. First, the robot explores the environment, then the human tutor provides feedback. The feedback can be either human-led, when the tutor notices the robot is doing something wrong, or robot-led, when the robot requests feedback after it finished exploring. The words used to provide feedback are not known a priori by the robot, so it needs to learn how to associate them to the known meanings.

Respective to the interactional characteristics, these PF's training is explicit, as the human tutor is explicitly providing feedback to the robot; its interaction can be led by either the

human or the robot; the robot has autonomy to explore, but the feedback from the tutor might restrict it. Respective to the learning mechanisms, these PF's learning modality is exploration with tutor refinement, as the robot explores the environment to learn but also depends on the tutor's feedback.

This example is interesting because, much like this work, it shows PFs which the robot does not know entirely a priori, having to learn some of their aspects as the interactions develop.

3.2 Action model learning with FAMA

The chosen approach for action model learning was FAMA (Aineto, Jiménez Celorrio e Onaindia (2019), Aineto, Jiménez e Onaindia (2019)), as it allows learning STRIPS action models even when there is minimal observability of both states (NO) and actions (NO). It does so by compiling the action model learning problem into a classical planning problem. Then a regular planner can solve it, and the learned action models be extracted.

The compilation process starts by encoding all the predicates that can be inserted into each action model as propositional variables. Each variable encodes in its name the action name, the predicate name along with its parameters, and whether it is to be inserted as a precondition, a positive or a negative effect of said action. For example, if there was an action named $move$ with parameters $v1, v2$ and a predicate $topleft(v1)$, inserting it as a precondition could be encoded as $pre\_move\_topleft\_v1$. If there was another predicate $bottomright(v2)$, inserting it as a negative effect could be encoded as $del\_move\_bottomright\_v2$. Each of those propositional variables are used to encode how an action model is being built, and FAMA uses them to extract the action models at the end of the process.

For each pair of action model and predicate, two actions are added to the domain model that insert the respectively encoded propositional variables into the current state, one for the precondition and one for the effects. Their preconditions and effects ensure the STRIPS requirements are met for the action models being built. In the examples above, an action $insert\_pre\_move\_topleft\_v1$ would insert $pre\_move\_topleft\_v1$, and an action $insert\_eff\_move\_bottomright\_v2$ would insert either $del\_move\_bottomright\_v2$ or $add\_move\_bottomright\_v2$, depending on the STRIPS requirements met in the current state. Those actions allow the planner to explore the search space of domain models to find a candidate $\mathcal{M}'$.

After finding a candidate domain model, the planner needs to be able to validate it, that is, to ensure that it satisfies the initial model $\mathcal{M}$ and the set of plan traces $\mathcal{T}$. For this, one action is added to the domain model for each action model learned, which applies the respective action model to the current state, respecting its preconditions and effects. In the examples above, an action $apply\_move$ would apply the learned action model for $move$. For each observed state in $\mathcal{T}$, an action is also added to validate that the applied actions produce the respective observed

state. If there were 3 observed states in $\mathcal{T}$, $validate\_0$, $validate\_1$ and $validate\_2$ would validate each of them respectively.

The planner is only able to reach the end goal of this planning problem if, after proposing a candidate domain model $\mathcal{M}'$, it is able to reproduce the traces in $\mathcal{T}$, validating all observed states when applying all observed actions (other actions might be added to the solution, as long as the observed states are validated). Otherwise, the planner will have to propose another candidate model, and this repeats until a model is found that can be validated.

3.3 Moving forward

Given a proper explanation of what PFs are, their structures, their features, how they enable learning and how to classify them, along with an understanding of how the FAMA action model learning approach works, a solution to the problem of learning PFs can be proposed.

4 SOLUTION PROPOSAL: A PRAGMATIC FRAME REPRESENTATION

In this work, in addition to using a taxonomy already available in the literature, a new taxonomy for PFs is also proposed. To be able to propose a computational solution to the problem of learning a PF, a computational model for PFs is proposed, as well as an example PF, which is used in the solution implementation and experiments. Then the solution to learning a PF is proposed, integrating all the concepts, taxonomies and models presented so far.

4.1 Proposed taxonomy

This taxonomy mostly serves the same purposes as the others available in the literature (listed below), but it focuses on aspects of a PF that are relevant to this work. Neither of those taxonomies are comprehensive, and they are complementary to each other. In future work, they should be expanded for a more complete description of PFs.

4.1.1 Rationale

Creating a taxonomy for PFs serves 5 main purposes:

- To improve their understanding and explaining, furthering their study in an orderly and systematic fashion;
- To enable their direct comparison, allowing the understanding of the implications of each difference or similarity between them, which otherwise would be impractical;
- To enable their identification, which is essential in a multi-interaction context such as an open environment, where the agents must be able to identify which interaction they are engaging in, as well as their role in it, so they can play it;
- To enable their building in contexts where creativity, adaptability or flexibility are required, such as open environments or when humans are involved;
- To enable their learning, which takes the agent's adaptability and flexibility to a whole new level, improving their social skills while leveraging them to share knowledge.

The proposed solution was developed taking only one type of PF into account, but it could be adapted to leverage this taxonomy to be able to potentially learn any kind of PF.

4.1.2 Dimensions

In this proposed taxonomy, there are 4 main dimensions considered important to classify a PF: *Interaction purpose/goal*, which encodes the (meta-)purpose of the interaction; *Interaction*

development stage, which encodes how well developed is the interactional structure; *Interaction competence balance*, which encodes how distant or how close are the interaction levels of competence of the agents; and *Interaction dynamics*, which encodes how the interaction structure changes over time.

The *Interaction purpose/goal* dimension has 5 possible values: *Learning*, which means learning not related to social dynamics; *Skill practicing*, which means to practice the already-learned skills (not related to social dynamics); *Social dynamics assertion/role playing*, which means to learn or to practice the social dynamics of a social group; *Social bonding*, which means to establish or to maintain social bonds; and *Joint task execution*, which means to execute coordinated actions to reach a common goal, not related to learning, skill practicing or social activities.

The *Interaction development stage* dimension has 4 possible values: *Emerging*, which means it is being built by the agents and there is no clear definition or consensus; *Negotiating*, which means its definition is becoming clearer and consensus is being built between the agents; *Exchanging*, which means it is already well defined and reached wide consensus, being transmitted to new agents; and *Established*, which means all the interacting agents reached high competence in the interaction, with transmission to other agents no longer being necessary.

The *Interaction competence balance* dimension has 2 possible values: *Symmetrical*, which means their competence levels are very close; and *Asymmetrical*, which means their competence levels are very distant.

The *Interaction dynamics* dimension has 4 possible values: *Static*, which means it does not change; *Scaffolding*, which means it increases its level of complexity/difficulty but does not increase its size too much; *Branching*, which means it opens up more paths of relatively the same level of complexity; and *Sidetracking*, which means it transitions to different (but related) PFs or sub-PFs.

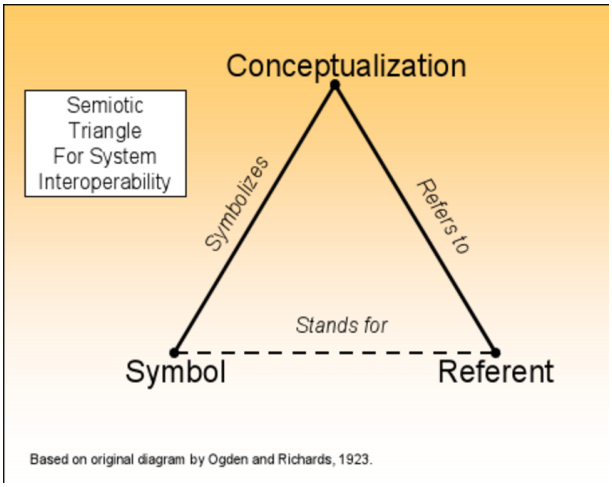
When the *Interaction purpose/goal* is *Learning*, 5 additional dimensions are considered: *Learning modality*, which encodes the domain, type or modality of the knowledge to be learned; *Abstraction level*, which encodes the level of abstraction of the knowledge to be learned (VILLANI *et al.*, 2019); *Knowledge type*, which encodes the type of the knowledge to be learned, expressed as an association pair between two of the three corners of the semiotic triangle (OGDEN; RICHARDS, 1989), depicted in Figure 1; *Knowledge development stage*, which encodes how mature is the knowledge to be learned; and *Knowledge competence balance*, which encodes how distant or how close are the knowledge levels of competence of the agents.

The *Learning modality* dimension has 8 possible values: *Sensory*, which means the sensory modality; *Motor*, which means the motor modality; *Social*, which means the social domain; *Cultural/religious*, which means the cultural and religious domain; *Linguistic*, which means the linguistic type; *Technical*, which means the technical type; *Scientific*, which means the scientific type; and *Mathematical/logical*, which means the mathematical and logical type.

The *Abstraction level* dimension has 5 possible values: *Concrete*, which refers directly to sensory and motor experience; *Physical, Spatio-Temporal and Quantitative (PSTQ)*, which refers to physical notions, quantifiable bodily sensations, quantities, numbers and operations, quantifiable temporal concepts and spatial concepts; *Emotional and Inner state (EI)*, which refers to emotions, at different levels of complexity, mental states, emotionally connoted social situations and characteristics of the self with respect to others; *Self and Sociality (SS)*, which refers to psychological and physical characteristics of the self, mainly with a positive connotation, of social situations regulated by norms, together with concepts related to social institutions, and to social situations characterized by the presence of more people; and *Philosophical/Spiritual (PS)*, which refers to imagery entities, religious words, principles, disciplines, concepts linked to argumentation, reasoning and decision making, and mainly negatively connoted words, related to characteristics of the self.

The *Knowledge type* dimension has 6 possible values: *Symbol-referents*, which means the referents associated to a symbol; *Symbol-concepts*, which means the concepts associated to a symbol; *Referent-symbols*, which means the symbols associated to a referent; *Referent-concepts*, which means the concepts associated to a referent; *Concept-symbols*, which means the symbols associated to a concept; and *Concept-referents*, which means the referents associated to a concept.

Figure 1 – Semiotic triangle for system interoperability



Source: Turnitsa e Tolk (2008).

The *Knowledge development stage* dimension has 3 possible values: *Emerging*, which means it is being built by the agents and there is no clear definition or consensus; *Negotiating*, which means its definition is becoming clearer and consensus is being built between the agents; and *Exchanging*, which means it is already well defined and reached wide consensus, being transmitted to new agents.

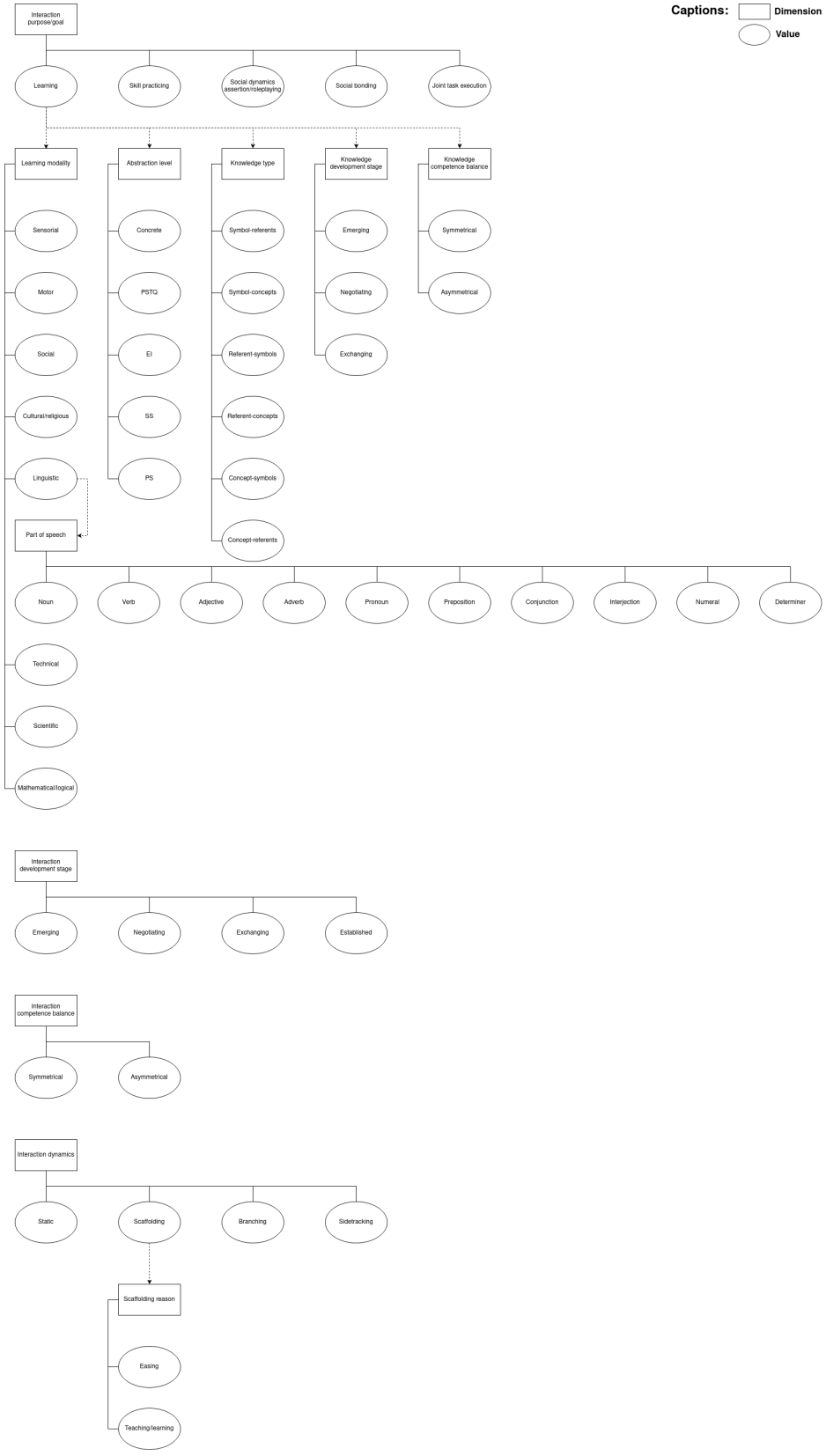
The *Knowledge competence balance* dimension has 2 possible values: *Symmetrical*, which means their competence levels are very close; and *Asymmetrical*, which means their competence levels are very distant.

When the *Learning modality* is *Linguistic*, 1 additional dimension is considered: *Part of speech*, which encodes the part of speech the knowledge plays the role of. The *Part of speech* dimension has 10 possible values: *Noun*, *Verb*, *Adjective*, *Adverb*, *Pronoun*, *Preposition*, *Conjunction*, *Interjection*, *Numeral* and *Determiner*.

When the *Interaction dynamics* is Scaffolding, 1 additional dimension is considered: *Scaffolding reason*, which encodes what scaffolding is used for. The *Scaffolding reason* dimension has 2 possible values: *Easing*, which means to assess and to synchronize the levels of competence between the agents; and *Teaching/learning*, which means to transmit an interaction to a new agent.

Figure 2 has a graphical representation of all the dimensions, their values and how they relate to each other. To exemplify how this taxonomy is used and what its dimensions mean, the same two examples of PFs presented in section 3.1 were selected to be classified by this taxonomy, allowing the two taxonomies to be compared. The classifications of the other PFs presented in Vollmer *et al.* (2016) are available in Appendix A.

Figure 2 – Dimensions and values of the proposed taxonomy



Source: Own authorship (2022).

4.1.3 Classification of Akgun *et al.* (2012)

The interaction purpose/goal is learning. The interaction development stage is established, as both robot and tutor have high competence in the interaction. The interaction competence balance is symmetrical, as robot and tutor have the same level of competence in the interaction. The interaction dynamics is sidetracking, as the interaction is divided into two separate but related PFs, one where a new demonstration is added and another where the previous demonstration is reviewed.

The learning modalities are motor and technical, as the robot not only learns the motor positions required to perform a movement, but also which goals and means can be achieved by it. The abstraction levels are concrete and PSTQ, as it refers directly to motor experience (the key positions of each movement), as well as physical notions and spatial concepts (the positions are relative to each other). The knowledge type is symbol-referents, as the robot learns how to physically realise a movement (the referent) it already has a symbol for. The knowledge development stage is negotiating, as the tutor already has a notion of the desired movement, but only reasons about how to physically realise it once they actually use the robot's arm to perform it. The knowledge competence balance is asymmetrical, as the robot does not know a priori anything about the desired movements.

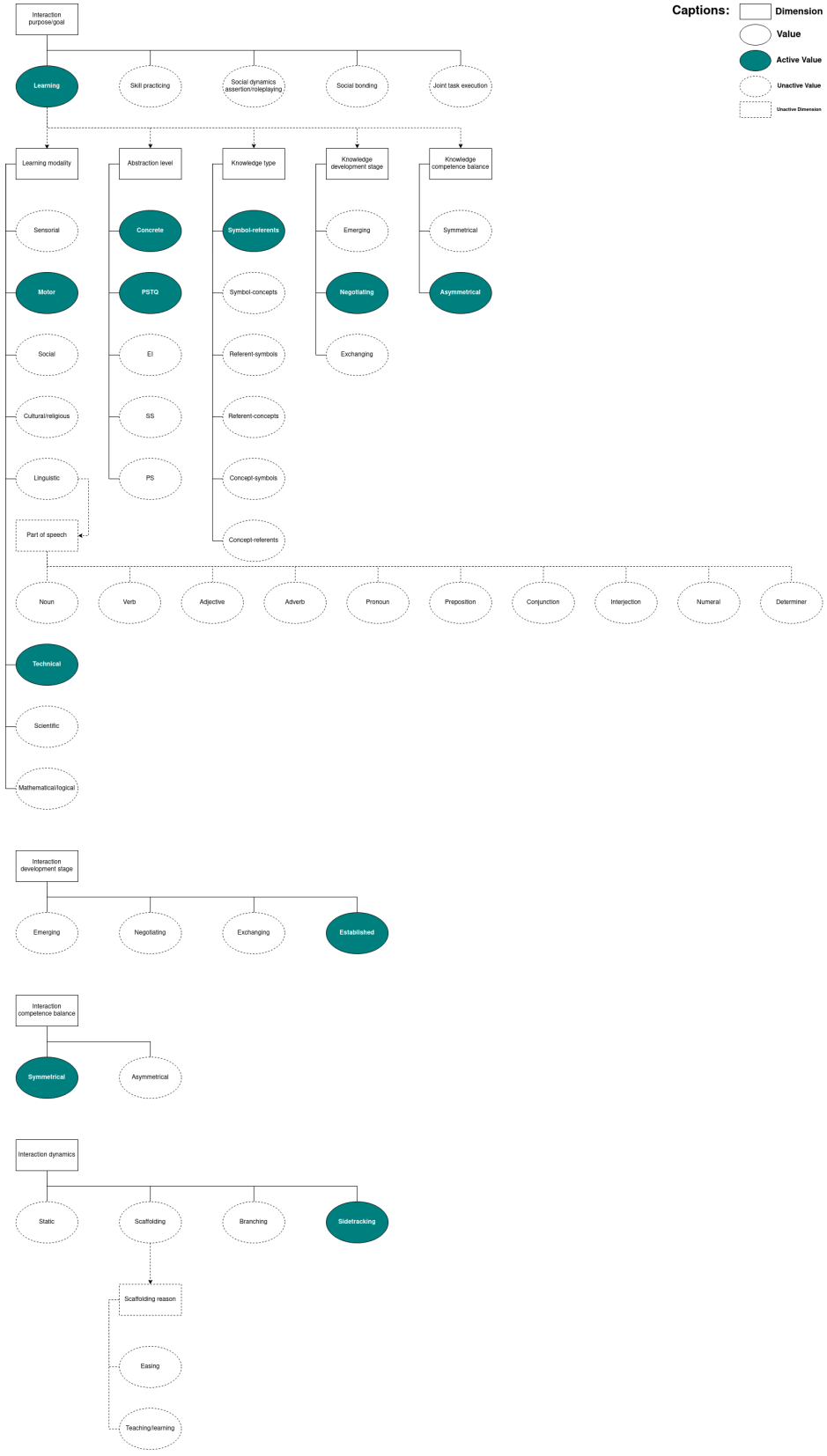
4.1.4 Classification of Grizou, Lopes e Oudeyer (2013), Grizou *et al.* (2014)

The interaction purpose/goal is learning. The interaction development stage is exchanging, as the robot needs to learn the syntax of the feedback signals chosen by the tutor. The interaction competence balance is asymmetrical, as the robot does not know a priori anything about the syntax of the feedback signals. The interaction dynamics is scaffolding, as the robot initially not knowing some aspects of the PF forces the interaction to be simpler, getting more complex as the robot learns them.

The learning modality is technical, as the robot learns how (the techniques) to achieve established goals. The abstraction level is PSTQ, as it refers to physical notions, quantities (how many blocks are in the stack, how many blocks can be stacked, how far the goal is), operations (add/remove a block to/from the stack, move in a direction) and spatial concepts (which block is on top of which, which direction the goal is in). The knowledge type is symbol-referents, as the robot learns how to physically realise a movement (the referent) to achieve a given goal (the symbol). The knowledge development stage is exchanging, as the tutor has it well defined and only transmits it to the robot through feedback. The knowledge competence balance is asymmetrical, as the robot does not know a priori anything about the desired movements.

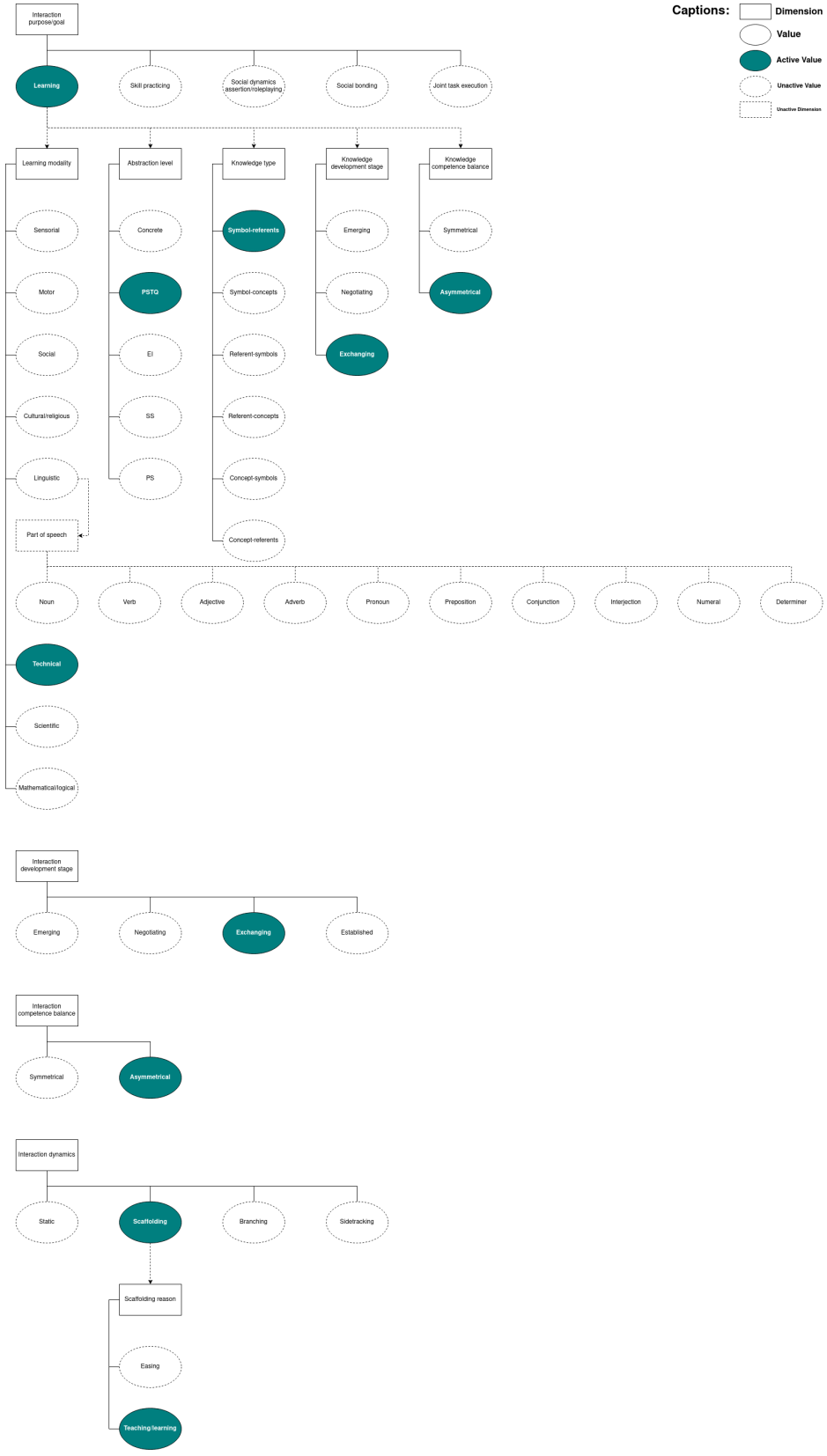
The scaffolding reason is teaching/learning, as the robot needs to learn the syntax of the feedback signals.

Figure 3 – Classification of the PF in Akgun *et al.* (2012)



Source: Own authorship (2022).

Figure 4 – Classification of the PFs in Grizou, Lopes e Oudeyer (2013), Grizou *et al.* (2014)



Source: Own authorship (2022).

4.2 Computational model of Pragmatic Frames

As PF is a theory from Psychology and this work approaches it from a Computing perspective, modeling the relevant aspects of such theory to a language understandable by Computer researchers and professionals is essential. Only once a computational model is available that a proper understanding of the problem for which this work proposes a solution is possible.

4.2.1 Pragmatic Frames as state machines

In this work, when modeling a PF as a state machine, some restrictions are imposed:

- The interaction must be between two agents;
- Each agent must perform a single role throughout the entire interaction;
- The agents must take turns to perform their actions;

Those restrictions greatly simplify the state machine, making the problem of learning some of its components much more tractable. In such machine, each state transition represents an action that, as long as its preconditions are met, can be performed by a specific agent in a specific state of the interaction, producing effects and moving the interaction to another state. Each state then represents a moment in the interaction where one of the agents decides which action to take next.

4.2.2 Pragmatic Frames as planning frames

Turning the state machine previously described into a planning frame is quite straightforward. Each action represented by a state transition can be directly modeled as an action in the planning frame, with its preconditions and effects expressed in terms of propositional variables describing the environment. The state of the interaction can then be expressed in terms of those same variables. Once a PF is modeled as a planning frame, learning the actions in such frame can be done using action model learning methods.

4.3 Proposed Pragmatic Frame

This work presents two complementary PFs, each modeling a part of a two-part interaction. In such interaction, the robot learns names of objects present in the environment by interacting with a human. In the first part, the robot wants to learn the name of an object that both agents see. In the second part, it wants to know other objects that have the same names it already learned. In this interaction, one agent plays the role of learner (robot) and the other,

the role of teacher (human, or rather a simulation of one). They interact by text, with the syntax described in Frame 1 and Frame 2. The interaction has in total 17 actions, whose algorithms are presented in Appendix B.

Frame 1 – First part of the interaction

Action	**Current Agent**	**Syntax**	**Information Slot**	**Depends On**	**Description**
Transmit Referent (TR)	Learner	Integer	o_{i1}		It tries to establish a common referent by representing the index of an object in the environment.
Request Referent Confirmation 1 (RRC1)	Teacher	Integer + "?"	o_i	o_{i1}	It requests confirmation through feedback (repeating the index and appending a question mark).
Request Referent Confirmation 2 (RRC2)	Learner		o_{i2}	o_i, o_{i1}	It confirms in case the referent is successfully established, otherwise returns to action *TR*.
Confirm Referent (CR)	Learner	"True"			It confirms the referent was successfully established.
Transmit Word (TW)	Teacher	String	w_1	o_i	It represents a word that corresponds to the referent.
Request Word Confirmation 1 (RWC1)	Learner	String + "?"	w	w_1	It requests confirmation through feedback (repeating the word and appending a question mark).
Request Word Confirmation 2 (RWC2)	Teacher		w_2	w, w_1	It confirms in case the word is successfully repeated, otherwise returns to action *TW*.
Confirm Word 1 (CW1)	Teacher	"True"			It confirms the word was successfully repeated.
Confirm Word 2 (CW2)	Learner			o_{i1}, w	It adds the pair $\langle o_{i1}, w \rangle$ to its knowledge base.

Source: Own authorship (2022).

The first part of the interaction, with an example shown in Figure 7, starts with the referent loop, whose purpose is to establish a common referent between the two agents. This loop execution is determined by whether or not the referent information was successfully transmitted. If it was, the interaction continues. If not, the loop repeats. It works as such because the rest of the interaction depends on that information, allowing or prohibiting it to continue. The loop ends when the information transmission is confirmed.

The interaction continues with the word loop, whose purpose is to establish a common word to name the referent. It can only proceed once the referent is already established, since the teacher needs that information to choose a word. This loop execution is determined by whether or not the word information was successfully transmitted. Once both the referent and the word are established, the interaction can continue for the association of the two to be learned.

The second part of the interaction, with an example shown in Figure 8, starts with the referent loop, whose purpose is the same as in the first part. The interaction continues with the word loop, whose purpose is also the same as in the first part. The teacher needs to decide

Frame 2 – Second part of the interaction

Action	Current Agent	Syntax	Information Slot	Depends On	Description
Transmit Induced Referent (TIR)	Learner	Integer	o_{i1}		It tries to establish a common referent by representing the index of an object in the environment.
Request Induced Referent Confirmation 1 (RIRC1)	Teacher	Integer + "?"	o_i	o_{i1}	It requests confirmation through feedback (repeating the index and appending a question mark).
Request Induced Referent Confirmation 2 (RIRC2)	Learner		o_{i2}	o_i, o_{i1}	It confirms in case the referent is successfully established, otherwise returns to action *TIR*.
Confirm Induced Referent (CIR)	Learner	"True"			It confirms the referent was successfully established.
Request Induction Confirmation 1 (RIC1)	Learner	String + "?"	w		It requests confirmation of the hypothesis that the referent can be referred to by some word it already knows, by representing the word and appending a question mark.
Request Induction Confirmation 2 (RIC2)	Teacher		w_2	w, o_i	It confirms in case the hypothesis is true, otherwise it starts a new interaction (first part) from action *TW*.
Confirm Induction 1 (CI1)	Teacher	"True"			It confirms the hypothesis was true.
Confirm Induction 2 (CI2)	Learner			o_{i1}, w	It adds the pair $\langle o_{i1}, w \rangle$ to its knowledge base.

Source: Own authorship (2022).

whether the chosen word can be used to refer to the referent. This loop execution is determined by whether or not the word matches the referent, and if that is not the case, the word loop of the first interaction is used instead.

4.3.1 Classification by the taxonomy of Vollmer *et al.* (2016)

Respective to the interactional characteristics, this PF's training is explicit, as the human teacher (or a simulation) explicitly provides the robot learner with the words for each object; its interaction is led by the robot, who starts, ends and makes most decisions in the interaction; the learner has autonomy to explore, as the learner chooses autonomously which information to learn next. Respective to the learning mechanisms, this PF's learning modality is exploration with initial tutor demonstration in the first part of the interaction, and tutor refinement in both parts.

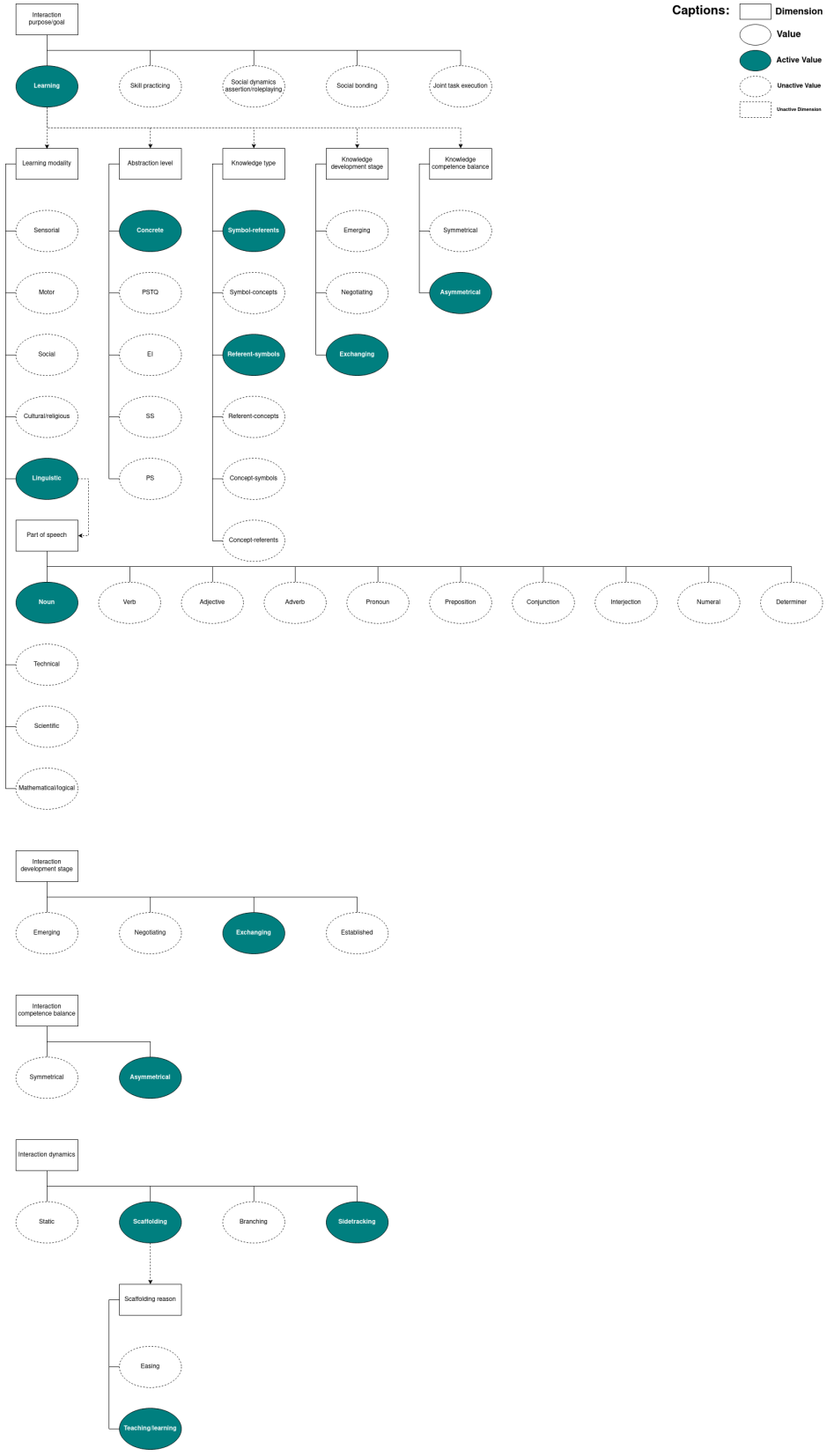
4.3.2 Classification by the proposed taxonomy

The interaction purpose/goal is learning. The interaction development stage is exchanging, as the learner needs to learn the actions of the interaction. The interaction competence balance is asymmetrical, as the learner does not know a priori most things about the actions of the interaction. The interaction dynamics are scaffolding and sidetracking, as the learner initially not knowing some aspects of the PF forces the interaction to be simpler, getting more complex as the learner learns them, and as the interaction is divided into two separate but related parts, one where new words for objects are learned and another where new objects for already known words are learned.

The learning modality is linguistic, as the learner learns the association between words and objects. The abstraction level is concrete, as the words refer directly to sensory experience (the objects perceived in the environment). The knowledge types are referent-symbols in the first part of the interaction and symbol-referents in the second part, as in the first part new words (the symbols) for objects (the referents) are learned and in the second part new objects (the referents) for already known words (the symbols) are learned. The knowledge development stage is exchanging, as the teacher has it well defined and only transmits it to the learner through initial demonstration and refinement. The Knowledge competence balance is asymmetrical, as the learner does not know a priori anything about the words and their association to the objects.

The part of speech is noun, as all words used to refer to objects are simple nouns. The scaffolding reason is teaching/learning, as the learner needs to learn the actions of the interaction.

Figure 5 – Classification of the proposed PF



Source: Own authorship (2022).

4.4 Learning Pragmatic Frames with FAMA

Now that a way to model a PF as a planning frame and an approach to action model learning were presented, a solution to the problem of learning a PF's state transitions can be proposed. By modeling the PF as a planning frame, each interaction round can be turned into a plan trace, as observed by the learning agent. Since such agent does not know the action models a priori, the plan trace can have PO or even NO action sequences/state trajectories, which FAMA is ideal for. From the plan traces, FAMA can be leveraged to learn the action models, and thus the PF's state transitions.

Algorithm 1 and Algorithm 2 show the algorithms for the main loop of the robot and human agents, respectively. These loops provide the base for a synchronous interaction between two agents, with the synchronization represented by the $notify(agent)$ and $wait(agent)$ functions. This synchronization is also used to inform the current action of the robot agent to the human agent, so that it can adapt itself accordingly, compute the robot's score or kill the robot when an illegal action transition is attempted.

Algorithm 1 – Robot main loop, where the action synchronization is simplified and learning is omitted.

Input: robot agent g_r, human agent g_h, starting action A_s, initial action A_i, interaction i, model $\mathcal{M}$

$a'_r \leftarrow A_s$ {Previous action of robot}
$a_r \leftarrow A_i$ {Current action of robot}
loop
    $notify(g_h)$ {Wait for human reaction}
    $wait(g_h)$
    {Check for a message in the message queue}
    if $poll(g_h)$ **then**
        $m_h \leftarrow recv(g_h)$
        $a_p \leftarrow possible\_actions(m_h, i)$ {Guess current action based on received message}
        $a_r \leftarrow guess\_next\_action(\mathcal{M}, a'_r, i, a_p)$
        $notify(g_h)$ {Wait for human reaction}
        $wait(g_h)$
    else if $agent(a_r) = g_r$ **then**
        $run(a_r)$
    end if
    $a'_r \leftarrow a_r$
    $a_r \leftarrow guess\_next\_action(\mathcal{M}, a_r, i, \varnothing)$
end loop

Source: Own authorship (2022).

$possible\_actions$ returns the subset of the human agent's actions that match the corresponding message's syntactic structure. Its algorithm is shown in Algorithm 20.

Two different algorithms are used to implement the $guess\_next\_action$ function, Algorithm 21 and Algorithm 22. The first one is a greedy one, that determines, based solely on the model learned so far and the current action, the best action to execute next. The second one is a Limited-Depth Depth-First-Search (LD-DFS) one, that besides leveraging the learned model and

Algorithm 2 – Human main loop, where the action synchronization is simplified and the check for the interaction's action transition limit is omitted.

Input: robot agent g_r, human agent g_h, starting action A_s, robot action a_r, interaction i

$a'_h \leftarrow A_s$ {Previous action of human}
$a_h \leftarrow A_s$ {Current action of human}
loop
    $wait(g_r)$
    $a'_h \leftarrow a_h$
    if $agent(next\_action(a_h, i)) = g_h$ **then**
        $a_h \leftarrow next\_action(a_h, i)$
        $run(a_h)$
        $notify(g_r)$ {Wait for robot reaction}
        $wait(g_r)$
        if $can\_be\_next(a'_h, i, a_r)$ **then**
            $a_h \leftarrow a_r$
        else
            $kill\_robot()$
        end if
    else
        if $can\_be\_next(a_h, i, a_r)$ **then**
            $a_h \leftarrow a_r$
            if $agent(a_h) = g_h$ **then**
                $run(a_h)$
                $notify(g_r)$ {Wait for robot reaction}
                $wait(g_r)$
                if $can\_be\_next(a'_h, i, a_r)$ **then**
                    $a_h \leftarrow a_r$
                else
                    $kill\_robot()$
                end if
            end if
        else
            $kill\_robot()$
        end if
    end if
    $notify(g_r)$ {Wait for robot reaction}
end loop

Source: Own authorship (2022).

the current action, also explores the possible future trajectories and chooses the next action that leads to the best overall trajectory. Both algorithms are non-deterministic, allowing for exploration of the search space.

$next\_action$ returns the next typical action, as shown in Frame 1 and Frame 2 (when not explicitly said, the next action is the one in the next row).

After each interaction step, the robot uses the information it has available (exchanged messages and local variables) to encode the performed action and the resulting state using the Planning Domain Definition Language (PDDL), adding them to the plan trace being built. When the interaction round ends, the robot adds its plan trace to $\mathcal{T}$ and updates $\mathcal{M}$ using FAMA.

4.5 Moving forward

Given the proposed solution to the problem of learning PFs, a way to classify them from the PF learning perspective, how to model them as a Computing problem, along with a PF to be used in the experiments, it is important to understand how those solutions and representations were developed.

5 MATERIALS AND METHODS

This chapter describes how the development of this work took place. It covers the motivations for choosing the example PF presented in section 4.3, the process of creating a computational and a taxonomic representation for PFs, how and why the part to be learned in a PF was chosen for this work, the literature review process employed to propose a solution, the implementation of the solution and the design choices for the experiments.

5.1 Designing an example Pragmatic Frame

Language is one of the most important elements of a people's culture, as it reflects and shapes their way of seeing the world. It also determines their structures of social interaction, as it greatly contributes to the construction of a sense of identity. When an individual comes into contact with a culture different from their own, language is the first barrier of interaction, but also the most important, since once overcome, it will immensely catalyze cultural exchange. Seeking to understand the role of language in this exchange process, the interaction scenario chosen for this work portraits the specific form of interaction that takes place during the learning of a language when a linguistic-cultural context is not available.

This interaction is fundamentally different from the one that occurs in traditional language learning, where learning resources and teachers are available, in addition to a broad cultural repertoire to learn from. While in the later the interlocutor interacts with people with greater knowledge of the language in question, who will make use of the most varied linguistic resources to facilitate his/her learning and understanding, in the former the people with whom the interlocutor will interact know as little about his/her language as he/she does about theirs.

The taxonomy proposed in this work can be used to describe several types of social interaction, but the ones where linguistic learning takes place are described in more detail.

5.2 Developing an initial model for Pragmatic Frames

Until a solution involving planning was developed, a model of PFs as state machines was developed to enable the initial reasoning about the proposed example PF in a Computing context. The state machine approach was chosen for its simplicity and expressive power, being able to represent the complexity that PFs can present in a manner easy to read. It also fits well with the stateful, turn-taking nature of most human interactions.

5.3 Finding and developing taxonomies for Pragmatic Frames

In the literature review of PFs, the works of Rohlfing *et al.* (2016), Vollmer *et al.* (2016) and others from the same research group were found as making up the foundations of the modern study of PFs, building on the initial work of Bruner (1983) but expanding it to Computing and Robotics. They are essential to the understanding of PFs, especially the taxonomy for PFs presented in Vollmer *et al.* (2016), but it became clear that to dive deeper into how a PF could be learned, another taxonomy had to be created with that in mind.

5.4 Choosing which part of a Pragmatic Frame to learn

Since the focus of this work is in how a PF can be learned, there are a few of its different parts that could be learned individually, each with its own peculiarity:

- The agents' roles;
- The syntax:
  - The messages exchanged;
  - The states;
  - The state transitions/actions;
  - The action transitions.
- The meaning:
  - The goal;
  - The execution of the actions;
  - The information slots.

Due to the complexity and time constraints of this work, only the state transitions were chosen to be learned, with all the other parts given a priori. The syntax of the state transitions are very often sparse and ambiguous, which already makes their learning hard enough.

5.5 Proposing a solution

To propose a solution to learning PFs, a literature review was conducted to find progresses already made on this subject. Since the study of PFs has only regained interest over the last decade, no work could be found specifically about learning PFs, and the review had to be expanded to other related topics, such as language acquisition, human-robot interaction, pragmatics, social learning and action learning. Reading the work of Aineto, Jiménez Celorrio e

Onaindia (2019) showed that an approach involving planning could be viable, as long as learning a PF could be modeled as a planning problem. Given the initial modeling of PFs as state machines, modeling them as planning frames proved to be considerably simpler.

5.6 Implementing the solution

The implementation is organized in two large packages (as shown in Figure 6), $agent\_core$ – responsible for implementing everything related to the agents, such as the agents' main interaction loops, the action executions and syntactic parsing of messages –, and $learning\_core$ – responsible for implementing everything related to the robot's learning of the interaction, such as the translation of the interaction to the equivalent actions and states in the PDDL model, building the plan trace from actions and states, learning the model from the observations and choosing the next action in the interaction from the learned model.

In the $agent\_core$ package, the $Agent$ module centralizes all the common code of the two agents, such as their communication, synchronization and the simulation parameters. The $RobotAgent$ module implements the robot agent, its main loop, action executions and the connection to the Learner module. The $HumanAgent$ module implements the human agent, its main loop, action executions, computing of the robot's score and logging of the simulation results.

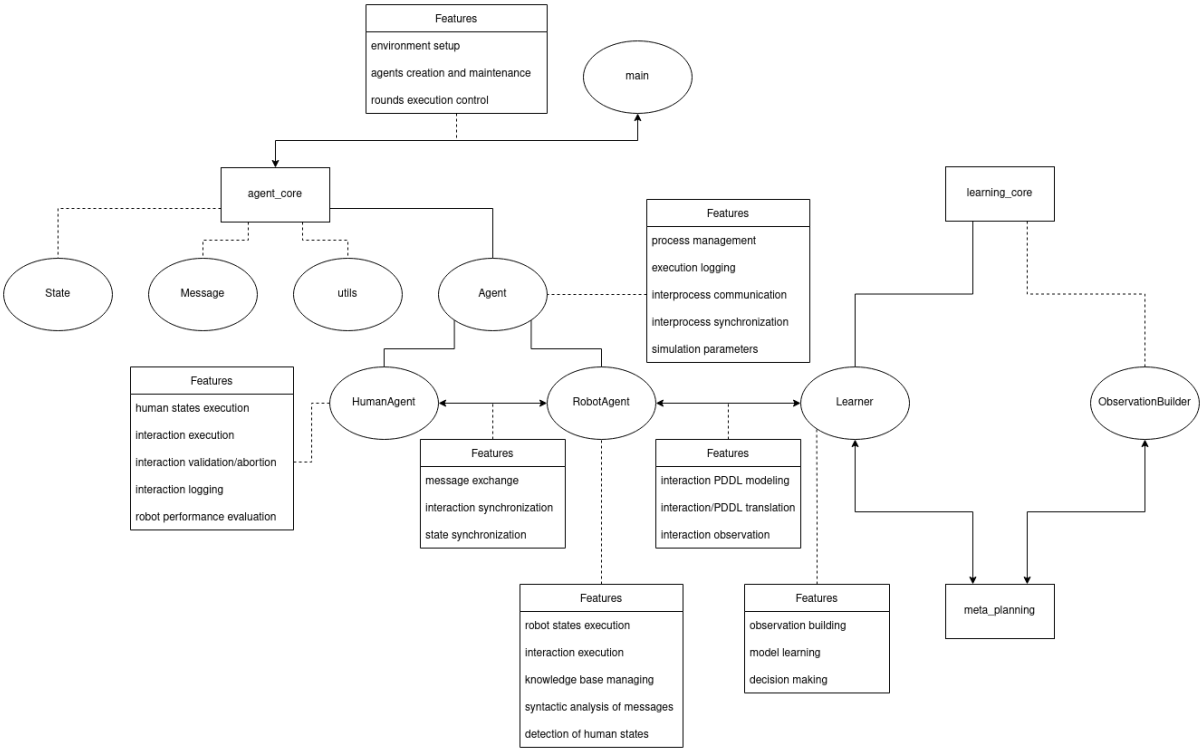
In the $learning\_core$ package, the $ObservationBuilder$ module implements the building of the plan trace from individual actions and states, interfacing with FAMA's package, $meta\_planning$. The $Learner$ module implements the creation of the PDDL actions and states from the interaction's actions and variables, respectively, builds the plan trace using $ObservationBuilder$, learns the PDDL model using $meta\_planning$'s $LearningTask$ and chooses the next action using the appropriate algorithm and parameters.

The implementation of the proposed solution, as well as the example PF and everything needed to perform the experiments are available at <https://github.com/natanjunges/TCC>.

5.7 Moving forward

Once the development process of this work is understood, the only things left to understand are how the experiments work, their hypotheses, objectives, theses, expected results, the results obtained and what they mean to this work.

Figure 6 – The functional map of the experimental artifact, where packages are represented as rectangles, modules as ellipses, interfaces between modules and other modules or packages as double-arrowed lines, and features are listed for each relevant module or interface.



Source: Own authorship (2022).

6 EXPERIMENTS AND RESULTS

To evaluate how well the proposed solution works and what effects different algorithms and parameter values have in its performance, some experiments are proposed using the example PF provided in section 4.3. Different metrics are used to evaluate the solution's performance, each accounting for a specific aspect considered relevant to this work.

6.1 Experimental setup

This work proposes two experiments, each with a part of the interaction. The general goals of the experiments are to analyze the completeness (space exploration) of the proposed solution, its ability to generalize and solve ambiguities. The experiments are divided into rounds. If the learner chooses an illegal action transition, there are too many action transitions (there might be an infinite loop) or if the interaction reaches its end (actions *CW2* or *CI2*, depending on which part of the interaction), the round ends. Then the teacher computes a score to rate the learner's performance in that round.

The metrics used to assess the performance of the learner agent are as follows:

- Action transition coverage (T_1, T_2): The number of unique legal action transitions covered by the agent divided by the total number of possible unique action transitions. This measures the exploration the learning agent will perform to learn the PF.
- Average performance score ($\bar{S}_1$, $\bar{S}_2$): The average of the performance score of all executed rounds, described by Equation 1 and Equation 7. This measures how successful the agent is at performing the interaction after learning the PF.
- Syntactic evaluation score ($F_{1,1}$, $F_{1,2}$): A syntactic comparison between the learned model and the reference model, described in Aineto, Jiménez Celorrio e Onaindia (2019).

The algorithms for actions RIRC1, RRC1 and RWC1 of the proposed PF have a parameter p that controls the probability of the received messages to mutate, simulating a communication noise. This parameter is fixed at $p = 10\%$ for all the experiments.

The two algorithms for $guess\_next\_action$ have two parameters that can be used to fine-tune their distributions:

- Exploration factor (x): Controls how frequently an action that can be applied (according to the learned model) is chosen instead of one that cannot. The proportion is $x : 1$.
- Goal-drivenness factor (g): Controls how frequently a goal action is chosen instead of a non-goal one. The proportion is $g : 1$.

In the experiments, both algorithms are compared performance-wise with different values for the parameters x and g. For the greedy one, five values are used for x ($x \in \{2, 4, 8, 16, 32\}$), and only one value is used for g ($g = 1$). For the LD-DFS one, four values are used for x ($x \in \{2, 5, 10, 20\}$) with g fixed ($g = 2$), and five values are used for g ($g \in \{2, 5, 10, 20, 50\}$) with x fixed ($x = 2$).

For each combination of part of experiment, action-choosing algorithm and value of parameters, 21 robots are simulated: 7 learn for 1000 rounds before being evaluated, 7 are evaluated without learning anything (the null model), and 7 are evaluated using the reference model. They are all evaluated for 1000 rounds.

6.1.1 Hypotheses

The interactions are turn-taking, with only one agent executing an action each turn, but they do not necessarily alternate each turn. In each interaction there are only two agents: a robot/learner and a human/teacher, each with a well defined role. The robot leads the interaction, choosing after each turn which should be the next action. The human plays along the robot's choices, but it can disapprove them if they are illegal, by either ignoring them and choosing the next action by itself or aborting the interaction entirely.

- The human agent is an oracle. It knows:
  - The name and parameters of the actions used in the interactions;
  - The initial actions;
  - The goal actions;
  - Which actions are its own and which are the robot's;
  - How to execute its own actions;
  - How to parse the syntactic type of the received messages;
  - How the exchanged information is extracted and which variables it is stored in;
  - Which robot action actually sent a received message;
  - Whether or not the robot agent executed a hidden action;
  - The variables needed to store the information exchanged in the interactions;
  - The type of the variables;
  - The value of all the variables;
  - Which variables and message types are relevant to each action;
  - How the variables and message types are logically connected to each action;
  - When to approve or disapprove the robot's decision of action transition; and

  - Which actions can or cannot be executed in a certain moment of the interactions.

- The robot agent knows *a priori*:
  - The name and parameters of the actions used in the interactions;
  - The initial actions;
  - The goal actions;
  - Which actions are its own and which are the human's;
  - How to execute its own actions;
  - How to parse the syntactic type of the received messages;
  - How the exchanged information is extracted and which variables it is stored in;
  - Which human actions can send a message of the same syntactic type as a received one;
  - The variables needed to store the information exchanged in the interactions;
  - The type of the variables;
  - The value of some of the variables;
  - Which variables and message types are relevant to each action; and
  - How to detect whether or not the human agent approves its decision of action transition.

- The robot agent does **not** know *a priori*:
  - How to execute the human's actions;
  - Which human action actually sent a received message;
  - Whether or not the human agent executed a hidden action;
  - The actual value of all the variables;
  - How the variables and message types are logically connected to each action; and
  - Which actions can or cannot be executed in a certain moment of the interactions.

6.1.2 Objectives

The robot agent is expected to learn:

- Whether or not the human agent executed a hidden action;
- How the variables and message types are logically connected to each action; and
- Which actions can or cannot be executed in a certain moment of the interactions.

6.1.3 Thesis and expected results

In the first experiment, the robot is expected to cover all legal action transitions ($T_1 = 1.0$) in the learning phase. The robot is expected to present a better average performance score than the null model ($\bar{S}_1 > \bar{S}_0$) in the evaluation phase. The robot's learned model is expected to have a good syntactic evaluation score ($F_{1,1} > 0.5$) when compared to the null and reference models ($F_{1,0} = 0.0$, $F_{1,ref} = 1.0$). In the second experiment, the robot is also expected to cover all legal action transitions ($T_2 = 1.0$), and perform better, both in average performance score and learned model syntactic evaluation score, than in the first experiment ($\bar{S}_2 > \bar{S}_1$, $F_{1,2} > F_{1,1}$).

6.1.4 First experiment

In this experiment, the learner performs only the first part of the interaction each round. For this part of the interaction, the limit of action transitions is 28 (twice as much as the First Simple Detour (FSD) interaction execution, to account for longer and more complex interactions). It starts with no knowledge, neither about the names of the objects around it, nor about the legal action transitions in the interaction. Over the rounds, the learner is expected to explore all action transitions, figuring out which ones are legal and which are illegal, choosing the legal ones over the illegal ones. The learner is expected to use the information acquisition to guide its learning of the correct ordering of the parts of the interaction.

The experiment aims at analyzing the completeness of the solution, by measuring the learner's exploration space coverage and average performance with different action-choosing algorithms. The learner's performance in a given round is measured by the score the teacher gives it at its end. This score is computed by verifying whether the interaction meets the following criteria:

- Equation 2: The final action (f, with each action represented as an integer, starting from 2 and increasing 1 each action, in the order shown in Frame 1) should be the last action of the interaction (in this case, *CW2*, $f = 10$);
- Equation 3: The last action transition (l) should be legal;
- Equation 4: The number of action transitions (n) should not exceed the limit imposed to the interaction (in this case, $n \leq 28$);

- Equation 5: The agent should perform all legal action transitions in the interaction (in this case, $t = 22$).

Each criterion not met implies a penalty over the score. The formula to compute the score is Equation 1:

$$S_1(f, l, n, t) = a_1(f)b_1(l)c_1(n)d_1(t) \tag{1}$$

$$a_1(f) = 0.02f + 0.8 \tag{2}$$

$$b_1(l) = \begin{cases} 1, & \text{if } l \in L_1 \\ 0.84, & \text{otherwise} \end{cases} \tag{3}$$

$$c_1(n) = \begin{cases} 1, & \text{if } n \leq 28 \\ 0.84, & \text{otherwise} \end{cases} \tag{4}$$

$$d_1(t) = 0.0072t + 0.84 \tag{5}$$

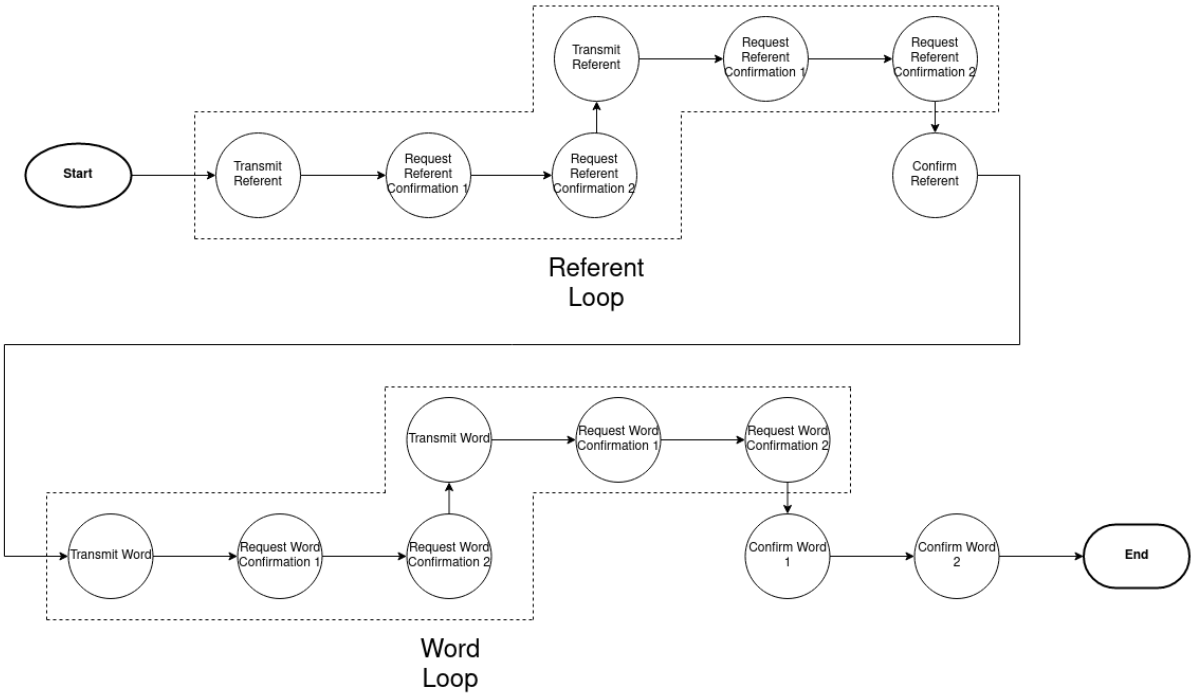
Where L is the set of legal action transitions. Notice the linear behavior of $a_1(f)$. It reflects the linear and progressive disposition of the actions. As the resultant score is computed as the product of the partial scores for each criteria, it could be zero if at least one of its partials were zero. To prevent that, the partials were chosen so that the minimum resultant score is approximately 0.5.

The FSD interaction execution is one where all actions in the first part of the interaction are executed at least once, following the standard sequence of actions but taking all the detours once:

$$\begin{aligned} FSD = \langle & TR, RRC1, RRC2, TR, RRC1, RRC2, CR, \\ & TW, RWC1, RWC2, TW, RWC1, RWC2, CW1, CW2 \rangle \end{aligned} \tag{6}$$

This execution is illustrated in Figure 7. It is relevant because it contains all the actions and main action transitions, without having any infinite loops. It is reasonable to consider it an upper limit to how long a typical interaction execution could be, with longer executions being up to twice as long.

Figure 7 – The FSD execution, where the referent and word loops are each repeated once.



Source: Own authorship (2022).

6.1.5 Second experiment

In this experiment, the learner can also perform the second part of the interaction, combining both parts as desired. For the second part of the interaction, the limit of action transitions is 20 (twice as much as the Second Simple Detour (SSD) execution). It already starts with knowledge from the first experiment, both about the names of the objects around it and about the legal action transitions in the first part of the interaction. Over the rounds, the learner is expected to perform better than in the first experiment, learning faster by reusing the knowledge already acquired in the first experiment.

The experiment aims at analyzing the ability of the solution to generalize and solve ambiguities, by measuring the learner's exploration space coverage and average performance (computed in a way similar to the first experiment) and comparing them to the first experiment. The learner's performance score is computed by verifying whether the interaction meets the following criteria:

- Equation 8: The final action (f, with each action represented as an integer, starting from 2 and increasing 1 each action, or 2 from CIR to RIC1, in the order shown in Frame 2; when actions from the first interaction are used, their values are the same of the ones used in that interaction) should be the last action of the interaction (in this case, *CW2* or *CI2*, $f = 10$);

- Equation 9: The last action transition (l) should be legal;
- Equation 10: The number of action transitions (n) should not exceed the limit imposed to the interaction (in this case, $n \leq 20$);
- Equation 11: The agent should perform all legal action transitions in the interaction (in this case, $t = 70$).

The formula to compute the score is Equation 7:

$$S_2(f, l, n, t) = a_2(f)b_2(l)c_2(n)d_2(t) \tag{7}$$

$$a_2(f) = 0.02f + 0.8 \tag{8}$$

$$b_2(l) = \begin{cases} 1, & \text{if } l \in L_2 \\ 0.84, & \text{otherwise} \end{cases} \tag{9}$$

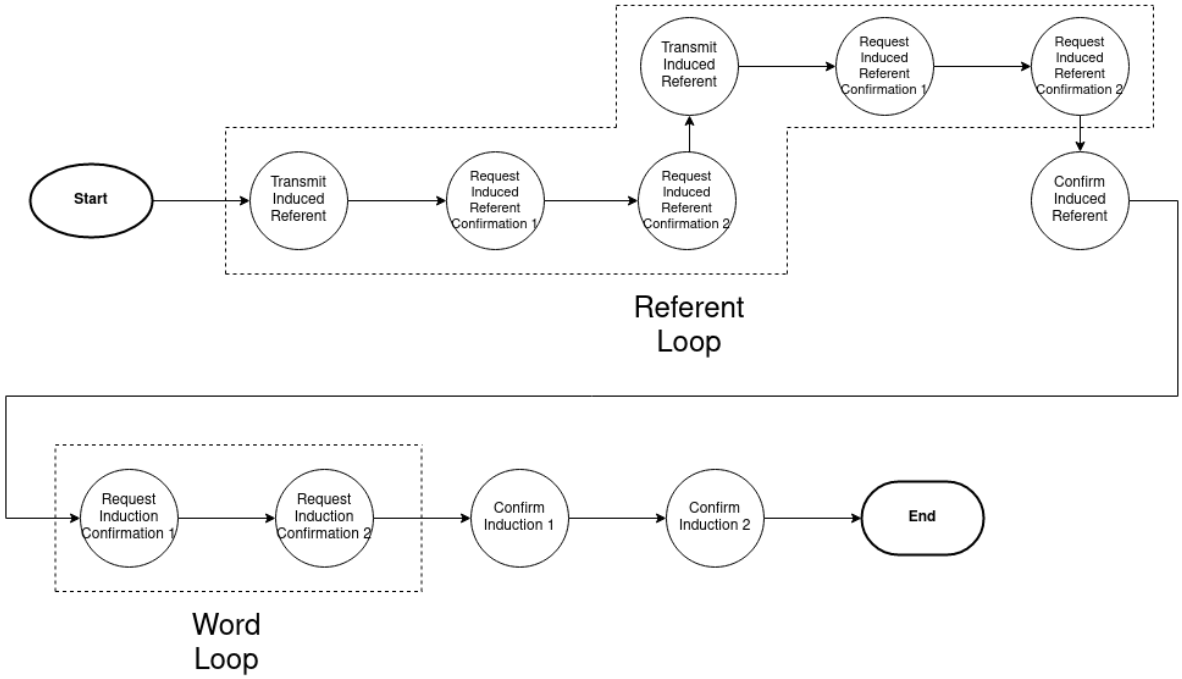
$$c_2(n) = \begin{cases} 1, & \text{if } n \leq 20 \\ 0.84, & \text{otherwise} \end{cases} \tag{10}$$

$$d_2(t) = 0.0022t + 0.84 \tag{11}$$

The SSD interaction execution is the equivalent of FSD for the second part of the execution. This execution is illustrated in Figure 8.

$$\begin{aligned} SSD = \langle & TIR, RIRC1, RIRC2, TIR, RIRC1, RIRC2, CIR, \\ & RIC1, RIC2, CI1, CI2 \rangle \end{aligned} \tag{12}$$

Figure 8 – The SSD execution, where the referent loop is repeated once.



Source: Own authorship (2022).

6.2 Analysis of results

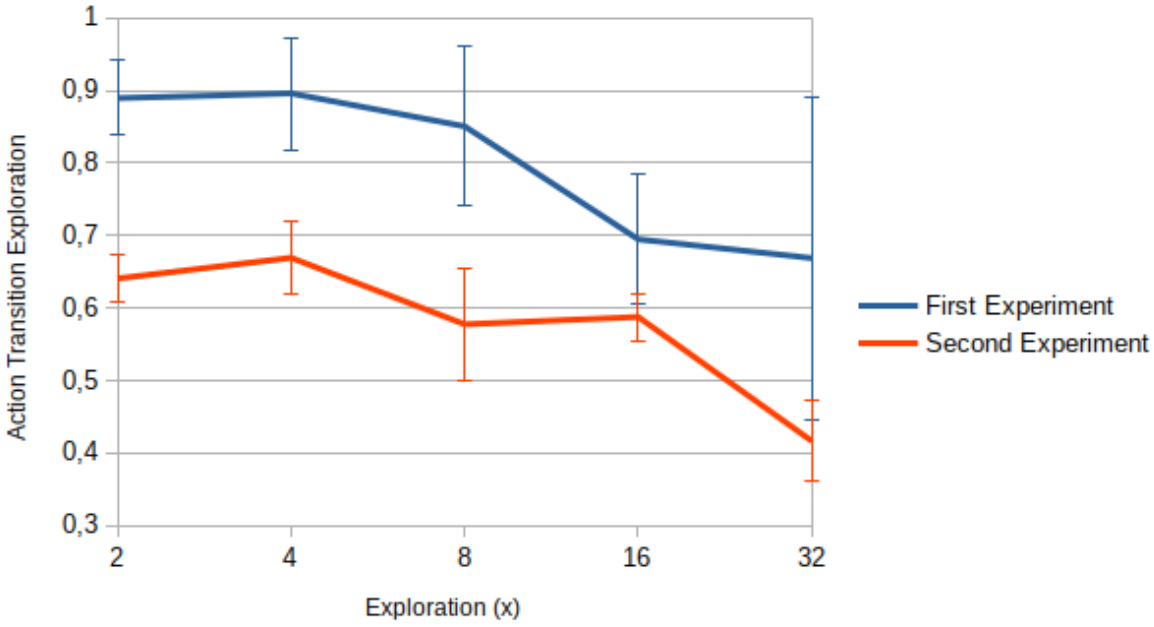
All raw results of the experiments are available at <https://github.com/natanjunges/TCC-data>, but a summary of them is also available in this document as a collection of graphs for easier visualization. In this section, 5 of the graphs considered more relevant are analysed, and the other ones are in Appendix D.

As can be seen in Graph 1, the action transition exploration seems to decrease as the exploration factor (x) increases, especially when the greedy algorithm is used. This result is counter-intuitive at first, but it might be an evidence of early abortion by the human agent. As the exploration factor increases, so does the chance of the robot choosing an illegal action transition, and thus being aborted by the human. If the abortion happens too early, the robot might not have the chance to explore valid action transitions.

When comparing the action transition exploration between the two experiments, it is significantly lower in the second experiment than in the first. The increased number of possible actions and action transitions in the second experiment could be making early abortions much more frequent.

Relative to the initial thesis, the robot was not capable of covering all the action transitions in either experiment. Even though, the results were good, with the average action transition exploration higher than 0.5 most of the time.

Graph 1 – Action transition exploration for greedy action-choosing.



Source: Own authorship (2022).

As can be seen in Graph 2, the performance score of the reference model seems to drastically decrease as the exploration factor (x) increases when using LD-DFS in the first experiment, while the performance scores of the null model and the learned model don't seem to be affected by it.

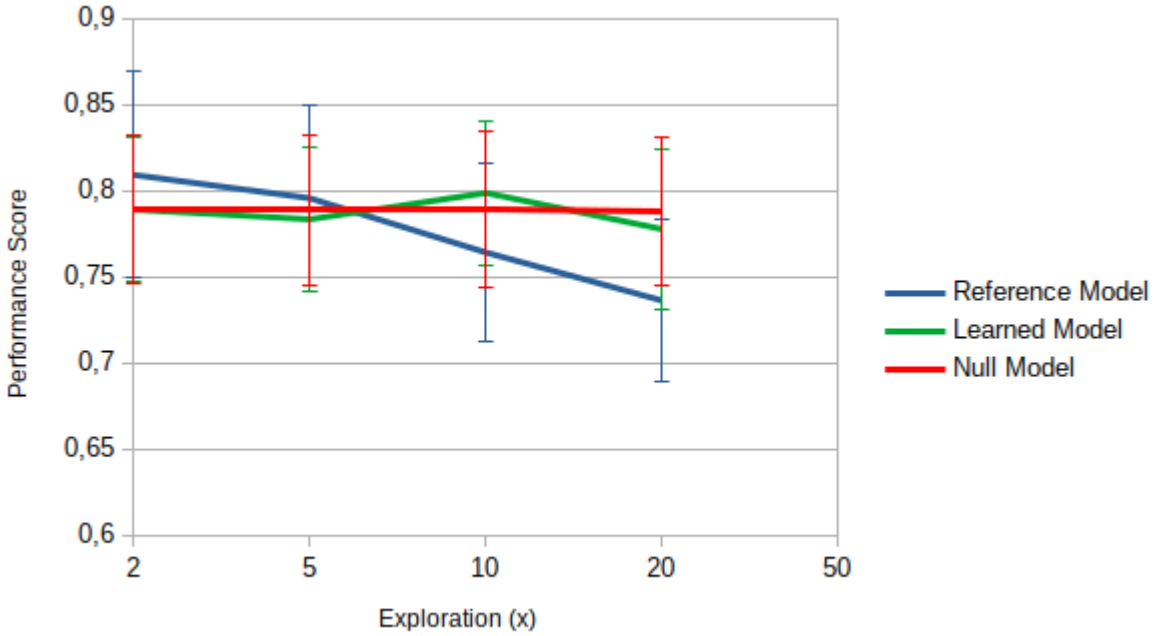
As can be seen in Graph 3, the performance score of the learned model seems to increase when using LD-DFS in the second experiment, especially as the goal-drivenness factor (g) increases.

As can be seen in Graph 4, the overall performance score of the learned model seems to increase slowly but steadily as the goal-drivenness factor (g) increases when using LD-DFS in both experiments.

When comparing the performance score of the learned model between the two experiments, it is in general higher in the second experiment than in the first. This might be due to the knowledge acquired in the first experiment being reused in the second.

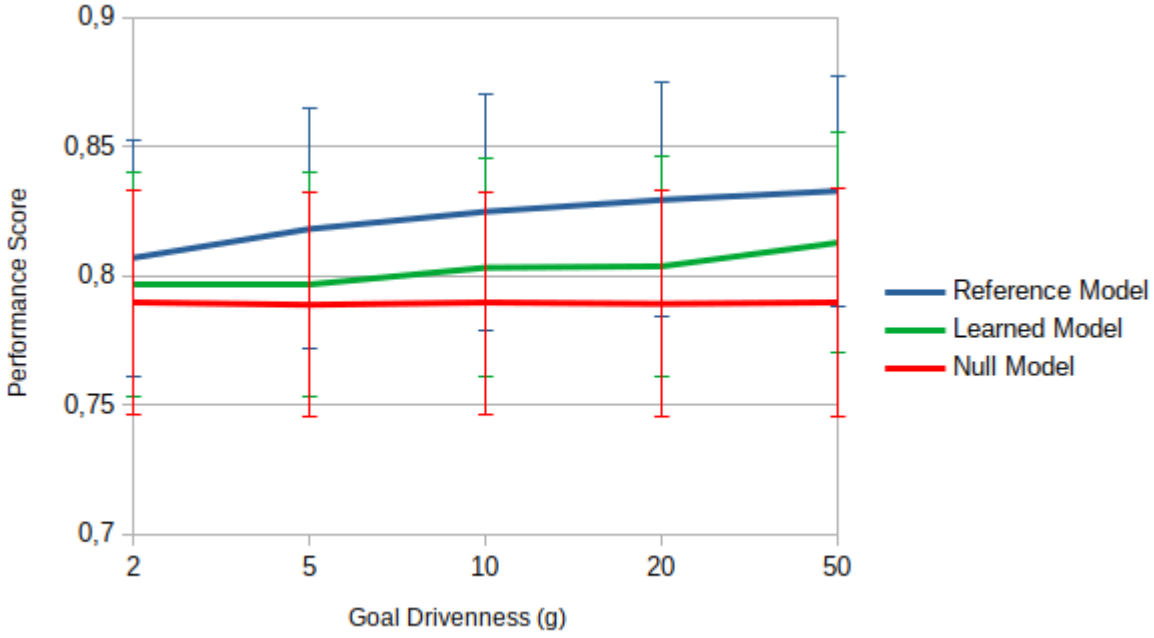
Relative to the initial thesis, the learned model presented a better, or at least similar, average performance than the null model in the first experiment, although the difference was not great. The performance in the second experiment was mostly higher than in the first.

Graph 2 – Performance score for LD-DFS action-choosing over x, first experiment.



Source: Own authorship (2022).

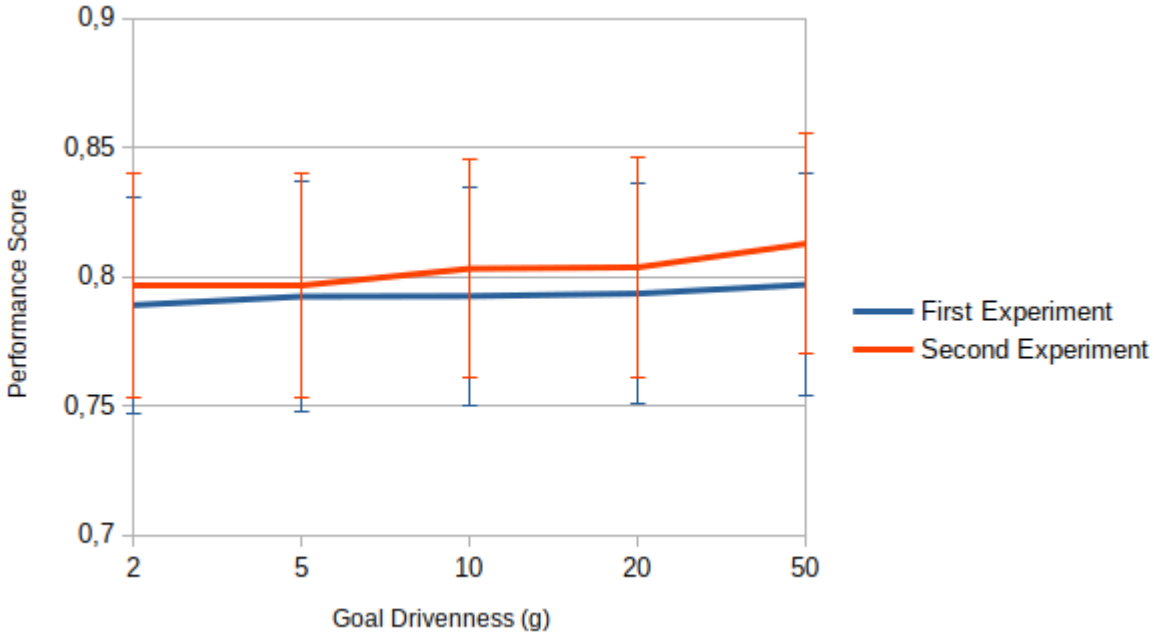
Graph 3 – Performance score for LD-DFS action-choosing over g, second experiment.



Source: Own authorship (2022).

As can be seen in Graph 5, the syntactic evaluation score seems to increase when LD-DFS is used, especially as the exploration factor (x) increases.

Graph 4 – Performance score of learned model with LD-DFS action-choosing over g.

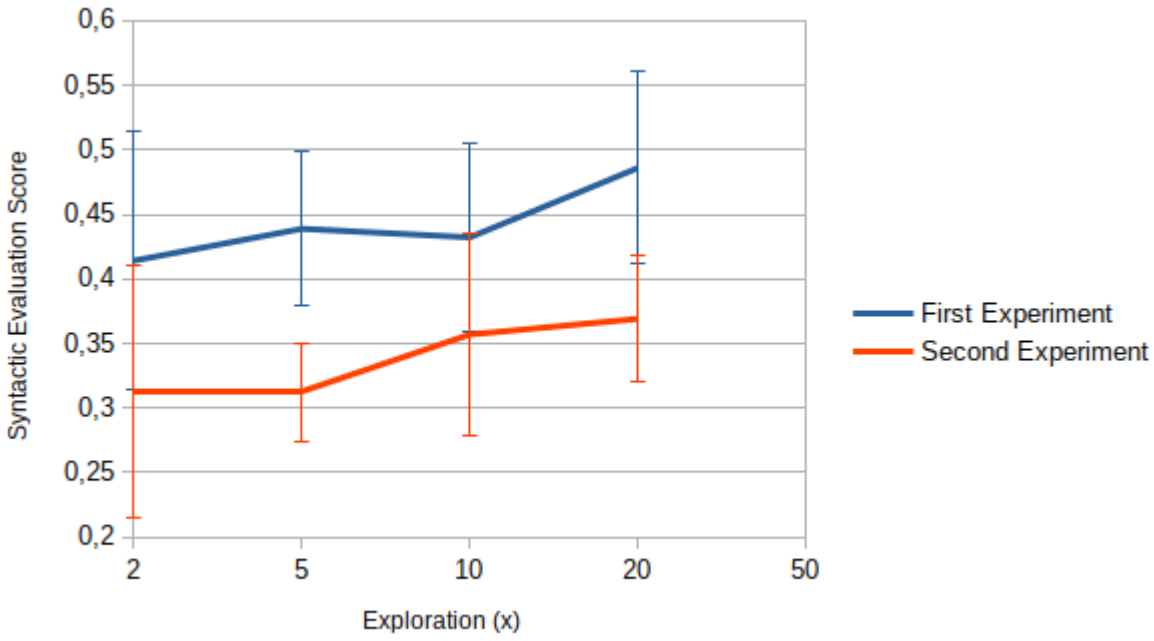


Source: Own authorship (2022).

When comparing the syntactic evaluation score between the two experiments, it is considerably lower in the second experiment. This can be an evidence of the actions learned in the first experiment being preferred over the ones introduced in the second experiment, preventing the novel actions from being learned. But this could also mean that the increased number of possible actions and action transitions in the second experiment make the learning more difficult.

Relative to the initial thesis, the robot's learned model did not have an average syntactic evaluation score higher than 0.5 in the first experiment and in the second experiment the syntactic evaluation score was even lower.

Graph 5 – Syntactic evaluation score for LD-DFS action-choosing over x.



Source: Own authorship (2022).

7 FINAL CONSIDERATIONS

The results obtained in the experiments show that, although there are many opportunities for improvement, this work successfully reached its main goal of providing computational representations of PFs. The goal of showing how a specific part of a PF can be learned was also reached, again with opportunities for improvement.

Combining the definition of PFs and the taxonomy provided by Vollmer *et al.* (2016) with the computational representation and the taxonomy provided by this work helped getting closer to understanding the central elements of a PF and how they could be learned. The taxonomies used in this work also helped getting closer to understanding the relevant dimensions for classifying PFs and how they are learned.

Evaluating the performance of the proposed solution was mostly done successfully, although the performance score doesn't seem to measure very accurately how successful was an interaction execution, and the syntactic evaluation score could be replaced by the semantic evaluation score, also proposed by Aineto, Jiménez Celorrio e Onaindia (2019), which evaluates better how well a model was learned when there are syntactic ambiguities.

Since the action-choosing algorithm determines how the agent explores the space of valid action transitions, it is essential to guide the learning of the PF in a scenario where the agent doesn't get any explicit feedback. In that regard, the results obtained in the experiments show that both action-choosing algorithms are not very good, since most interactions are aborted too early, preventing any significant learning.

Since the action-choosing algorithm also determines how the agent follows the learned model, the results obtained in the experiments also show that both action-choosing algorithms are not very good, since even the reference model didn't perform well in the interactions.

7.1 Future work

In future works, other psychological theories, such as WAT, could be combined with PFs, expanding the possibilities of abstraction, generalization, transfer learning and overall complexity of the PFs analysed.

The understanding of the central elements of a PF and how they could be learned could be furthered. Different parts of a PF could also be considered for learning, exploring the power and limitations of the proposed solution to learn them and exploring new solutions.

An expansion to the proposed taxonomy could also be beneficial to future works, being able to classify a wider number of PFs in a larger number of dimensions. A taxonomy comprehensive enough could also be employed to build novel PFs on demand, and potentially learn a PF with zero knowledge a priori about it.

Better options for action-choosing algorithms, especially the ones already used for planning could be explored and developed. Intrinsic motivations could be integrated into the action-

choosing algorithms, allowing for an automatic, life-long learning curriculum building and thus learning in an optimal pace.

A better performance score, and the semantic evaluation score for evaluating the learned model could be used, and other experimental metrics could be included for a more detailed analysis of the solution.

BIBLIOGRAPHY

AINETO, D.; JIMÉNEZ, S.; ONAINDIA, E. Learning STRIPS action models with classical planning. **CoRR**, abs/1903.01153, 2019. Disponível em: http://arxiv.org/abs/1903.01153.

AINETO, D.; Jiménez Celorrio, S.; ONAINDIA, E. Learning action models with minimal observability. **Artificial Intelligence**, v. 275, p. 104–137, 2019. ISSN 0004-3702. Disponível em: https://www.sciencedirect.com/science/article/pii/S0004370218304259.

AKGUN, B. *et al.* Keyframe-based learning from demonstration. **International Journal of Social Robotics**, v. 4, n. 4, p. 343–355, Nov 2012. ISSN 1875-4805. Disponível em: https://link.springer.com/article/10.1007/s12369-012-0160-0.

BORGHI, A. M. *et al.* Words as social tools: Language, sociality and inner grounding in abstract concepts. **Physics of Life Reviews**, v. 29, p. 120–153, 2019. ISSN 1571-0645. Disponível em: https://www.sciencedirect.com/science/article/pii/S1571064518301271.

BRUNER, J. **Child's talk: Learning to use language**. London: Oxford University Press, 1983. Disponível em: https://archive.org/details/childstalklearni0000brun.

CAKMAK, M.; THOMAZ, A. L. Designing robot learners that ask good questions. *In*: **Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction**. New York, NY, USA: Association for Computing Machinery, 2012. (HRI '12), p. 17–24. ISBN 9781450310635. Disponível em: https://dl.acm.org/doi/10.1145/2157689.2157693.

CALINON, S.; BILLARD, A. Teaching a humanoid robot to recognize and reproduce social cues. *In*: **ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication**. IEEE, 2006. p. 346–351. Disponível em: https://ieeexplore.ieee.org/document/4107832/.

CALINON, S. *et al.* Learning and reproduction of gestures by imitation. **IEEE Robotics & Automation Magazine**, v. 17, n. 2, p. 44–54, 2010. Disponível em: https://ieeexplore.ieee.org/document/5480475.

CASTELFRANCHI, C. Intentions in the light of goals. **Topoi**, v. 33, n. 1, p. 103–116, 2014. ISSN 1572-8749. Disponível em: https://link.springer.com/article/10.1007/s11245-013-9218-3.

CUAYÁHUITL, H. Robot learning from verbal interaction: A brief survey. *In*: **AISB Convention 2015**. Canterbury: Society for the Study of Artificial Intelligence and the Simulation of Behaviour, 2015. Disponível em: https://www.cs.kent.ac.uk/events/2015/AISB2015/proceedings/hri/14-Cuayahuitl-robotlearningfrom.pdf.

GIENGER, M.; MüHLIG, M.; STEIL, J. J. Imitating object movement skills with robots — a task-level approach exploiting generalization and invariance. *In*: **2010 IEEE/RSJ International Conference on Intelligent Robots and Systems**. IEEE, 2010. p. 1262–1269. Disponível em: https://ieeexplore.ieee.org/document/5649990/.

GRIZOU, J. *et al.* Interactive learning from unlabeled instructions. *In*: **Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI)**. AUAI Press, 2014. Disponível em: http://infoscience.epfl.ch/record/205138.

GRIZOU, J.; LOPES, M.; OUDEYER, P.-Y. Robot learning simultaneously a task and how to interpret human instructions. *In*: **2013 IEEE Third Joint International Conference on**

Development and Learning and Epigenetic Robotics (ICDL). IEEE, 2013. p. 1–8. Disponível em: https://ieeexplore.ieee.org/document/6652523.

GROLLMAN, D. H.; BILLARD, A. Donut as i do: Learning from failed demonstrations. *In*: **2011 IEEE International Conference on Robotics and Automation**. IEEE, 2011. p. 3804–3809. Disponível em: https://ieeexplore.ieee.org/document/5979757/.

HASLUM, P. **Admissible Heuristics for Automated Planning**. 2006. 164 p. Tese (Doutorado) — Linköping University, KPLAB - Knowledge Processing Lab, The Institute of Technology, 2006.

KAPLAN, F. *et al.* Robotic clicker training. **Robotics and Autonomous Systems**, v. 38, n. 3, p. 197–206, 2002. ISSN 0921-8890. Advances in Robot Skill Learning. Disponível em: https://www.sciencedirect.com/science/article/pii/S0921889002001689.

LALLEE, S. *et al.* Human-robot cooperation based on interaction learning. *In*: ____. **From Motor Learning to Interaction Learning in Robots**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 491–536. ISBN 978-3-642-05181-4. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-05181-4_21.

LOPES, M.; MELO, F. S.; MONTESANO, L. Affordance-based imitation learning in robots. *In*: **2007 IEEE/RSJ International Conference on Intelligent Robots and Systems**. IEEE, 2007. p. 1015–1021. Disponível em: https://ieeexplore.ieee.org/document/4399517/.

MÜHLIG, M.; GIENGER, M.; STEIL, J. J. Interactive imitation learning of object movement skills. **Autonomous Robots**, v. 32, n. 2, p. 97–114, Feb 2012. ISSN 1573-7527. Disponível em: https://link.springer.com/article/10.1007/s10514-011-9261-0.

NICOLESCU, M.; MATARIC, M. J. Task learning through imitation and human-robot interaction. **Models and mechanisms of imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions**, Citeseer, p. 407–424, 2005. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8ad46327b735ddc3576ad962509733d57eac601e.

OGDEN, C.; RICHARDS, I. **The Meaning of Meaning**. San Diego: Harcourt Brace Jovanovich, 1989.

ROHLFING, K. J. *et al.* An alternative to mapping a word onto a concept in language acquisition: Pragmatic frames. **Frontiers in Psychology**, v. 7, 2016. ISSN 1664-1078. Disponível em: https://www.frontiersin.org/article/10.3389/fpsyg.2016.00470.

RUSSELL, S.; NORVIG, P. **Artificial intelligence: A Modern Approach**. 4. ed. Hoboken: Pearson, 2020.

SAUNDERS, J.; NEHANIV, C. L.; DAUTENHAHN, K. Teaching robots by moulding behavior and scaffolding the environment. *In*: **Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction**. New York, NY, USA: Association for Computing Machinery, 2006. (HRI '06), p. 118–125. ISBN 1595932941. Disponível em: https://dl.acm.org/doi/10.1145/1121241.1121263.

STEELS, L.; KAPLAN, F. Aibo's first words: The social learning of language and meaning. **Evolution of Communication**, John Benjamins, v. 4, n. 1, p. 3–32, 2000. ISSN 1387-5337. Disponível em: https://www.jbe-platform.com/content/journals/10.1075/eoc.4.1.03ste.

THOMAZ, A. L.; BREAZEAL, C. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. *In*: **Proceedings of the 21st**

National Conference on Artificial Intelligence. Boston: AAAI Press, 2006. p. 1000–1005. ISBN 9781577352815. Disponível em: https://www.aaai.org/Papers/AAAI/2006/AAAI06-157.pdf.

THOMAZ, A. L.; CAKMAK, M. Learning about objects with human teachers. *In*: **Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction**. New York, NY, USA: Association for Computing Machinery, 2009. (HRI '09), p. 15–22. ISBN 9781605584041. Disponível em: https://dl.acm.org/doi/10.1145/1514095.1514101.

TURNITSA, C.; TOLK, A. Knowledge representation and the dimensions of a multi-model relationship. *In*: . [*s.n.*], 2008. p. 1148–1156. Disponível em: https://www.researchgate.net/publication/221525084_Knowledge_representation_and_the_dimensions_of_a_multi-model_relationship.

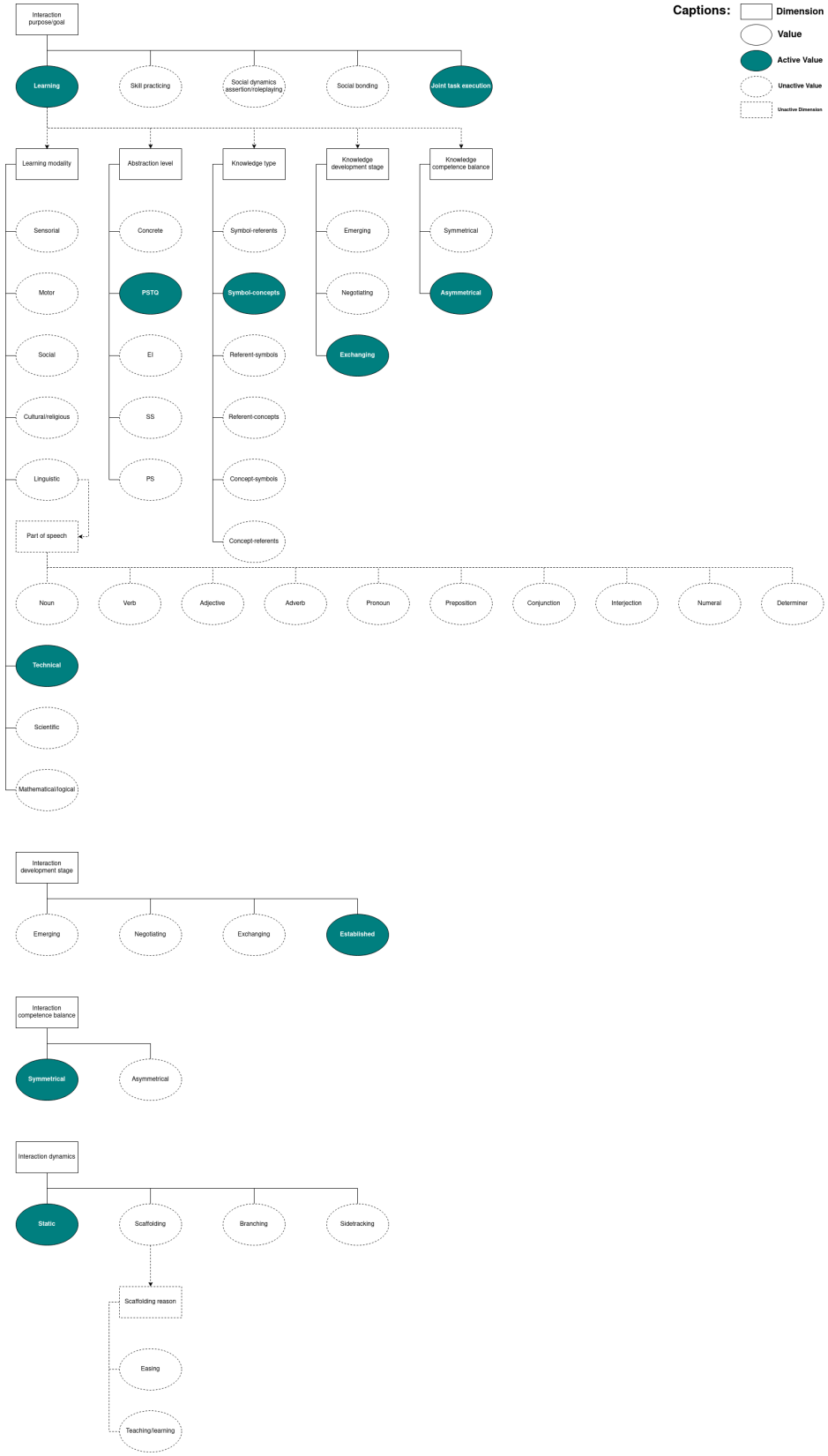
VILLANI, C. *et al.* Varieties of abstract concepts and their multiple dimensions. **Language and Cognition**, Cambridge University Press, v. 11, n. 3, p. 403–430, 2019. Disponível em: https://www.cambridge.org/core/journals/language-and-cognition/article/abs/varieties-of-abstract-concepts-and-their-multiple-dimensions/85D1BB9A35E0C3A041C73DFE35D3E0FF.

VOLLMER, A.-L. *et al.* Pragmatic frames for teaching and learning in human–robot interaction: Review and challenges. **Frontiers in Neurorobotics**, v. 10, 2016. ISSN 1662-5218. Disponível em: https://www.frontiersin.org/article/10.3389/fnbot.2016.00010.

YAMASHITA, Y.; TANI, J. Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. **PLOS Computational Biology**, Public Library of Science, v. 4, n. 11, p. 1–18, 11 2008. Disponível em: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000220.

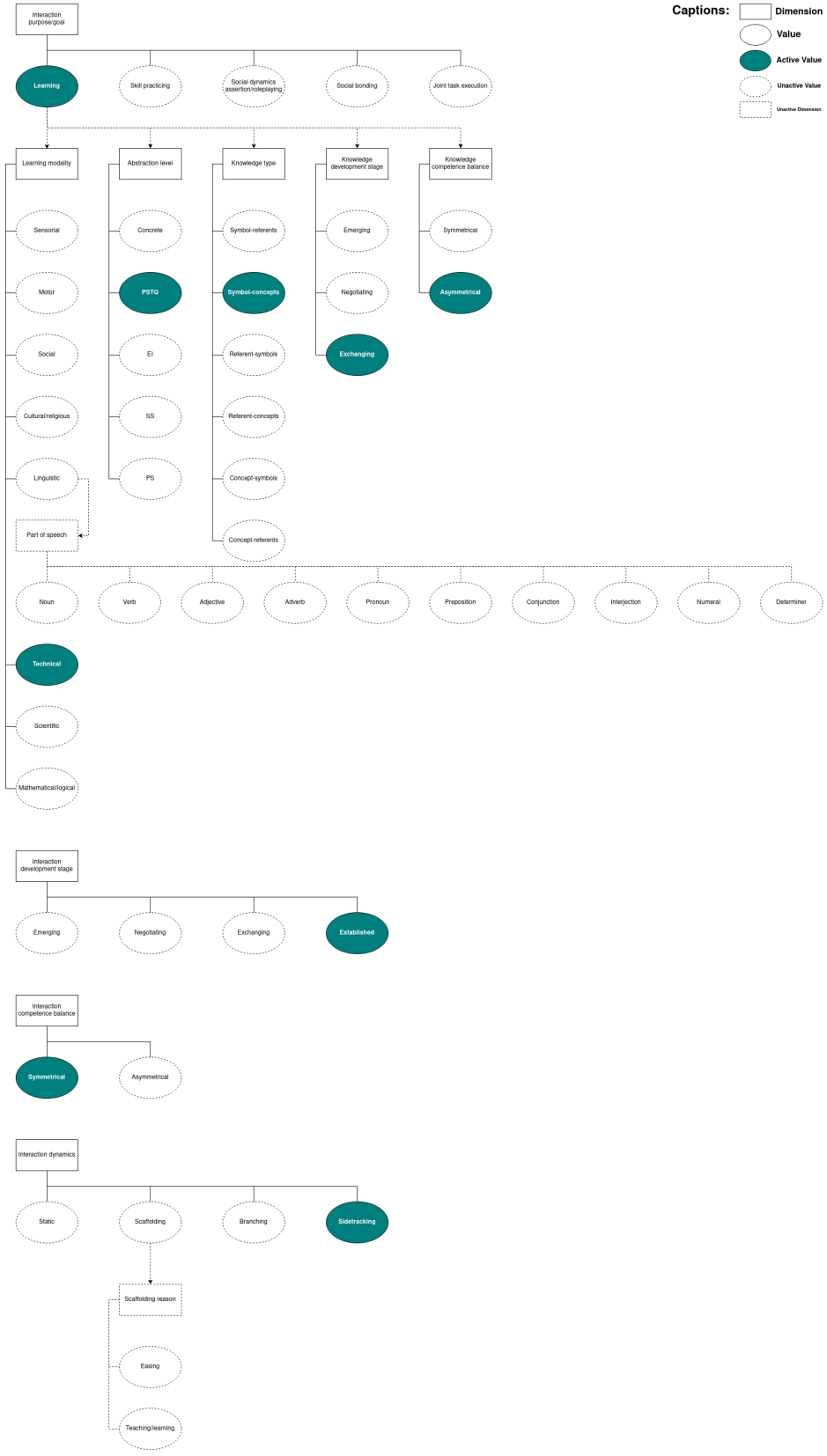
APPENDIX A – Classification of Pragmatic Frames from Vollmer et al. (2016)

Figure 9 – Classification of the PF in Lallee *et al.* (2010)



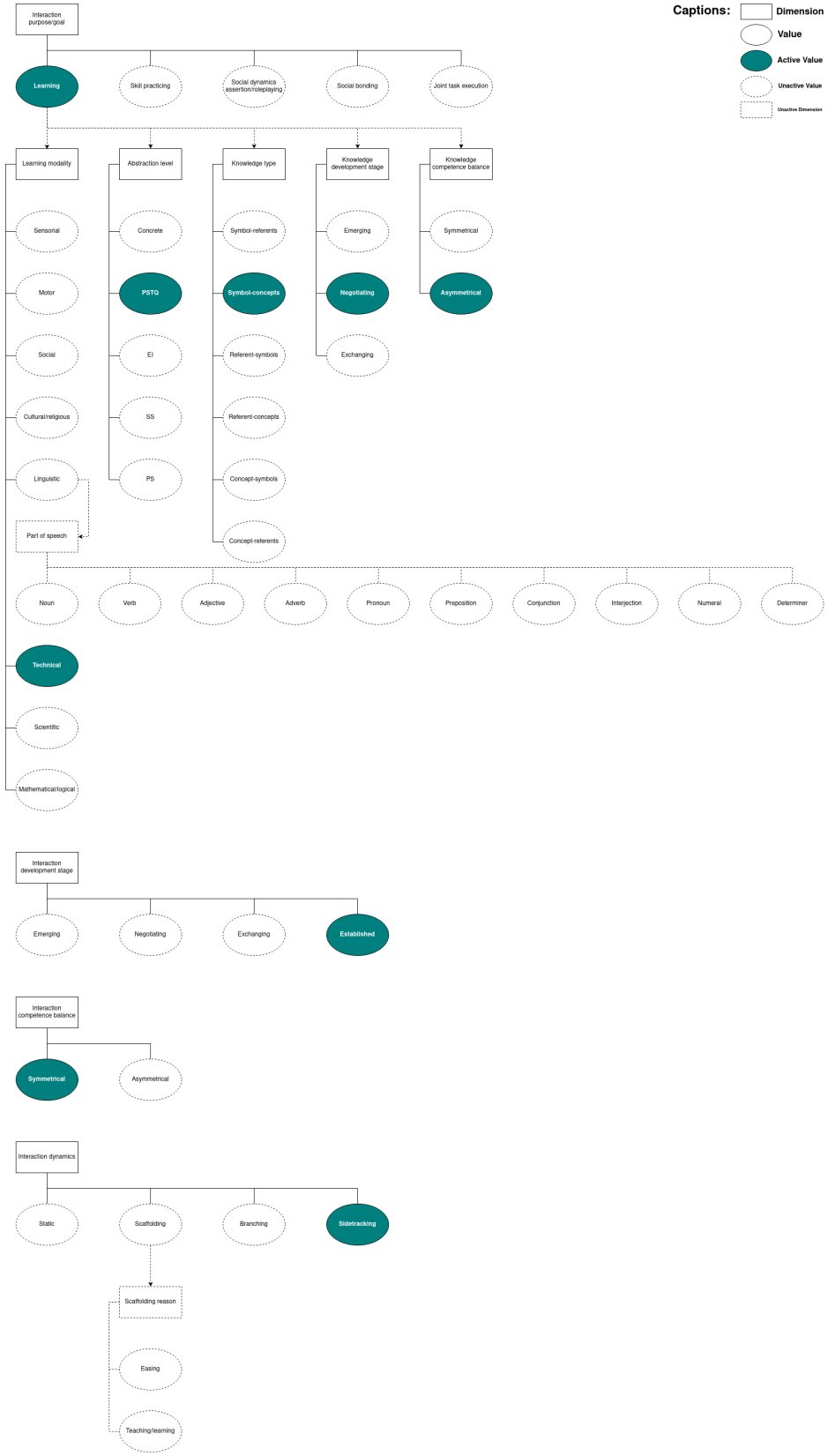
Source: Own authorship (2022).

Figure 10 – Classification of the PF in Saunders, Nehaniv e Dautenhahn (2006)



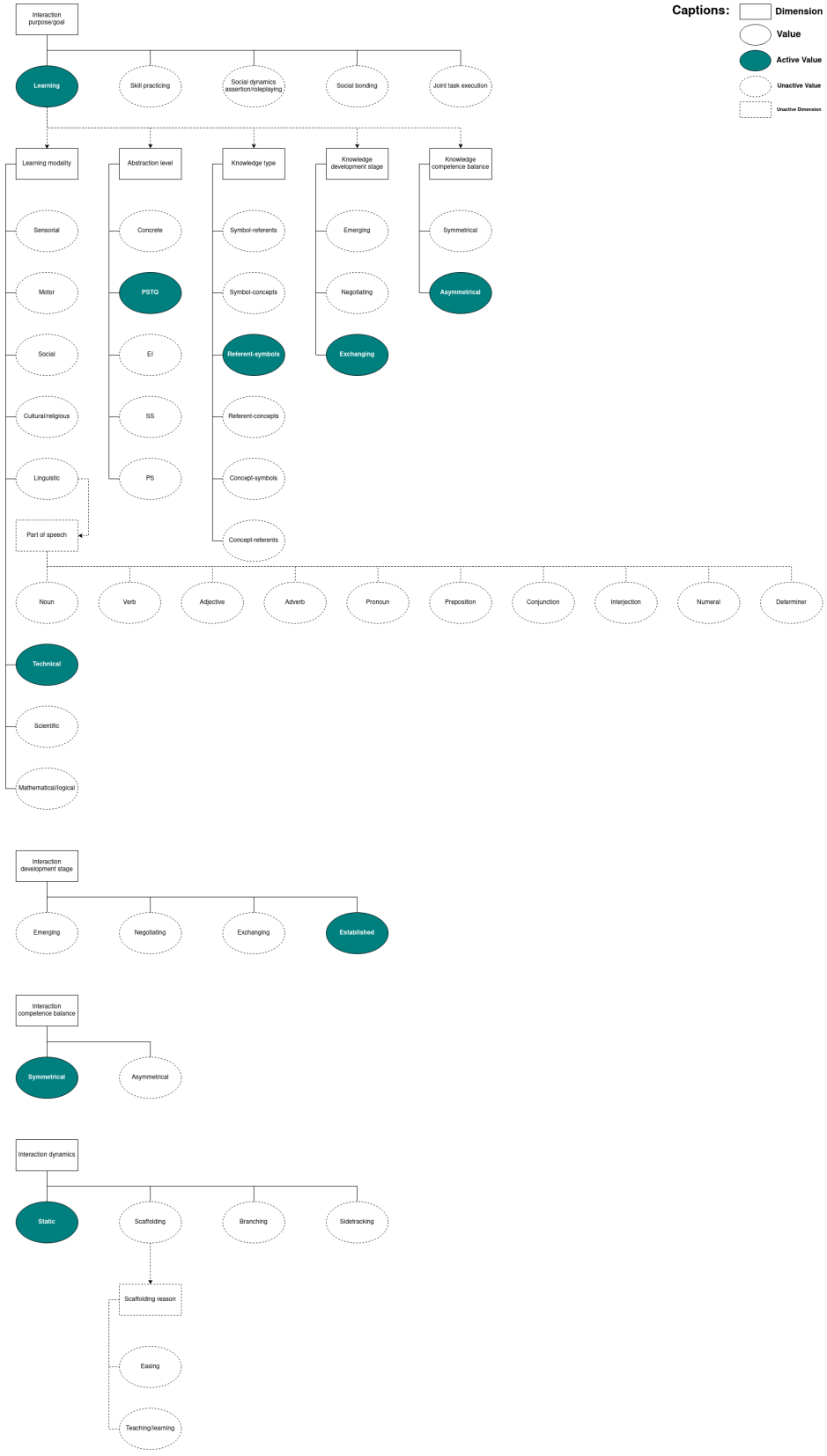
Source: Own authorship (2022).

Figure 11 – Classification of the PF in Nicolescu e Mataric (2005)



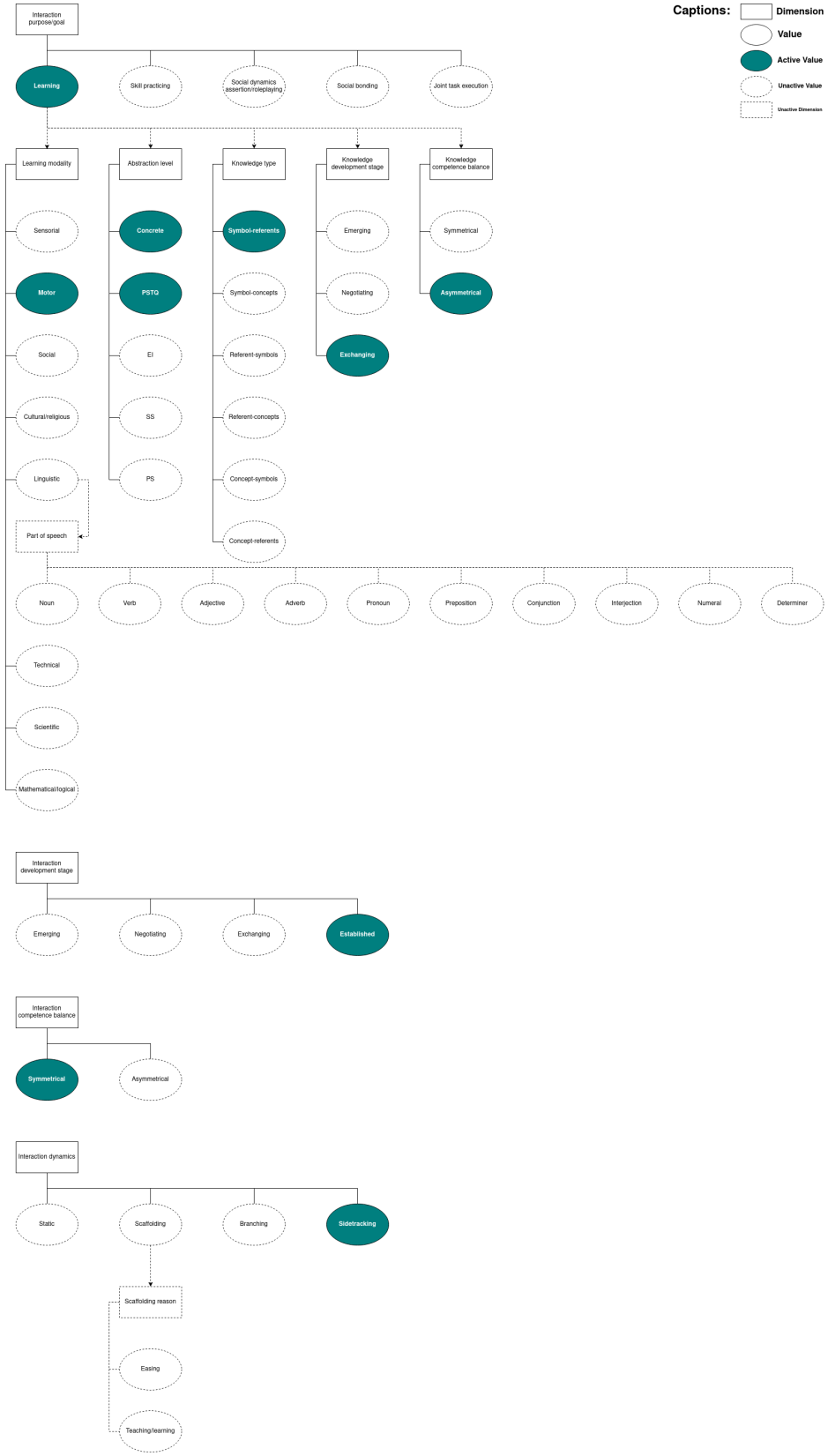
Source: Own authorship (2022).

Figure 12 – Classification of the PF in Thomaz e Cakmak (2009)



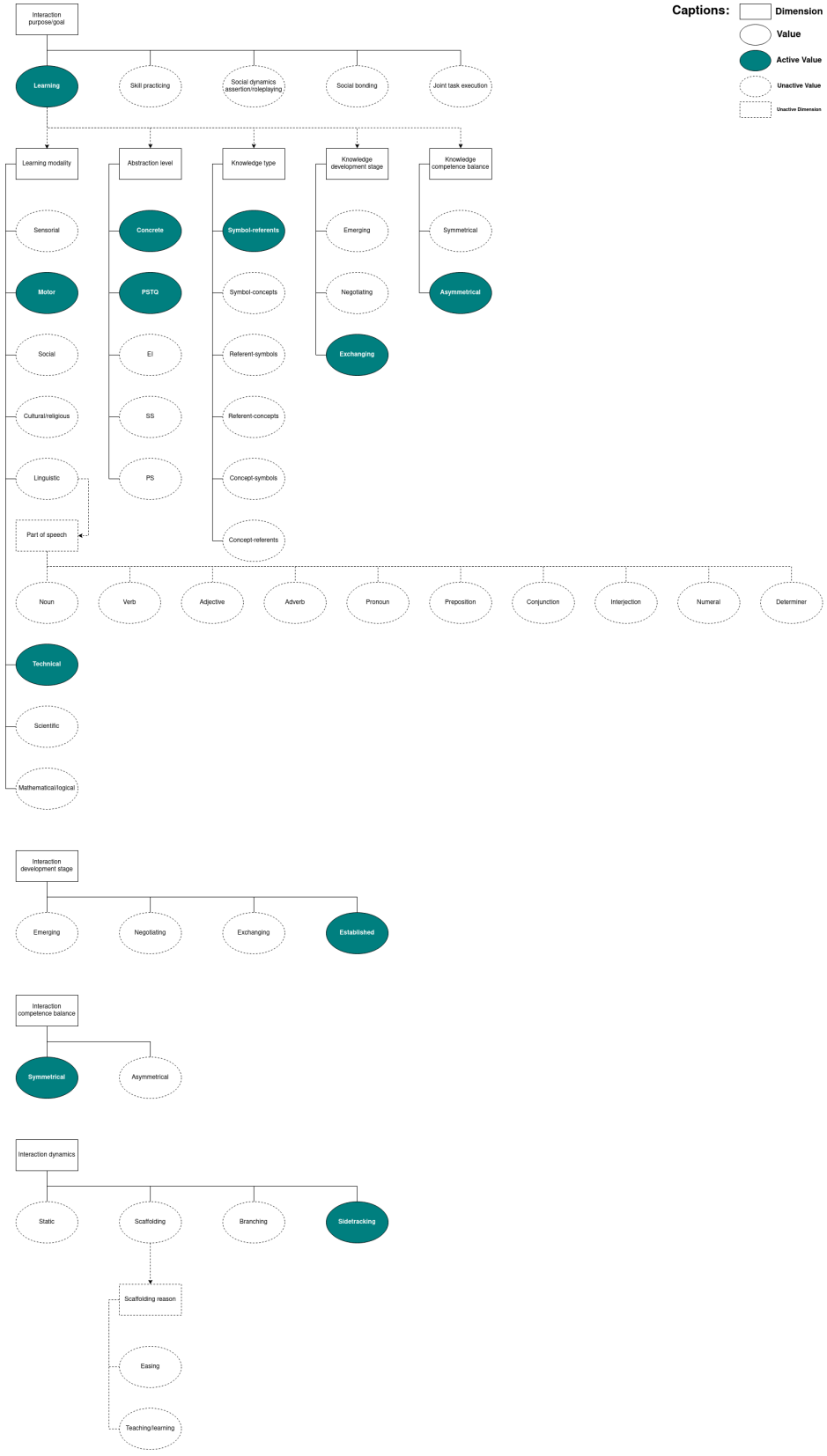
Source: Own authorship (2022).

Figure 13 – Classification of the PF in Yamashita e Tani (2008)



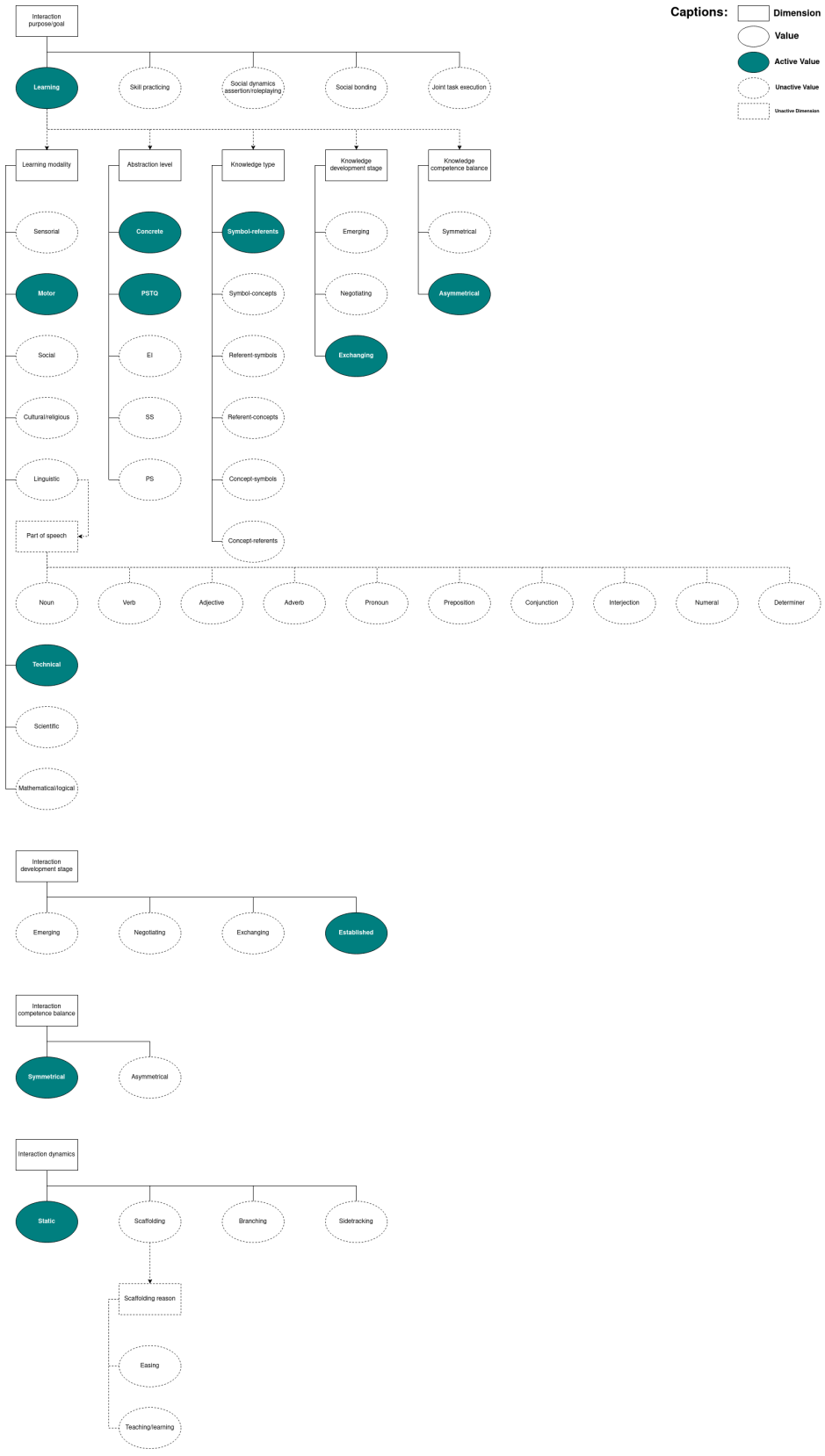
Source: Own authorship (2022).

Figure 14 – Classification of the PF in Calinon *et al.* (2010)



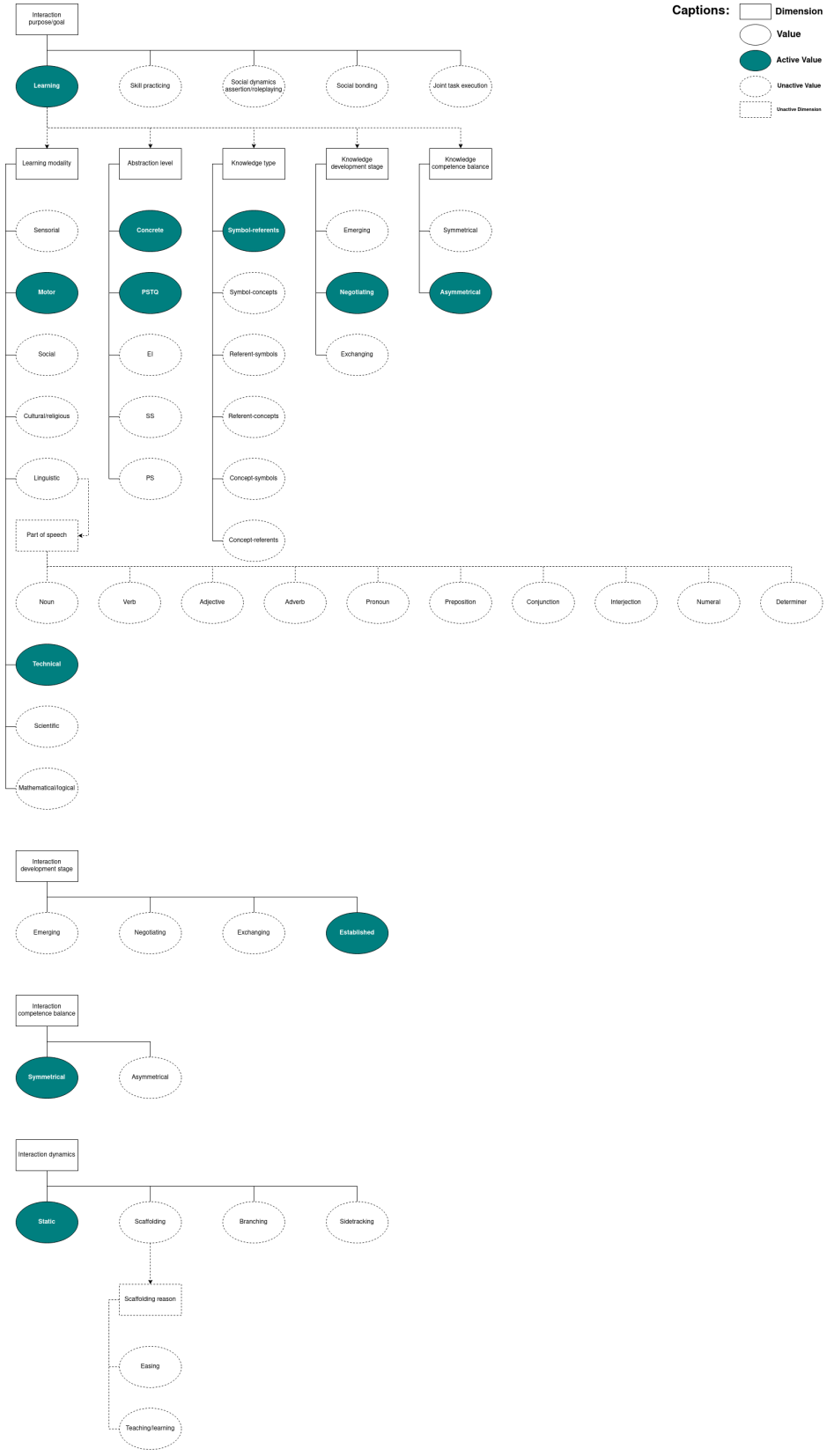
Source: Own authorship (2022).

Figure 15 – Classification of the PFs in Mühlig, Gienger e Steil (2012) and Gienger, Mühlig e Steil (2010)



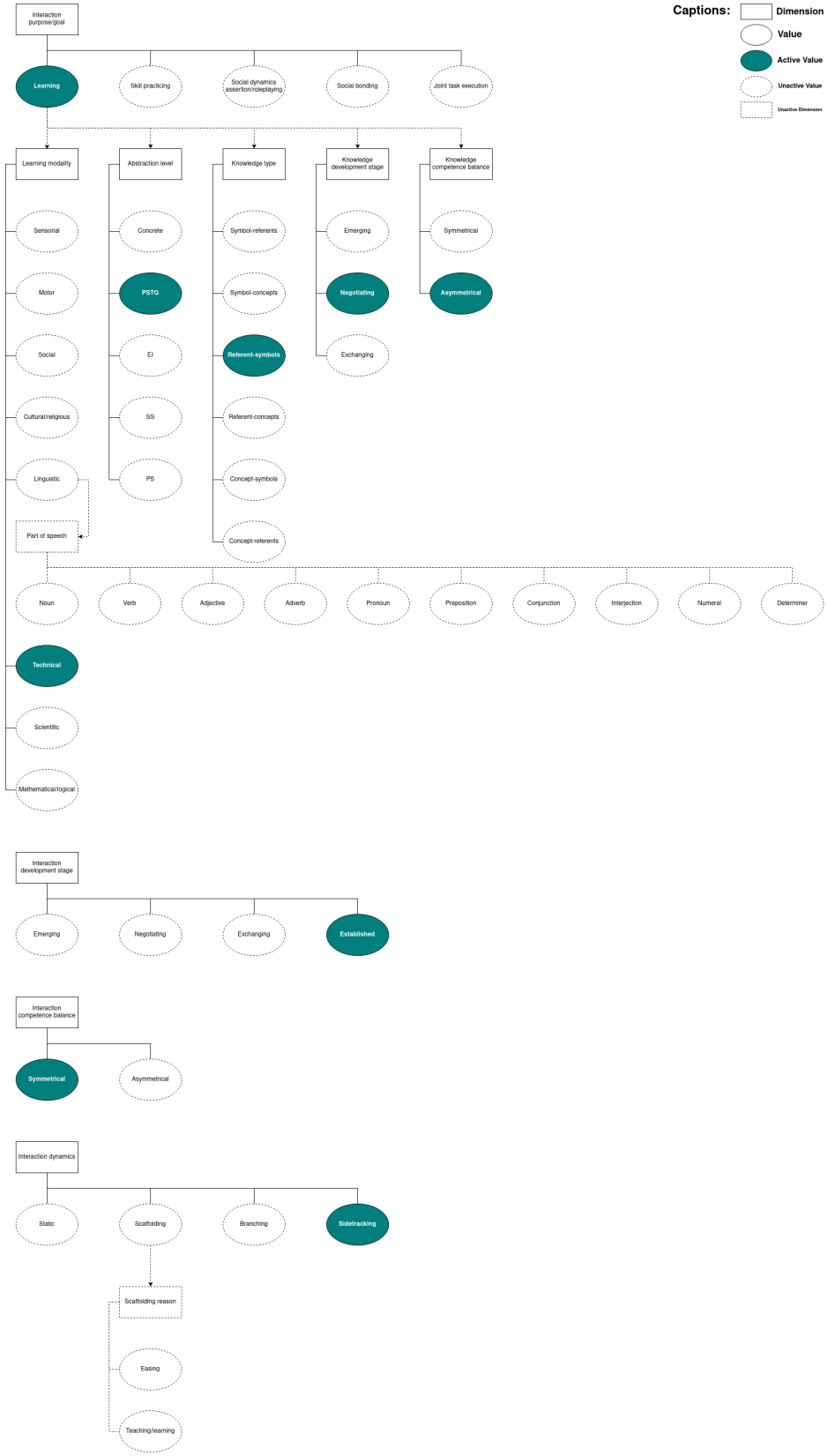
Source: Own authorship (2022).

Figure 16 – Classification of the PF in Grollman e Billard (2011)



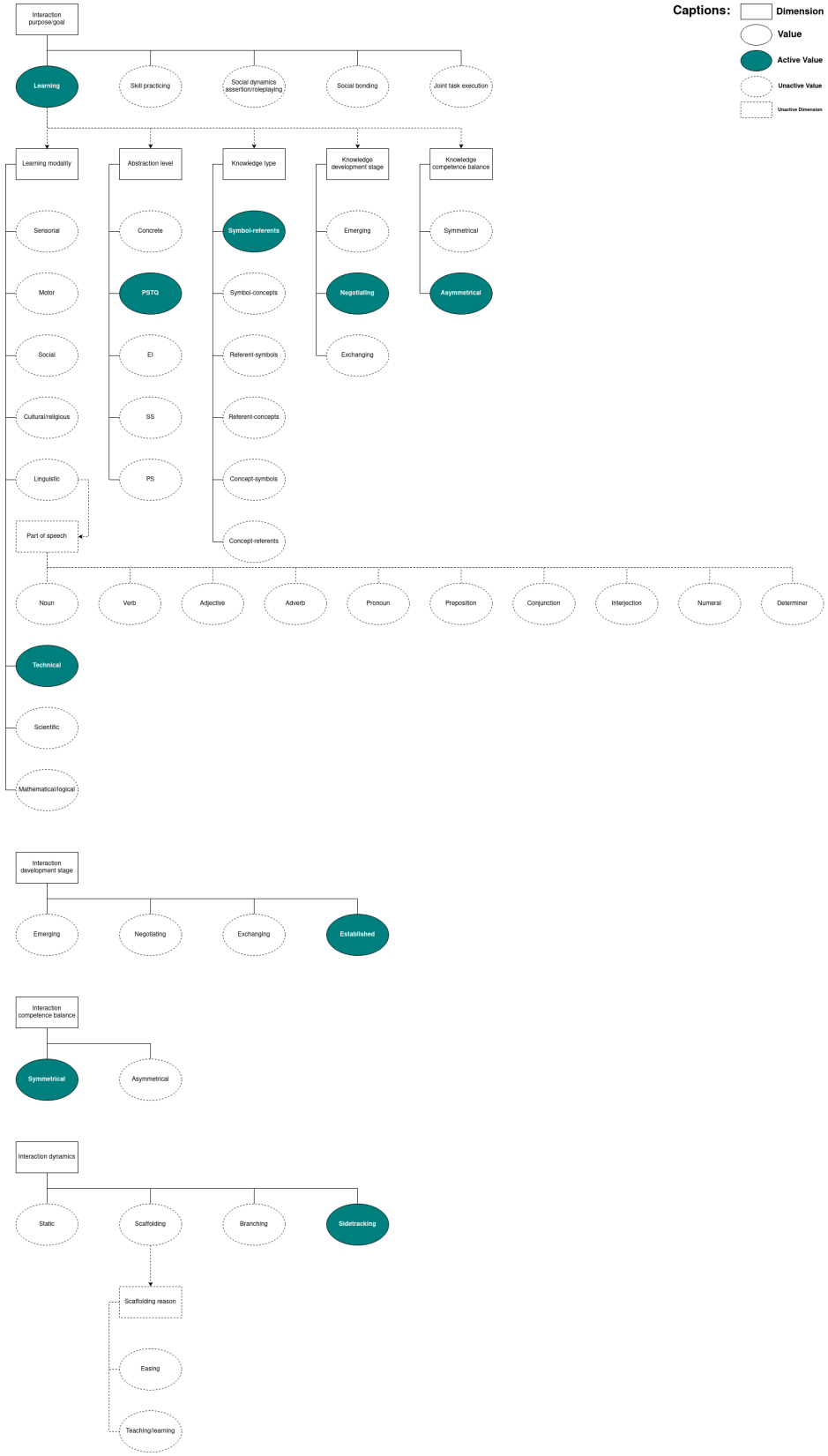
Source: Own authorship (2022).

Figure 17 – Classification of the PF in Lopes, Melo e Montesano (2007)



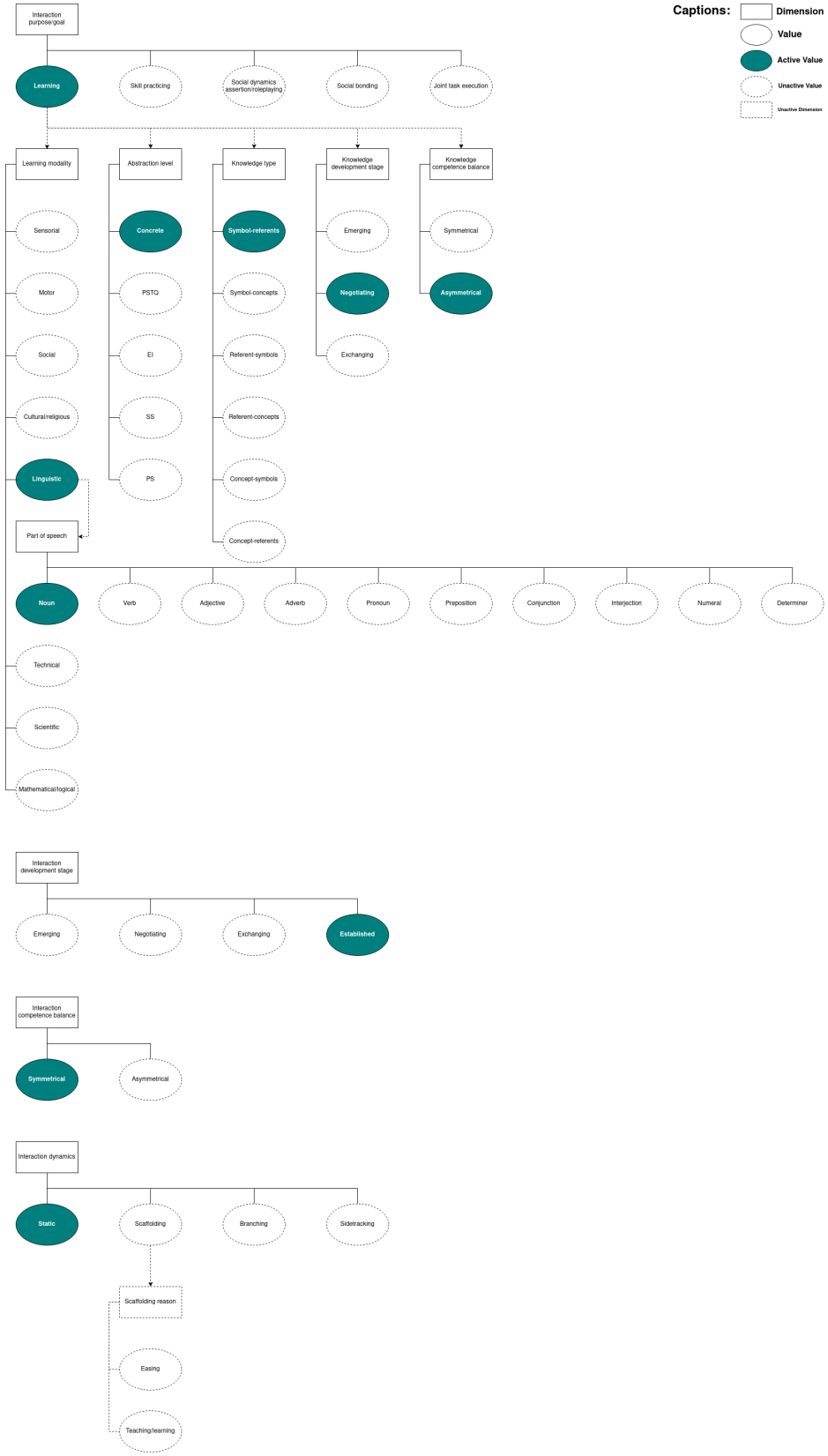
Source: Own authorship (2022).

Figure 18 – Classification of the PF in Kaplan *et al.* (2002)



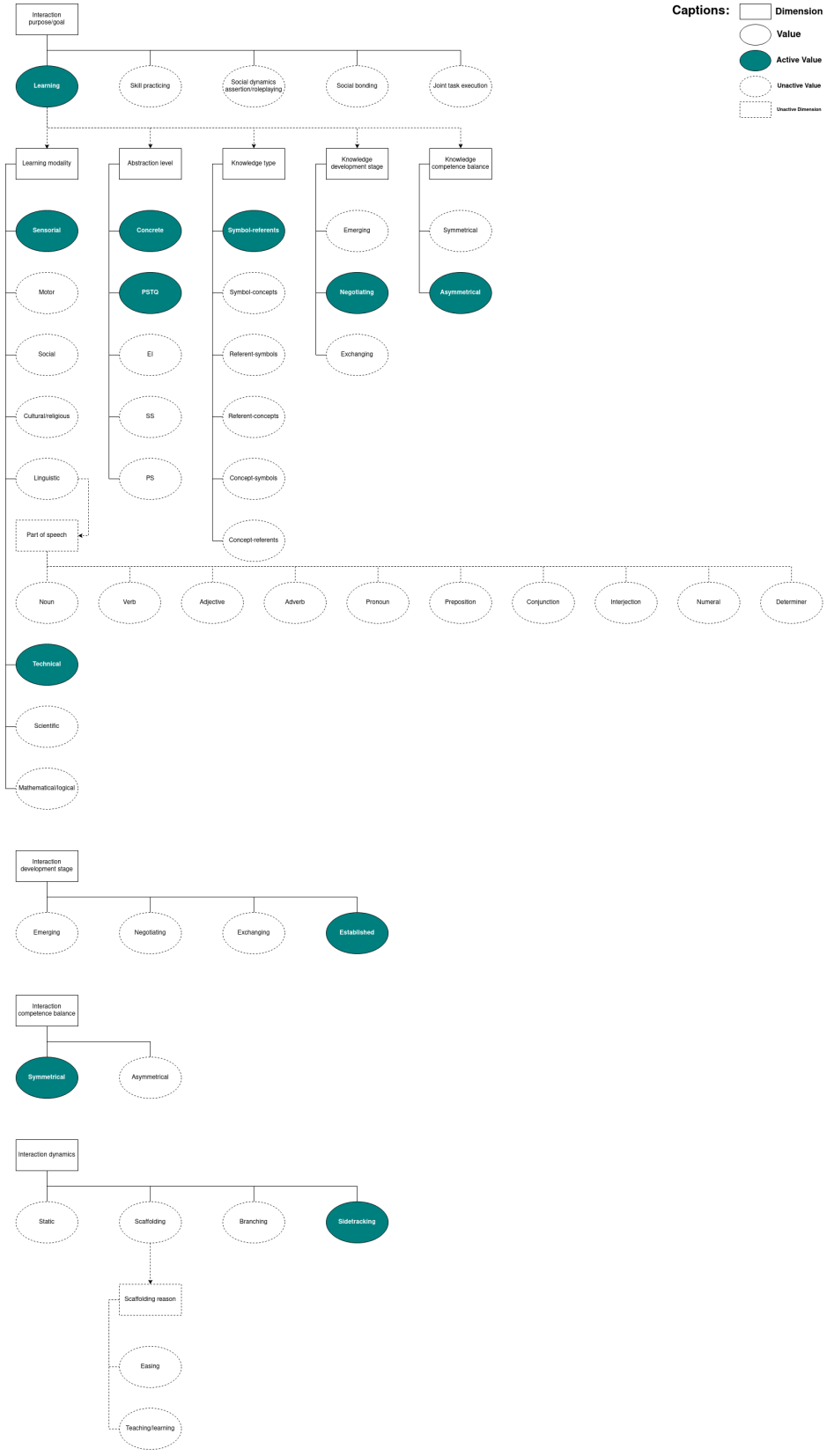
Source: Own authorship (2022).

Figure 19 – Classification of the PF in Steels e Kaplan (2000)



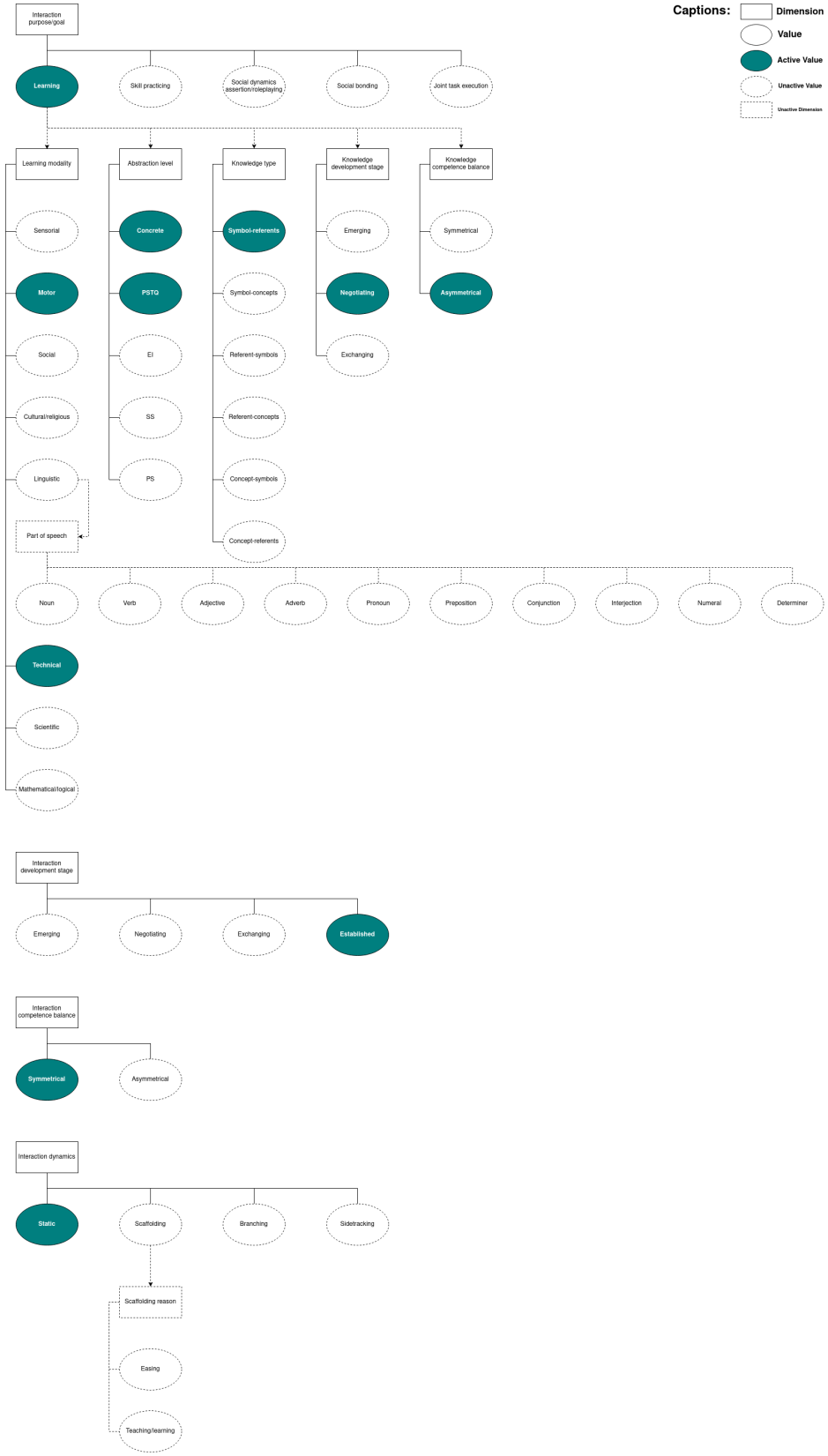
Source: Own authorship (2022).

Figure 20 – Classification of the PF in Calinon e Billard (2006)



Source: Own authorship (2022).

Figure 21 – Classification of the PF in Cakmak e Thomaz (2012)



Source: Own authorship (2022).

APPENDIX B – Algorithms for the actions of the proposed Pragmatic Frame

B.1 First part of the interaction

Algorithm 3 – TR

Input: objects O, human agent g_h
Output: chosen object index o_{i1}
1: $o_{i1} \leftarrow index(choose(O))$
2: $send(g_h, o_{i1})$

Source: Own authorship (2022).

Algorithm 4 – RRC1, with probability p of mutating o_i.

Input: objects O, robot agent g_r
Output: received object index o_i
1: $o_i \leftarrow recv(g_r)$
2: $o_i \leftarrow mutate(o_i, \{index(o') | \forall o' \in O\}, p)$
3: $send(g_r, o_i \cup "?")$

Source: Own authorship (2022).

Algorithm 5 – RRC2

Input: human agent g_h, chosen object index o_{i1}
Output: robot action a_r, confirmed object index o_{i2}
1: $o_{i2} \leftarrow recv(g_h) - "?"$
2: **if** $o_{i1} = o_{i2}$ **then**
3: $\quad a_r \leftarrow CR$
4: **else**
5: $\quad a_r \leftarrow TR$
6: **end if**

Source: Own authorship (2022).

Algorithm 6 – CR

Input: human agent g_h
1: $send(g_h, "True")$

Source: Own authorship (2022).

Algorithm 7 – TW

Input: objects O, robot agent g_r, received object index o_i
Output: chosen word w_1
1: $w_1 \leftarrow choose(names(o' \in O | index(o') = o_i))$
2: $send(g_r, w_1)$

Source: Own authorship (2022).

Algorithm 8 – RWC1, with probability p of mutating w.

Input: human agent g_h
Output: received word w
1: $w \leftarrow recv(g_h)$
2: $w \leftarrow mutate(w, p)$
3: $send(g_h, w \cup "?")$

Source: Own authorship (2022).

Algorithm 9 – RWC2

Input: robot agent g_r, chosen word w_1
Output: human action a_h, confirmed word w_2
1: $w_2 \leftarrow recv(g_r) - "?"$
2: **if** $w_1 = w_2$ **then**
3: $\quad a_h \leftarrow CW1$
4: **else**
5: $\quad a_h \leftarrow TW$
6: **end if**

Source: Own authorship (2022).

Algorithm 10 – CW1

Input: robot agent g_r
1: $send(g_r, "True")$

Source: Own authorship (2022).

Algorithm 11 – CW2

Input: knowledge base K_b, chosen object index o_{i1}, received word w
1: $K_b \leftarrow K_b \cup \{\langle o_{i1}, w \rangle\}$

Source: Own authorship (2022).

B.2 Second part of the interaction

Algorithm 12 – TIR

Input: objects O, human agent g_h
Output: chosen object index o_{i1}
1: $o_{i1} \leftarrow index(choose(O))$
2: $send(g_h, o_{i1})$

Source: Own authorship (2022).

Algorithm 13 – RIRC1, with probability p of mutating o_i.

Input: objects O, robot agent g_r
Output: received object index o_i
1: $o_i \leftarrow recv(g_r)$
2: $o_i \leftarrow mutate(o_i, \{index(o') | \forall o' \in O\}, p)$
3: $send(g_r, o_i \cup "?")$

Source: Own authorship (2022).

Algorithm 14 – RIRC2

Input: human agent g_h, chosen object index o_{i1}
Output: robot action a_r, confirmed object index o_{i2}
1: $o_{i2} \leftarrow recv(g_h) - "?"$
2: **if** $o_{i1} = o_{i2}$ **then**
3: $\quad a_r \leftarrow CIR$
4: **else**
5: $\quad a_r \leftarrow TIR$
6: **end if**

Source: Own authorship (2022).

Algorithm 15 – CIR

Input: human agent g_h
1: $send(g_h, "True")$

Source: Own authorship (2022).

Algorithm 16 – RIC1

Input: knowledge base K_b, human agent g_h
Output: chosen word w
1: $w \leftarrow choose(\bigcup_{\langle o', w' \rangle \in K_b} \{w'\})$
2: $send(g_h, w \cup "?")$

Source: Own authorship (2022).

Algorithm 17 – RIC2

Input: objects O, robot agent g_r, received object index o_i
Output: human action a_h, confirmed word w_2
1: $w_2 \leftarrow recv(g_r) - "?"$
2: **if** $w_2 \in names(o' \in O | index(o') = o_i)$ **then**
3: $\quad a_h \leftarrow CI1$
4: **else**
5: $\quad a_h \leftarrow TW$
6: **end if**

Source: Own authorship (2022).

Algorithm 18 – CI1

Input: robot agent g_r
1: $send(g_r, "True")$

Source: Own authorship (2022).

Algorithm 19 – CI2

Input: knowledge base K_b, chosen object index o_{i1}, chosen word w
1: $K_b \leftarrow K_b \cup \{\langle o_{i1}, w \rangle\}$

Source: Own authorship (2022).

APPENDIX C – Algorithms for the robot main loop

Algorithm 20 – $possible\_actions$ **that match the message's syntactic structure.**

Input: message m, interaction i
Output: set of possible actions a_p

if $is\_integer(m)$ **and** $ends\_with(m, "?")$ **then**
  if $i = FirstInteraction$ **then**
    $a_p \leftarrow \{RRC1\}$
  else if $i = SecondInteraction$ **then**
    $a_p \leftarrow \{RRC1, RIRC1\}$
  end if
else if $is\_string(m)$ **and not** $ends\_with(m, "?")$ **then**
  $a_p \leftarrow \{TW\}$
else if $m = "True"$ **then**
  if $i = FirstInteraction$ **then**
    $a_p \leftarrow \{CW1\}$
  else if $i = SecondInteraction$ **then**
    $a_p \leftarrow \{CW1, CI1\}$
  end if
end if

Source: Own authorship (2022).

Algorithm 21 – $guess\_next\_action$ **greedily, with probabilities proportional to parameter** x**.**

Input: current model $\mathcal{M}$, current action a_c, interaction i, possible actions a_p
Output: next action a_n

if $a_p = \varnothing$ **then**
  $a_p \leftarrow \{TR, RRC1, RRC2, CR, TW, RWC1, RWC2, CW1, CW2\}$
  if $i = SecondInteraction$ **then**
    $a_p \leftarrow a_p \cup \{TIR, RIRC1, RIRC2, CIR, RIC1, RIC2, CI1, CI2\}$
  end if
end if
for all $a \in a_p$ **do**
  if $can\_apply(\mathcal{M}, a_c, i, a)$ **then**
    $p(a|\mathcal{M},a_c,i) \leftarrow x$ {Probability proportional to x}
  else
    $p(a|\mathcal{M},a_c,i) \leftarrow 1$ {Probability proportional to 1}
  end if
end for
$normalize(p(|\mathcal{M},a_c,i))$ {Make all probabilities sum up to 1}
$a_n \leftarrow choose(a_p, p(|\mathcal{M},a_c,i))$

Source: Own authorship (2022).

Algorithm 22 – $guess\_next\_action$ **performing a LD-DFS, with probabilities proportional to parameters** x **and** g**.**

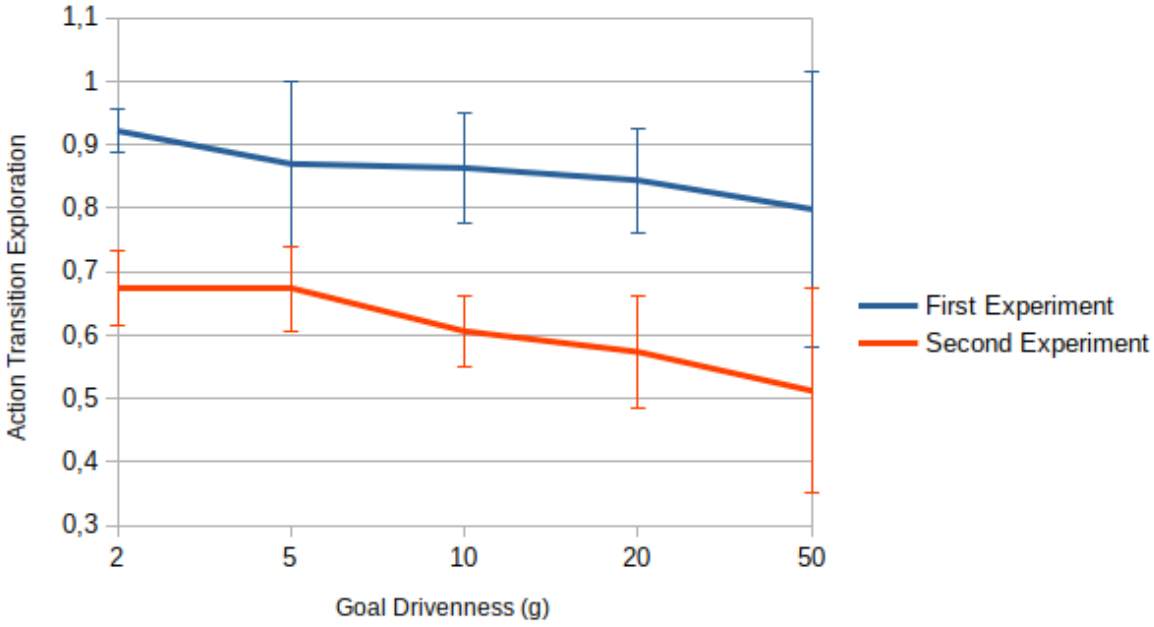
Input: current model $\mathcal{M}$, current action a_c, interaction i, possible actions a_p
Output: next action a_n

$A_n \leftarrow \{TR, RRC1, RRC2, CR, TW, RWC1, RWC2, CW1, CW2\}$ {Set of possible next actions for each step in the trajectory}
if $i = SecondInteraction$ **then**
  $A_n \leftarrow A_n \cup \{TIR, RIRC1, RIRC2, CIR, RIC1, RIC2, CI1, CI2\}$
end if
if $a_p = \varnothing$ **then**
  $a_p \leftarrow A_n$
end if
for all $a \in a_p$ **do**
  $A_s \leftarrow \langle\{a\}\rangle$ {Stack of unexplored next actions for each step in the trajectory}
  $W_s \leftarrow \langle\langle 0, 0\rangle\rangle$ {Stack of accumulated weights for unexplored next actions for each step in the trajectory}
  $t_s \leftarrow \langle a_c\rangle$ {Current trajectory}
  while $|A_s| > 0$ **do**
    $a_s \leftarrow pop(top(A_s))$
    {Check if a_s is a goal action}
    if $a_s \in \{CW2, CI2\}$ **then**
      if $can\_apply(\mathcal{M}, top(t_s), i, a_s)$ **then**
        $top(W_s) \leftarrow top(W_s) + \langle x \times g, 1\rangle$ {Probability proportional to $x \times g$}
      else
        $top(W_s) \leftarrow top(W_s) + \langle g, 1\rangle$ {Probability proportional to g}
      end if
    else if $can\_apply(\mathcal{M}, top(t_s), i, a_s)$ **then**
      $push(A_s, A_n)$ {Expand subtree}
      $push(W_s, \langle 0, 0\rangle)$
      $push(t_s, apply\_action(top(t_s), a_s))$
      continue
    else
      $top(W_s) \leftarrow top(W_s) + \langle 1, 1\rangle$ {Probability proportional to 1}
    end if
    {End of subtree}
    while $|A_s| > 0$ **and** $|top(A_s)| = 0$ **do**
      $pop(A_s)$
      $w_s \leftarrow pop(W_s)$
      if $w_s[2] > 0$ **then**
        $x_s \leftarrow \frac{x \times w_s[1]}{w_s[2]}$ {Probability proportional to the average of probabilities of its children trajectories, multiplied by x}
        if $|W_s| > 0$ **then**
          $top(W_s) \leftarrow top(W_s) + \langle x_s, 1\rangle$
        else
          $push(W_s, x_s)$ {Final weight for a}
        end if
      end if
      $pop(t_s)$
    end while
  end while
  {Final weight for a}
  if $|W_s| > 0$ **then**
    $p(a|\mathcal{M},a_c,i) \leftarrow pop(W_s)$
  else
    $p(a|\mathcal{M},a_c,i) \leftarrow 1$ {Probability proportional to 1}
  end if
end for
$normalize(p(|\mathcal{M},a_c,i))$ {Make all probabilities sum up to 1}
$a_n \leftarrow choose(a_p, p(|\mathcal{M},a_c,i))$

Source: Own authorship (2022).

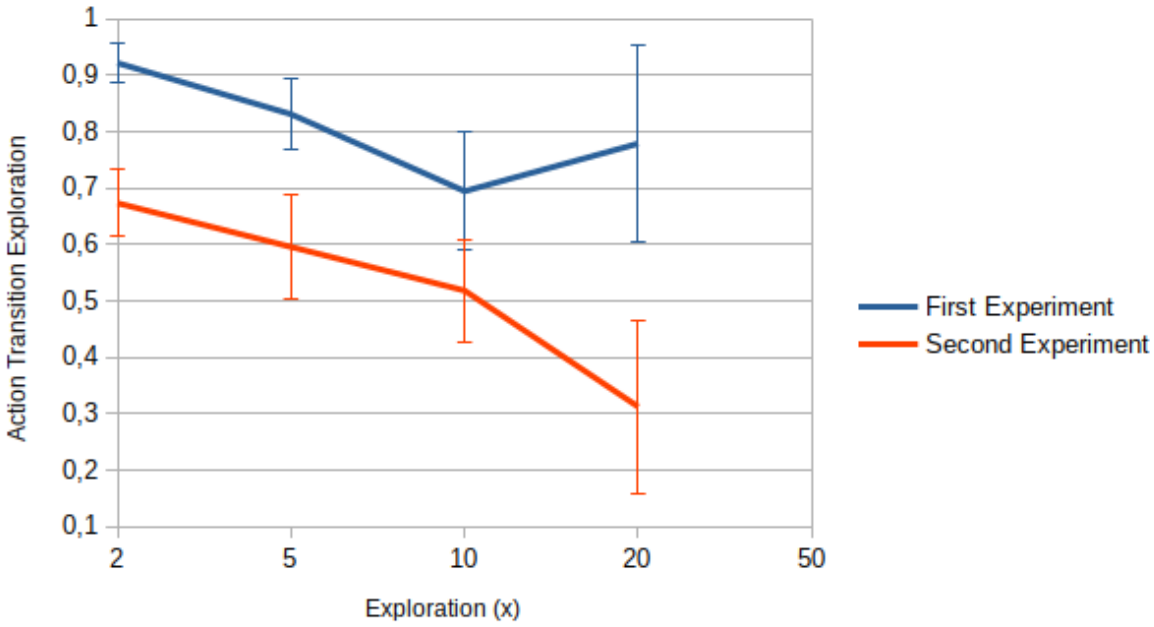
APPENDIX D – Graphs of results

Graph 6 – Action transition exploration for LD-DFS action-choosing over g.



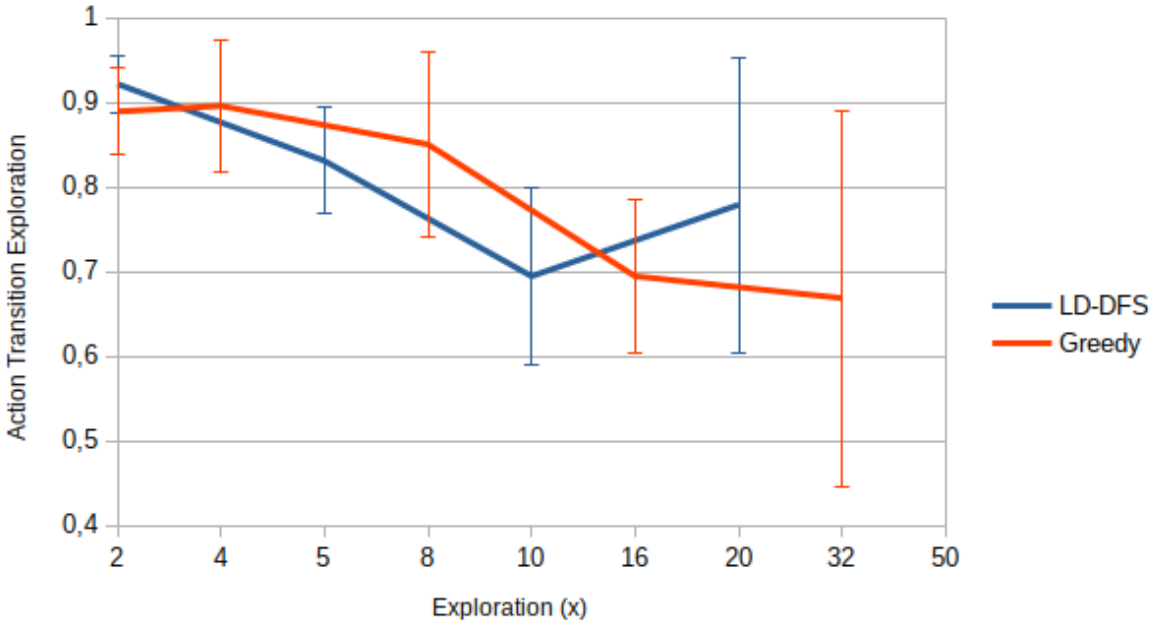
Source: Own authorship (2022).

Graph 7 – Action transition exploration for LD-DFS action-choosing over x.



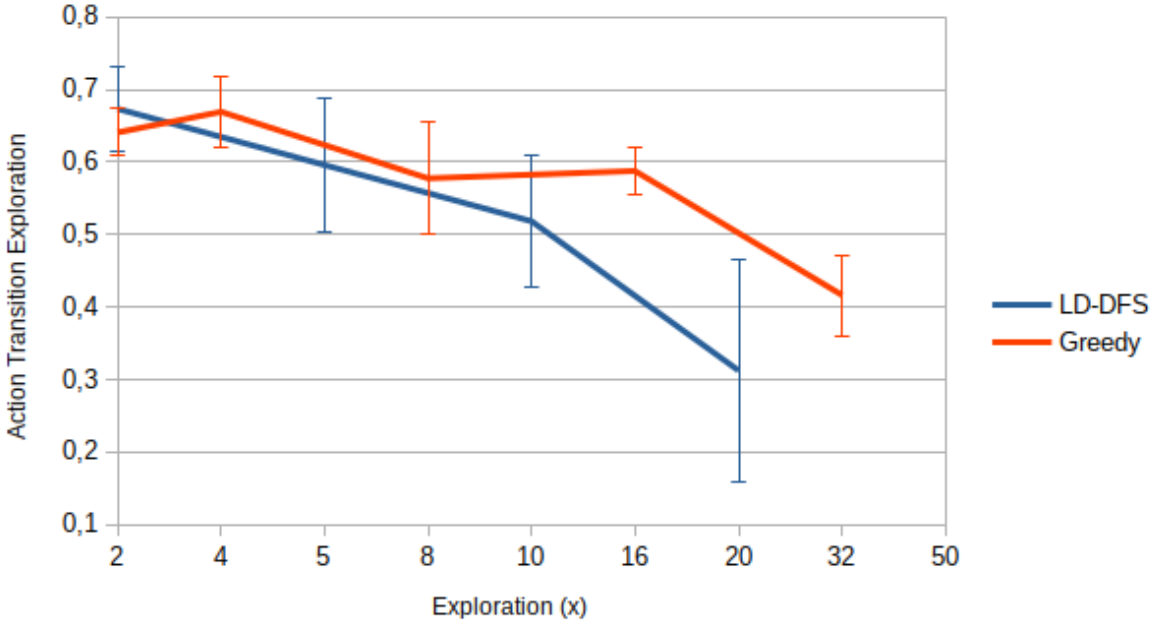
Source: Own authorship (2022).

Graph 8 – Action transition exploration of first experiment.



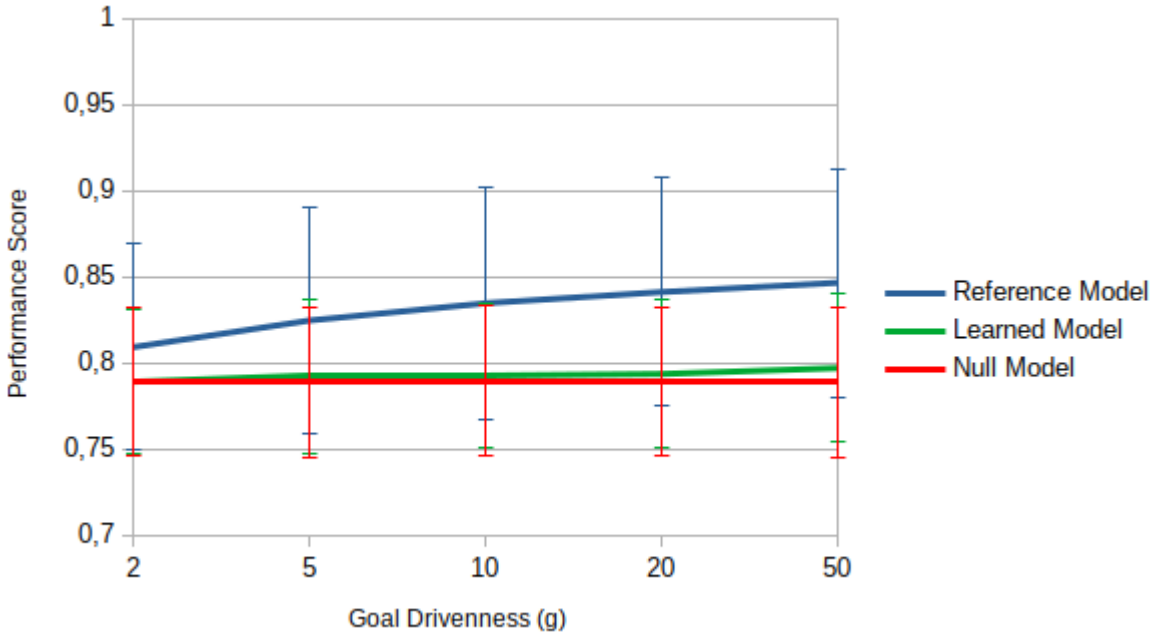
Source: Own authorship (2022).

Graph 9 – Action transition exploration of second experiment.



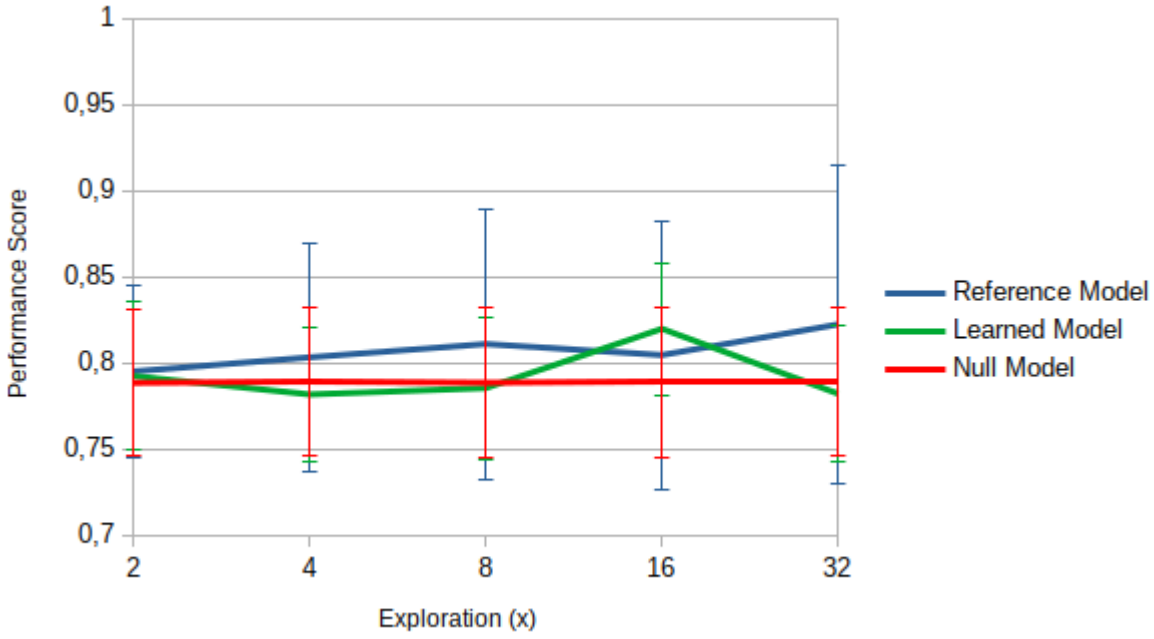
Source: Own authorship (2022).

Graph 10 – Performance score for LD-DFS action-choosing over g, first experiment.



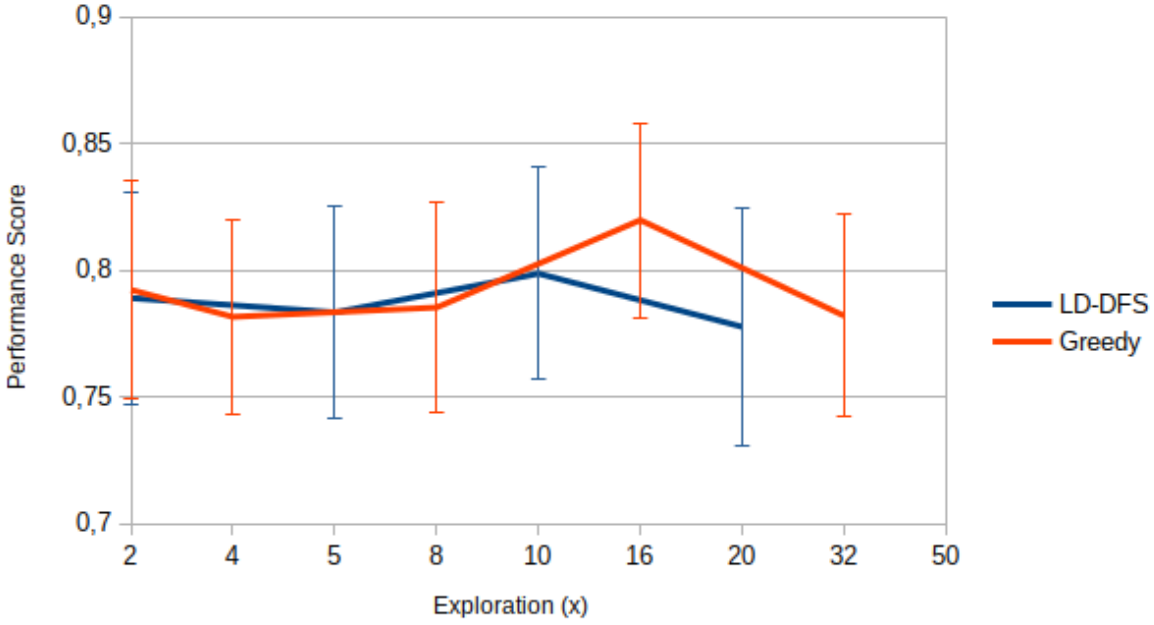
Source: Own authorship (2022).

Graph 11 – Performance score for greedy action-choosing, first experiment.



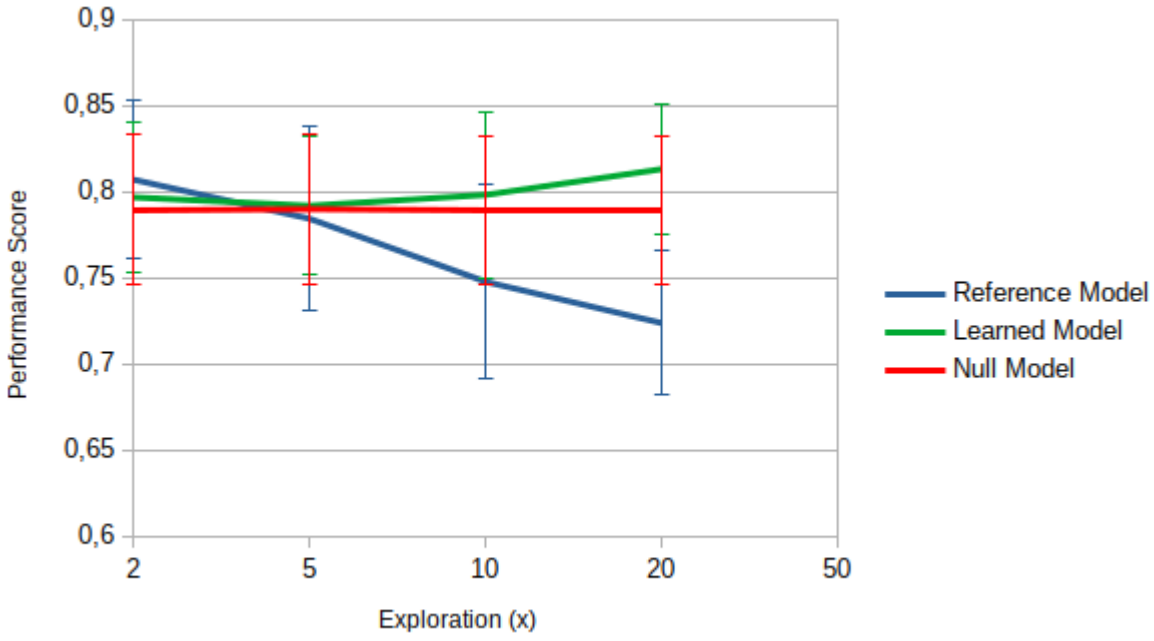
Source: Own authorship (2022).

Graph 12 – Performance score of learned model, first experiment.



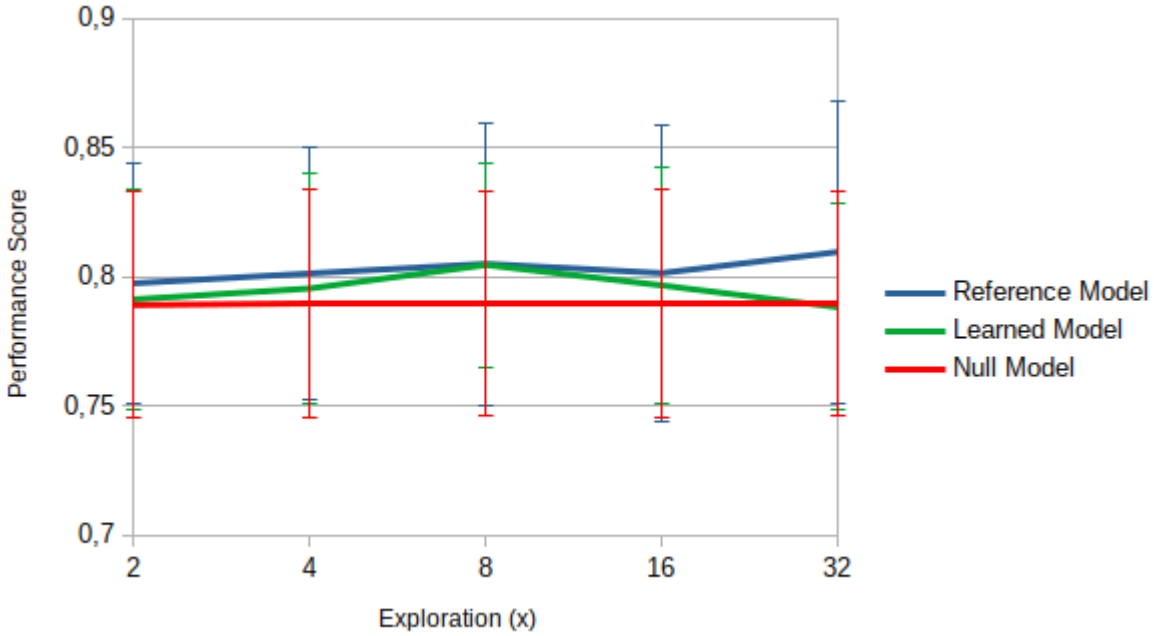
Source: Own authorship (2022).

Graph 13 – Performance score for LD-DFS action-choosing over x, second experiment.



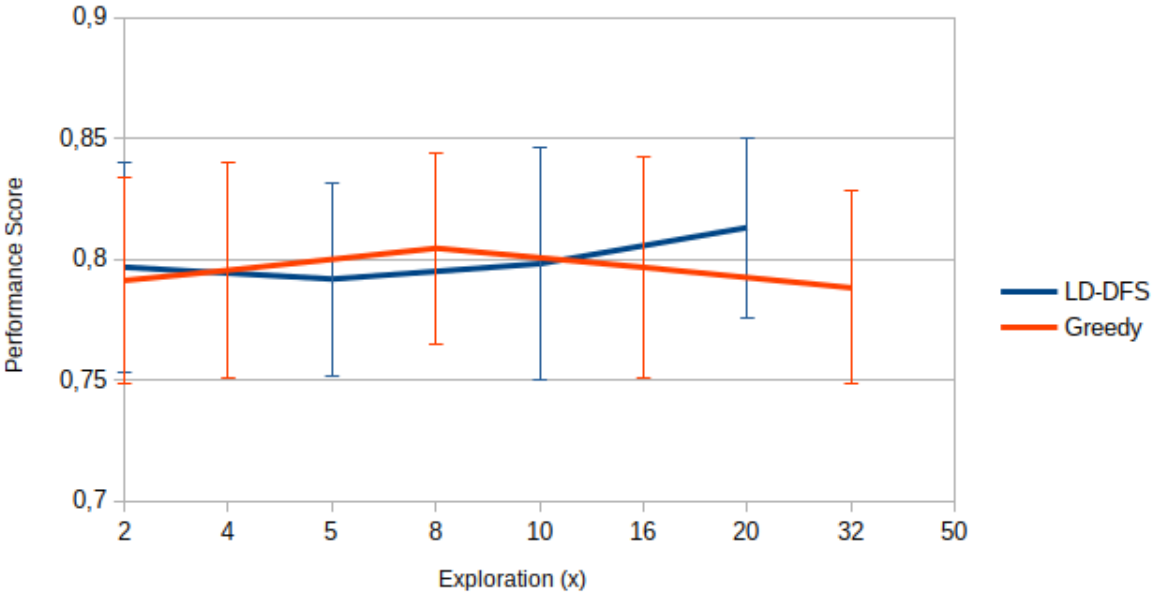
Source: Own authorship (2022).

Graph 14 – Performance score for greedy action-choosing, second experiment.



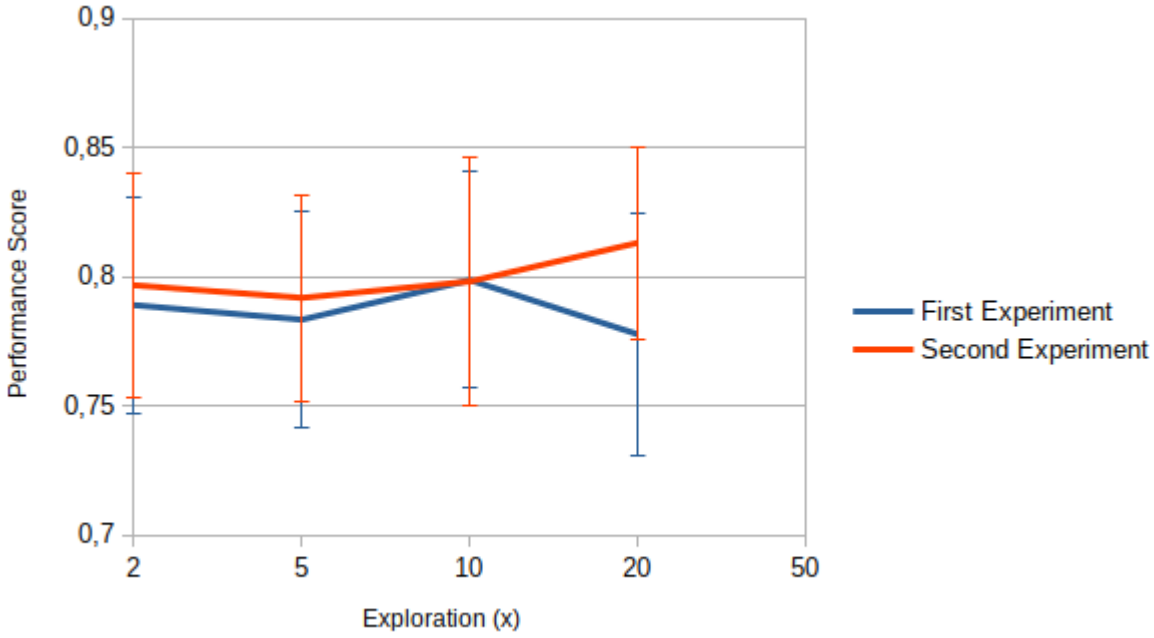
Source: Own authorship (2022).

Graph 15 – Performance score of learned model, second experiment.



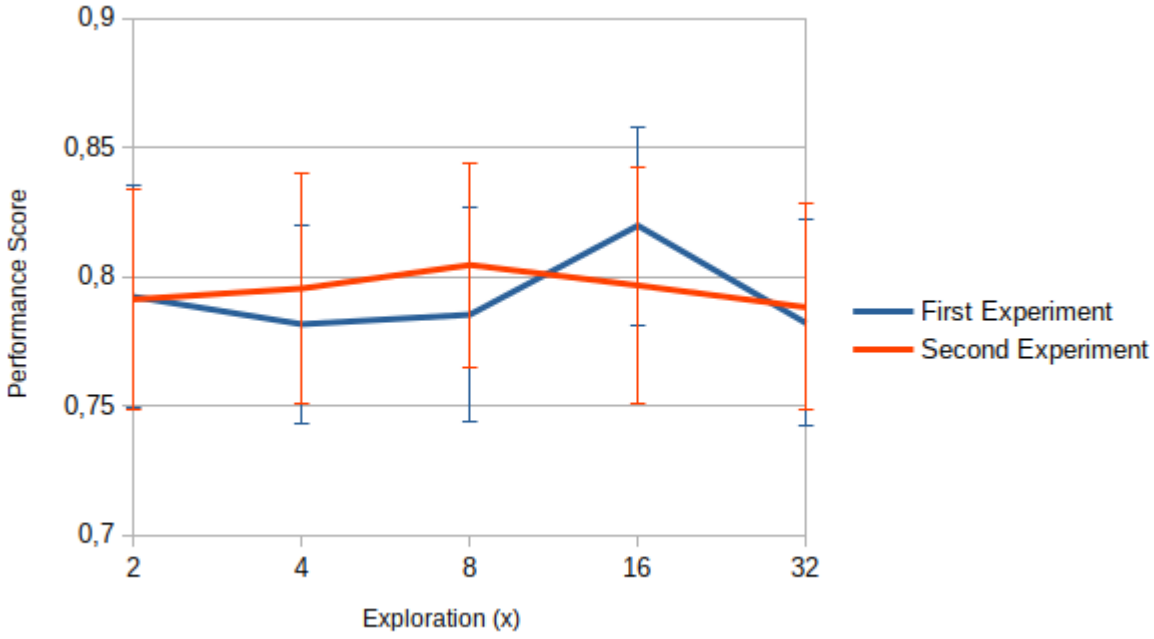
Source: Own authorship (2022).

Graph 16 – Performance score of learned model with LD-DFS action-choosing over x.



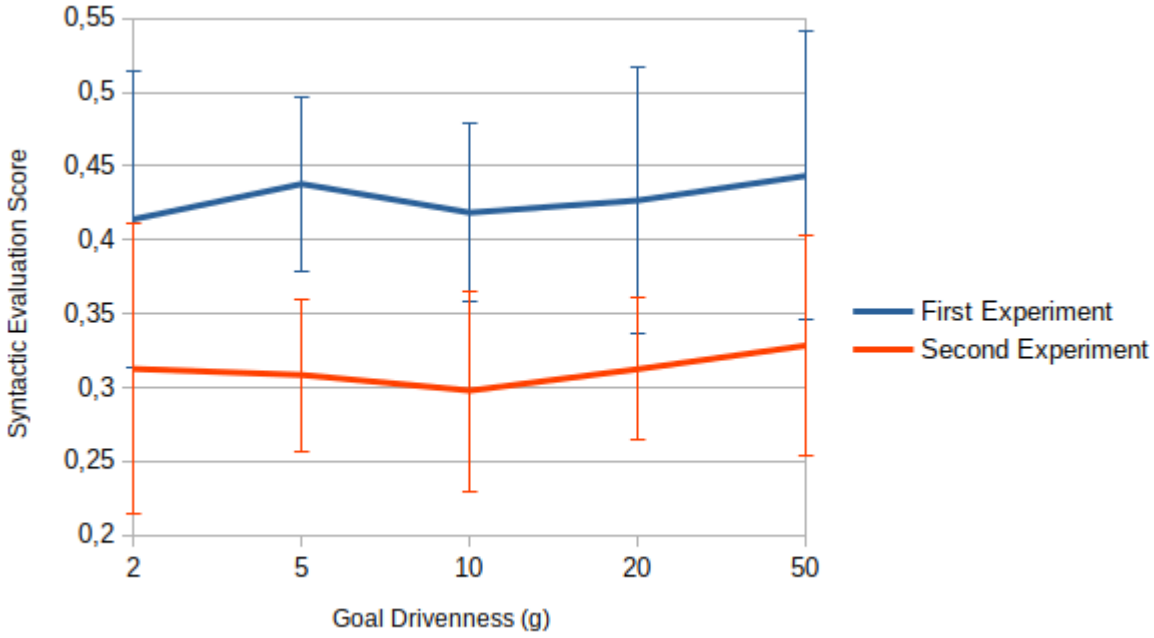
Source: Own authorship (2022).

Graph 17 – Performance score of learned model with greedy action-choosing.



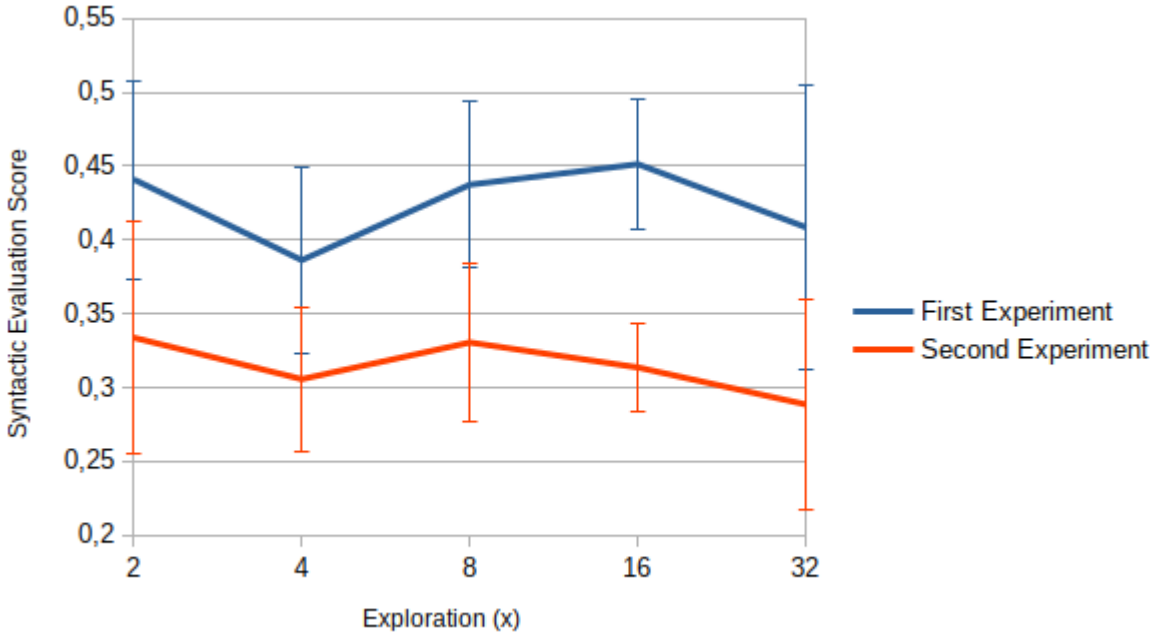
Source: Own authorship (2022).

Graph 18 – Syntactic evaluation score for LD-DFS action-choosing over g.



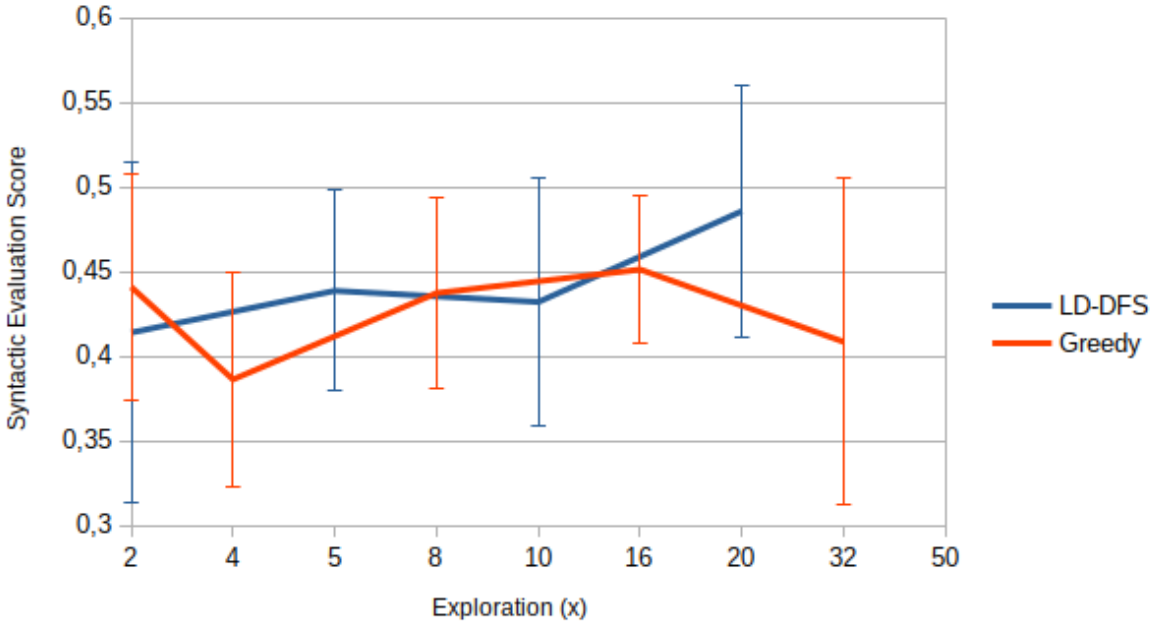
Source: Own authorship (2022).

Graph 19 – Syntactic evaluation score for greedy action-choosing.



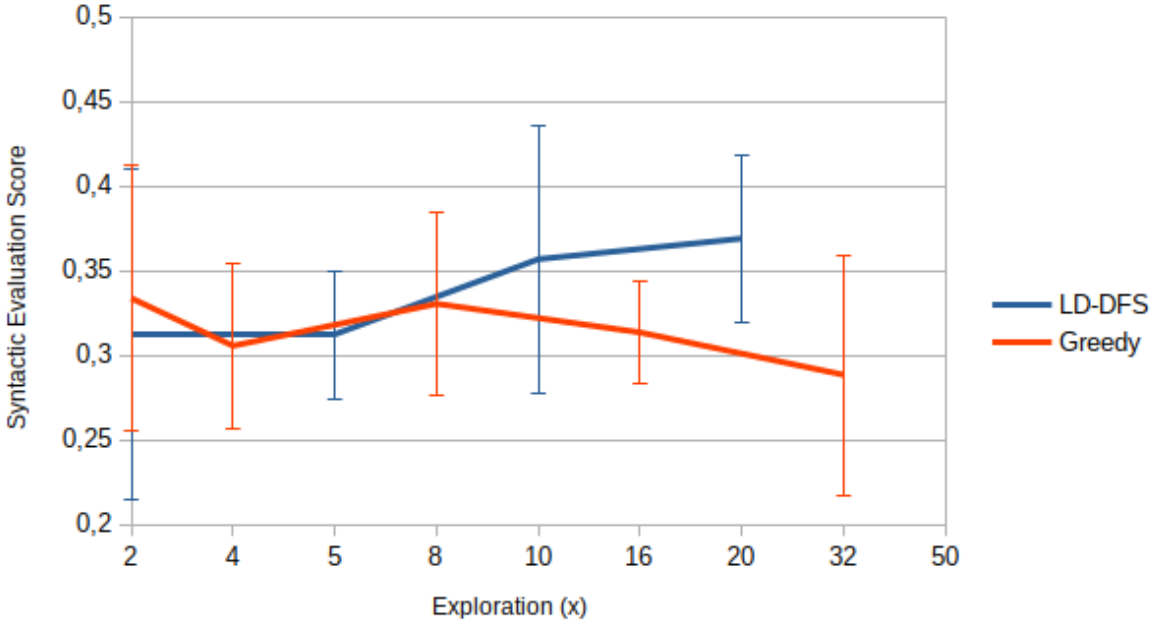
Source: Own authorship (2022).

Graph 20 – Syntactic evaluation score of first experiment.



Source: Own authorship (2022).

Graph 21 – Syntactic evaluation score of second experiment.



Source: Own authorship (2022).