Reykjavík University

Department of Computer Science

University of Camerino

School of Science and Technology

Master of Science in Computer Science

Towards a Theory of Causally Grounded Tasks

Candidate
Matteo Belenchia
Student ID: 110687

Supervisors
Prof. Dr. Kristinn R. Thórisson
Prof. Dr. Emanuela Merelli

A.Y. 2020/2021

Abstract

Artificial Intelligence systems that need to operate in the physical world require a firm understanding of causal relationships to efficiently carry out their tasks. At present, few cognitive architectures, artificial intelligence, or control systems consider causal relationships between phenomena for achieving goals in the real world, and few if any tests exist to verify understanding of these relationships in intelligent learners.

The goal of this thesis is to outline a theory of tasks grounded in models of real-time causal relationships, intended to be used to analyze tasks of varying complexity and build tests for evaluating the ability of artificial systems to learn, perform, and understand complex real-world task-environments. Like an engineer can design a bridge with certain physical properties without actually building it, such a task theory, grounded in physics, should enable researchers to evaluate and compare tasks and systems without having to resort physical experiments.

Historically, research on the principles of tasks has been scarce, and causal relationships have never been a primary focus in those attempts. Few tests of causal understanding have been proposed thus far and the lack of a proper theory of tasks limits their utility. No comprehensive theory of tasks exists. We envision that such a theory would enable comparison of similar and different asks, calculation of task difficulty for particular learners, and prescriptive ways for modifying existing tasks to make them more tractable for particular performers and environments. Comparison of tasks would be invaluable when evaluating and considering the pros and cons of various approaches to AI systems and learners.

We base our theory of tasks on previous work on causal analysis and cumulative learning, proposing a method for computing complexity and comparability using causal interpretation of directed acyclic graphs (DAGs). The theory allows tasks to be analyzed along different dimensions and can be composed and decomposed into subgoals. It is intended to be used for evaluating and testing intelligent agents in various ways.

Keywords: *Tasks, Environments, Task theory, Causality, Artificial Intelligence, General Machine Intelligence*

Contents

Listings

List of Figures

Acronyms

AERA	Autocatalytic Endogenous Reflective Architecture
AGI	Artificial General Intelligence
AI	Artificial Intelligence
AIKR	Assumption of Insufficient Knowledge and Resources
DAG	Directed Acyclic Graph
GMI	General Machine Intelligence
NARS	Non-Axiomatic Reasoning System
SCM	Structural Causal Model

Glossary

Agent An **agent** is an embodied system consisting of a controller (the mind) and a body. The body is the agent's interface to the **world** which allows the perception of the external **environment**, through the flow of data from the body's sensors to the controller, and the execution of atomic actions, by means of the commands sent from the controller to the body's actuators. The body contains two lists of variables that the controller can read and write to: $B = \langle V_S, V_A \rangle$. Since the body is a physical entity, its sensors and actuators are physical objects in the world as well and are treated as such (Thórisson, Bieger, et al., 2016).

Environment An **environment** is a view of a **world**. The body of an agent is considered to be part of it. (Thórisson, Bieger, et al., 2016).

Failure A **failure** state is an undesirable, possibly partial, **state** that the agent should avoid (Thórisson, Bieger, et al., 2016).

Goal A **goal** state is a desirable, possibly partial, **state** that the agent should reach (Thórisson, Bieger, et al., 2016).

Phenomenon A **phenomenon** (process, state of affairs, occurrence) Φ, where W is the **world** and $\Phi \subset W$, is composed of a set of elements $\{\phi_1, \phi_2, ..., \phi_n \in \Phi\}$ of various kinds including relations $\mathfrak{R}_\Phi$ that couple elements of Φ with each other and with those of other phenomena. The elements that a phenomenon is made up of can be any subdivision of Φ, including sub-structures, causal relations, whole-part relations and so on. The relations $\mathfrak{R}_\Phi \subseteq 2^W \times 2^W$ that extend to other phenomena identify the phenomenon's *context.* The set of relations can be partitioned in *inward facing* relations $\mathfrak{R}_\Phi^{in} = \mathfrak{R}_\Phi \cap (2^\Phi \times 2^\Phi)$ and *outward facing* relations $\mathfrak{R}_\Phi^{out} = \mathfrak{R}_\Phi \setminus \mathfrak{R}_\Phi^{in}$ (Thórisson, Kremelberg, et al., 2016).

Problem A **problem** can be *atomic* or *compound.* An atomic problem is specified by an initial **state**, **goal** states and **failure** states. A compound problem can be created by composition of atomic problems using operators such as conjunction, disjunction and negation. A problem for which a solution is known to exist is called a closed problem (Thórisson, Bieger, et al., 2016).

Problem space The problem space is the set of all valid states of the task.

Solution A **solution** is a sequence of atomic actions that results in a path through the **state** space that reaches all of the **goal** states and none of the **failure** states (Thórisson, Bieger, et al., 2016).

Solution space The solution space is the subset of the problem space defined by the task's goals and constraints, made up of all the solution states reachable from any initial state of the task.

State A **state** can be *concrete* or *partial*. A *concrete* state S is a value assignment to all of the variables in a **task-environment**: $S = \bigcup_{v \in V}\{\langle v, x_v \mid x_v \in d_v \rangle\}$ A *partial* state S^- only assigns values to a subset of the variables. When considering real variables partial states can be represented using error bounds: $S^- = \bigcup_{v \in V^-}\{\langle v, x_l, x_u \mid x_l < x_u \wedge (x_l, x_u) \subseteq d_v \rangle\}$; this way a partial state covers a set of concrete states. A state is valid if and only if all invariant relations hold: valid$(S) \iff \forall_{r \in R} r(S)$. In practice the presence of noise and the partial observability of variables makes the use of partial states more practical than concrete states, therefore by state is always meant a partial state unless otherwise noted (Thórisson, Bieger, et al., 2016).

Task A **task** is a problem assigned to an **agent**, $T = \langle S_0, \mathcal{G}_{top}, \mathcal{G}_{sub}, G^-, B, t_{go}, t_{stop}, I \rangle$, where S_0 is the set of permissible initial states, $\mathcal{G}_{top}$ is the task's set of top-level goals, $\mathcal{G}_{sub}$ is the set of given sub-goals, G^- is its set of constraints, B is a controller's body, and t refers to the permissible start and stop times of the task. An *assigned* task will have all its variables bound and reference an agency that is to perform it (accepted assignments having their own timestamp t_{assign}). This assignment includes the manner in which the task is communicated to the agent, for example if the agent is given a description of the task a priori, receives additional hints or if it only gets incremental reinforcement signals as certain **states** are reached. A task is performed successfully when the **world**'s history contains a path of states that solved the problem (Thórisson, Bieger, et al., 2016).

Task-Environment By **task-environment** is meant the tuple of a **task** and the **environment** in which it is to be performed. The separation of a task from its environment is not always clear and somewhat arbitrary, therefore the term task-environment is used to encompass all the relevant aspects of both (Thórisson, Bieger, et al., 2016; Bieger and Thórisson, 2017).

World A **world** W is a interactive system consisting of a set of variables V, dynamics functions F, an initial state S_0, domains D of possible clusters of particular constraints on their values, and a set of relations between the variables R: $W = \langle V, F, S_0, D, R \rangle$. The variables $V = \{v_1, v_2, ..., v_{||V||}\}$ represent anything that may change or hold a particular value in the world. The dynamics functions act as the laws of nature in the world and as a whole can be seen as an automatically executed function that periodically or continually evolves the world's current state into the next: $S_{t+\delta} = F(S_t)$. In practice it is useful to the decompose the dynamics into a set of transition functions: $F = \{f_1, f_2, ..., f_n\}$ where $f_i : S^- \to S^-$ and S^- is a partial state. The domains $d_v \in D$ specify which values each variable v can take, and for physical domains these are usually subsets of real numbers. The relations are Boolean functions over variables that hold true in any state the world will ever find itself in. If the world is a closed system with no outside interference, the domains and relations are implicitly fully determined by the dynamics functions and the initial state. In an open system where changes can be caused externally, instead, the explicit definition of domains and invariant relations can restrict the range of possible interactions (Thórisson, Bieger, et al., 2016).

Nomenclature

X	Random variable
x	Value of a random variable
$\mathbf{X}$	Vector
$P(Y \mid X = x)$	Conditional distribution
$P(Y \mid \text{do}(X = x))$	Intervention distribution
$P(Y \mid Z = \hat{z}, X = \hat{x}, \text{do}(X = x))$	Counterfactual distribution

CHAPTER 1

Introduction

The purpose of an AI system is to execute tasks, in particular in the physical world. The concept of task is thus fundamental for Artificial Intelligence as tasks are necessary for training and evaluation of intelligent systems. Tasks are important both for narrow AI systems, which are built to carry out the single task they have been designed for, and general AI systems, which deal with a wide range of tasks, unknown at design time. Unlike other engineering fields, little work has been done to derive a task theory for AI, a theory which can be used to specify properties of tasks, compare tasks, estimate resource requirements for their solution and compose/decompose them, to mention a few (Thórisson, Bieger, et al., 2016).

The lack of a task theory has made the comparison of different AI systems largely impractical; such comparisons are costly and difficult to perform and rarely carried out at all. When AI systems are evaluated, it is often on a task tailored for the specific system. While this situation might work out for narrow-AI systems, such is not the case for GMI research: AGI-aspiring systems need to be designed and tested on a wide range of task-environments, and domain knowledge or psychometric tests such as I.Q. tests, the Lovelace test (Riedl, 2014) or the Turing test (Turing, 1950) don't nearly cover the breadth of situations these systems will find themselves in. For the purpose of moving towards AGI, a task theory that allows the analysis and comparison of tasks along the relevant axes is thus required.

This thesis is a first step in that direction: to approach the definition of a task theory, we try to focus on causal relationships, which are of fundamental importance to get things done in the real world, and therefore are of primary interest for embodied AGI-aspiring systems. On the topic of causality, we refer to Turing award winner Judea Pearl's work on graphical causal diagrams and structural causal models for causal analysis in statistics (Pearl, 1988; Pearl, 2009; Pearl and Bareinboim, 2014).

The thesis is structured in four chapters:

1. In Chapter 1, we present the field of Artificial Intelligence, we describe what we mean by task theory, what are its uses and requirements, and we conclude with a brief historical overview of its development;

2. In Chapter 2, we present the necessary background notions required for the following chapters: bi-directional causal-relational models (Thórisson and Talbot, 2018a; Thórisson and Talbot, 2018b), understanding (Thórisson, Kremelberg, et al., 2016; Bieger and

Thórisson, 2017) and causality (Pearl, 2009).

3. In Chapter 3, lays the novelty of this thesis: most importantly we describe an extension of causal diagrams to allow their use to describe tasks, we set forth the foundational principles for moving towards a task theory, we define a measure of a task's "complicatedness" that only takes the task's physics into account and we define a measure of difficulty of a task's execution for a specific performing controller.

4. In Chapter 4, we conclude our work by pointing out some approaches to test the theory, trace a tentative connection to Constructor Theory (Deutsch, 2013), respond to a few criticisms that were raised against our work and lastly we give some pointers for future work and the issues left to address.

1.1 Artificial Intelligence and the quest for Generality

Artificial Intelligence (AI) is the field dedicated to the design and development of systems that can perform tasks requiring some degree of 'intelligence'. Intelligence is a natural phenomenon, which is most commonly associated with human minds but is also a characteristic of many animals (Thórisson, 2020b). Intelligence can be defined in many ways, but the definition we would like to refer to for this work is the one given by Legg and Hutter (2007):

> "Intelligence measures an agent's ability to achieve goals in a wide range of environments." (Legg and Hutter, 2007)

This definition of intelligence already implies that there is some association between the intelligence of a system and its generality: an agent that can achieve more goals in more environments is intuitively more intelligent than an agent with a narrower horizon of use. The goal of artificial general intelligence (AGI) is to develop systems that can "learn to perform multiple a-priory unknown tasks in multiple unknown environment" (Nivel, Thórisson, et al., 2013, p. 3)[1]. In contrast, most intelligent systems today (e.g. machine learning and deep learning systems) are limited to a single, pre-defined task in an unchanging, pre-defined fixed environment (Nivel, Thórisson, et al., 2013). A step further is taken by Wang (2019), where the emphasis of general intelligence is on the handling of novelty and "figuring things out" (Thórisson, 2020a):

"Intelligence is the capacity of an information-processing system to adapt to its environment while operating with insufficient knowledge and resources." (Wang, 2019, p. 17)

Given that even room temperature controllers "achieve goals in a wide range of situations," Wang's definition more robustly differentiates general intelligence – the kind we normally associate with the concept – from other controllers and processes that might also qualify. However, given how recent his definition (in spite of tracing its roots to 1996) is, it is not yet widely accepted.

In Artificial Intelligence it is possible to talk about three different types of controllers and agents agent architectures: reactive, predictive and reflective architectures.

Reactive agents Reactive agents only respond to the perceived sensory information from the environment, obtained through their sensors. Their architecture is mostly fixed through

[1]In other words, agents with human-level intelligence, as humans are the only known beings possessing this kind of intelligence.

their lifetime, and while learning is possible the agent only ever reacts to stimuli and is incapable of proactive behavior[2]. Most AI architectures are reactive, and examples of this type of systems span the very simple thermostats to the complex control systems of power plants. These types of systems are limited in the sense that they are built with an embedded model of the task they carry out which is unchangeable, bar the customization of a few parameters during run-time (Thórisson, 2020c).

Predictive agents Predictive agents are able to anticipate environment states and act in anticipation of sensory information. Their architecture is mostly fixed as in the case of reactive agents, but by means of predictive models they are able to act in a proactive, goal-oriented mode. Predictive agents also incorporate reactive control to achieve a more robust behavior. In particular, predictive agents are able to perform tasks which involve phenomena happening faster than the action-perception loop of the system. This type of agents, endowed with the capabilities of creating, selecting and evaluating models has the potential to be a truly general learner and also carries the potential to improve its own learning mechanism by modelling the learning itself (Thórisson, 2020b).

Reflective agents Reflective agents go a step beyond predictive architectures by enabling the agent to modify its own architecture (thus exhibiting cognitive growth) through introspection and meta-reasoning (Thórisson, 2020b). Two promiment examples of reflective agents are the Non-Axiomatic Reasoning System (Wang, 2004) and the Auto-catalytic Endogenous Reflective Architecture (Nivel, Thórisson, et al., 2013), both of which aspire to be *generally intelligent systems.*

1.2 What is a Task Theory?

As Thórisson, Bieger, et al. (2016) relate, tasks are of primary importance for Artificial Intelligence research. AI systems are built to perform tasks, some of them designed specifically to perform a single task, or a restricted set of very similar ones, while other systems are built to be able to perform a large set of tasks unknown during their design, or even any task.

Not only are tasks at center of stage during the design of AI systems, but they are also prominent during training and evaluation of AI. But despite their importance, and unlike other engineering fields, there is no general theory about the properties of tasks. In any other field of engineering it is possible to tune tasks for the effective evaluation of artifacts, use task parameters to guide their design and compose test batteries for thorough evaluation of their capabilities from their usual environment to their technical boundaries, among other things. On the other hand, in the field of AI, the lack of a task theory resulted in using extensive domain knowledge to guide the design of narrow-AI systems and the use of results from human psychology for evaluation, in this latter case with very underwhelming results. Furthermore, for the development, training and evaluation of generally intelligent systems, domain knowledge and tests like e.g. the Turing test or IQ tests don't nearly cover the breadth of situations these systems would be facing and a task theory that can model a broad range of tasks and environment becomes absolutely necessary (Thórisson, Bieger, et al., 2016).

The three main aspects where a task theory would be most useful are, as previously mentioned, evaluation, training (also including pedagogy) and design. The **Evaluation** of AI systems, in general, provides measures of progress during the development of the system and

[2]A system of this kind would be unable to hit a fastball in baseball; human brains typically employ a prediction mechanism to do that and excellent players still only hit a ball about 30% of the time.

a way to highlight strength and weaknesses. For AGI, evaluation is particularly difficult, for the chief reasons that AGI systems might be very different from each other and that their evaluation should provide some general measure of their cognitive abilities instead of focusing on performance on a particular task. A task theory would enable the evaluation of different systems, at different stages of development and on different tasks, by allowing the comparison of tasks on the relevant axes. This comparison can come about by relating the task's parameters to some physical or conceptual attributes, which can include determinism, ergodicity, continuity, asynchronicity, dynamism, observability, controllability, periodicity and repeatability. Furthermore, a task theory would make the construction of new task-environments and variations thereof easier, allowing the construction of tasks for the specialized evaluation of certain aspects of the system under test. Task-environments can also be composed or decomposed, and scaled up or down in complexity if allowed by a task theory. **Training** AI systems need to be trained for the tasks they are to carry out, and a task theory would be useful for the creation of training task-environments. Moreover, a task theory would be useful for pedagogy. Besides training environments, learning of a particular task can be sped up by teaching, in the form of providing additional information or by showing how to perform the task or a subtask. A task theory would equip researches with a better understanding of tasks and allow teaching by means of analogies and abstraction. The **design** of AI systems today is mostly a matter of trial-and-error, intuition and domain knowledge. A task theory would help and speed up the design of narrow-AI systems by allowing the prediction time, energy or other resources requirements for tasks. A task theory would also come, as already mentioned, with the ability to compare and describe properties of tasks, which can of course decrease the amount of uncertainty when designing these systems (Thórisson, Bieger, et al., 2016).

Summing up the previous discussion, in (Thórisson, Bieger, et al., 2016) the authors lay out the requirements for a task theory:

1. *Comparison* of similar and dissimilar tasks.
2. *Abstraction* and *concretization* of (composite) tasks and task elements.
3. Estimation of time, energy, cost of errors, and other resource requirements (and yields) for *task completion.*
4. Characterization of task complexity in terms of (emergent) quantitative measures like *observability*, *feedback latency*, form and nature of *information/instruction* provided to a performer, etc.
5. Decomposition of tasks into subtasks and their atomic elements.
6. Construction of new tasks based on combination, variation and specifications.

A task theory fulfilling these requirements would allow the development of frameworks that can construct task models following the theory, produce variants of tasks and execute tests in batch mode providing huge amounts of data for the AI system being tested. The inclusion of energy, time and other resources grounds the theory in physical reality and allows the performance evaluation of AI using energy as a function of time. Furthermore the precision of the task goals achieved can be compared against the limits imposed by the laws of nature (Thórisson, Bieger, et al., 2016).

One possible way to go to develop a task theory, which was the starting point for this thesis, is to consider the ratio between the solution space and the space of all possible actions. The intuition is that the smaller is the solution space compared to the action space, the harder the task is for any particular agent (and especially for an agent that acts randomly). This ratio can be measured in a number of additional dimensions, including time and energy, and it would allow the comparison of tasks by placing them on a multidimensional space with the measured dimensions as axes. The inclusion of time, energy and possibly other resources as dimensions would also help in estimating their usage for task completion (Thórisson, Bieger, et al., 2016).

1.3 Historical notes

Attempts to provide objective measures for task analysis was started around the middle of the 20th century, along with the field of human factors (ergonomics). Drury (1983) describes three forms of task analysis which trace their origins to military and industrial applications of that period: *Sequential* task analysis describes sequences tasks as rigid patterns with a minimal number of choice points (Miller, 1953); *Branching* task analysis where the sequence of tasks is determined by the outcomes of specific 'choice' tasks (Kurke, 1961); and *Process control* task analysis which considers a human operator in control of many variables using a specific strategy to monitor, sample and control them (Beishon, 1967).

Therefore the first attempts towards a task theory based on a cognitive analysis were also made with humans as the primary focus. In 1983, Stuart K. Card, Thomas P. Moran et al. published "The psychology of human-computer interaction," which was concerned with the interaction between humans and interactive computer systems. In this book they introduced the human processor model (MHP - Model Human Processor) and the GOMS (Goals, Operators, Methods, Selection rules) model. The MHP is mostly concerned with calculating how long it would take a human to perform a task, by taking into account experimental times for cognitive and motor processing as well as the working memory and long-term memory storages. The GOMS model is used to evaluate the usability of a computer system by means of quantitative and qualitative predictions of how humans would use it. A GOMS model is composed of four main parts: Goals, which define a state to be reached; Operators, which are the atomic actions on the perceptual, motor or cognitive level, and are the way with which an agent can interact with the environment; Methods, which describes a procedure to achieve a goal; and Selection Rules to select a specific method when multiple of them are available (Card, Newell, and Moran, 1983). The issue with both MHP and GOMS is that it is not a general theory at all, but is confined to the context of human-computer interaction, and that it is only concerned with user performance, rather than the tasks themselves.

Similarly, Cognitive Task Analysis (CTA) and Applied Cognitive Task Analysis (ACTA) are focused on gathering data about people performing cognitively demanding tasks. CTA uses interviews and observation strategies to capture expert knowledge, cognitive processes and decisions from domain experts (Crandall, Klein, and Hoffman, 2006; Clark et al., 2008). The outcome is typically in the form of performance objectives, equipment, procedural and conceptual knowledge and standards of performance (Crandall, Klein, and Hoffman, 2006; Clark et al., 2008). ACTA improves on CTA techniques, which are often resource intensive, to be of more practical use to system and instructional designers rather than scientists by streamlining CTA processes (Militello and Hutton, 1998).

In the field of AI proper, work on task theory has mostly consisted in frameworks for the generation of tasks. In 2014, Garrett, Bieger and Thórisson published a paper about Merlin (Multi-objective Environments for Reinforcement Learning), a tool for the automatic generation and tuning of Markov Decision Problems (MDPs), with the aim to evaluate multi-objective learning capabilities in reinforcement learners. The tool supported discrete and continuous MDPs, the specification of an arbitrary number and type of goals and it could be used for spatial navigation problems (e.g. mazes). In Merlin tasks could be constructed by affecting independently the state space, the action space the probabilities associated with state transitions given some action was executed and the reward signals. In particular, the degree of pair-wise positive or negative association between multiple tasks in the same problem can be fully customized for the generated MDP (Garrett, Bieger, and Thórisson, 2014).

Next came FraMoTEC (Modular Task-Environment Construction Framework). FraMoTEC is a more general tool that allows the construction and simulation of tasks in a modular way, using so-called building blocks. The building blocks include objects, which have physical properties such as position, mass, velocity, etc.; transitions which are functions evolving the state of objects; motors (i.e. actuators) and sensors which allow the controller to interact with the environment by spending energy; systems which act as containers of other building blocks; and finally goals, which define states to be reached with tolerance values and time constraints (Thorarensen et al., 2016).

The latest framework that has been developed is SAGE (Simulator for Autonomy & Generality Evaluation Framework). SAGE can be considered to be an evolution of FraMoTEC with the advantage that is general enough that it can be used for evaluation of both narrow and AGI-aspiring systems, while FraMoTEC was specifically catered to general learners (Eberding, Sheikhlar, and Thórisson, 2020).

On the analysis of tasks not much has been done since the 80s, but an approach relying on expert knowledge was published in 2018 by Bieger and Thórisson. Such approach requires a domain expert to explain the actions to be performed to carry out some task, with an interviewer following the typical requirements engineering workflow. The sequence of actions obtained at this step should allow the construction of a dependency graph and the interviewer should make sure that the actions are simple enough that cannot be further broken down in more sub-actions. Next comes the step of characterizing the actions in terms of their inputs, outputs and how the transformation of the former into the latter is carried out. Moreover, a way to assess the success or failure of an action should be specified. The task to be performed can then be decomposed in simpler components, either in the terms of the actions to be executed (Task-based decomposition), the variables that are part of the task (Feature-based decomposition) or by some specific functionality that is a sub-part of the task (Functionality-based decomposition). After this step, all that is left to do is to construct an appropriate teaching curriculum for the specific learner that is to perform the task (Bieger and Thórisson, 2018).

CHAPTER 2

Background

In this chapter we introduce the necessary notions on which our task theory rests on. In Section 2.1 we clarify what we mean by *model* and in particular what is meant by *causal-relational bi-directional model*, while in Section 2.2 Thórisson's theory of pragmatic understanding is introduced, which makes use of these models to define what it means to *understand* a phenomenon and how to characterize *meaning*. In Section 2.3 the concept of causality is discussed, introducing the work of Judea Pearl about causal diagrams and then moving onto the 'manipulative' approach to causality. Towards the end of the section, the mini-Turing test devised by Pearl is presented, together with my attempt to administer it to NARS.

2.1 Bi-directional models

A model is an information structure that is used as a representation of the thing being modelled. A model should resemble in some way the subject of the modelling, usually abridged of unimportant details and structured in such a way that it allows various manipulations on it for the purpose of making queries and obtaining answers about the thing it models. The particular type of manipulations that can be performed depends on the specific model, and different models of the same thing can allow different types of manipulations. A model by itself is useless without an appropriate process that makes use of it: the particular type of process with its limitations define what can be done with some model. A typical example of model would be a computational model, which can be used to answer what-if questions through the process of performing simulations. Models are characterized not only by the form of representation, but also by their comprehensiveness and level of detail, which determine what use can be made of the model (Thórisson, 2020d).

Models encode actionable information, in the sense that they can be used to get things done, like predicting future states, derive the causes of observed events, explain observed phenomena and, obviously, act as re-creation of the thing being modelled. The process by which a set of models allow an intelligent agent to perform these things can be considered as a sequence of steps: first find the relevant models, then apply them to derive predictions, actuate the actions based on the predictions and lastly monitor the outcomes of said actions. Incorrect models that don't lead to the expected goals are pruned and modified so that over time only correct models remain in memory, in this sense it can be said that they encode non-axiomatic and defeasible knowledge (Thórisson, 2020d).

A model is, by definition, a representation of something, that is, an encoding of some data or measurements of the physical world. Representation is fundamental, because all knowledge used for intelligent action must contain some sort of representation of the things it refers to. Given the fact that all the information in the world is much greater than the ability of any system to store all of it, effective information storage in the form of appropriate models is one of the fundamental building blocks of any theory of intelligence (Thórisson, 2020d). In fact, an even more general result applies, that can be summarized as "Every good regulator[1] of a system must be a model of that system" (Conant and Ashby, 1970). Conant and Ashby proved, in what they called the "Good regulator theorem", that any optimal regulator must in some way behave as a mirror image of the system it regulates, i.e. there must be some mapping from the states of the system to the states of the regulator, and such mapping makes the regulator a model of the system (Conant and Ashby, 1970). The implication of this result from control theory is that an intelligent learner must proceed by modelling the physical world to appropriately 'control' it, i.e. be an efficient survivor[2] in the complex environment that real life experience presents (Thórisson and Talbot, 2018b).

The kind of models that this thesis talks about are *bi-directional*, in the sense that can be executed in forward-chaining to produce predictions and in backward-chaining to achieve goals, by chaining back a path of states until a performable action is reached (Thórisson and Talbot, 2018a; Thórisson and Talbot, 2018b). These models are also *causal-relational models*, in the sense that they are an executable information structure that encodes procedural (causal) knowledge, where the left-hand side (LHS) is the cause and the right-hand side (RHS) is the effect. The LHS is the 'input' of the model, and represents a pre-conditional pattern composed of variables, values and so on and it specifies, through pattern matching, in which situation a certain model is relevant. The RHS then represents the post-conditions of the LHS pattern. In forward-chaining, when the LHS pattern is observed, a prediction based on the RHS is generated by a process of deduction. In backward-chaining, when the RHS pattern is observed and it is a goal, a sub-goal based on the LHS is generated. Sub-goals can be further backward-chained until a command for some actuator is produced, and in this way models can be used to produce effective plans to achieve goals. If the RHS is not a goal, backward-chaining can be used to derive potential causes for the observed state. Model also incorporate a number that specifies how many time the model was correct out of all the times it was applied, this value gives an idea of the accuracy of the model, and to discriminate between multiple models that could be used in the same situation. To transform the RHS into the LHS or viceversa, models incorporate two sets of (learned) functions that actually carry out the transformation, i.e. functions that model the actual physical mechanisms that in the real world act between the actually existing physical entities (Thórisson and Talbot, 2018a; Thórisson and Talbot, 2018b).

Furthermore, such bi-directional models can refer to other models in order to form a hierarchy, making it possible to represent compound phenomena and, in particular, to be used by an intelligent agent to model itself in a reflexive manner. A model M_a may be featured on the LHS of another model M_b, in such case specifying a post-condition of the successful the execution of M_a by predicting its outcome. Viceversa, if M_a features on the RHS of M_b it specifies a positive pre-condition of M_a, predicting the successful execution of M_a when the premise of M_b is observed. Models can be built in conjunctive form, in which case they specify a causal relationship whose effect is brought about by a certain context made up of multiple, temporally correlated, required pre-conditions. Models built in disjunctive form, on

[1]What is meant in control theory by 'regulator' is equivalent to the AI notion of 'controller'

[2]Intelligence can be seen as a survival tool to cope with the insufficient knowledge and resources that characterize life in the real world.

the other hand, specify a causal relationship where the effect is brought about by the occurrence of the most likely pre-condition: positive pre-conditions are a set of options that entail the success of the model (Nivel, Thórisson, Steunebrink, and Schmidhuber, 2015).

The simplest terms that can serve as inputs for the LHS or RHS of a model are called facts, and include a payload, which is the observed event, a likelihood value indicating the reliability of the observation and a time interval specifying the period within which the fact is believed to be true. Therefore facts are valid only within a certain time interval, and thus the execution of models is time-dependant. The ground-truth of facts is valid to the degree specified by the likelihood value and only within the specified time-interval (Nivel, Thórisson, Steunebrink, Dindo, et al., 2014).

2.2 A Theory of Understanding

In the context of this thesis the concept of understanding that is used comes from the theory of *pragmatic understanding* laid out in (Thórisson, Kremelberg, et al., 2016) and (Bieger and Thórisson, 2017). The pragmatism of this theory lies in its focus on the practical usefulness of a certain level of understanding in guiding behavior to achieve goals and perform tasks (Thórisson, Kremelberg, et al., 2016). For the purposes of anchoring our proposed concept of "intricacy" (see Section 3.3), we provide a detailed summary of this theory here.

Before defining understanding, it is necessary to introduce the concept of phenomenon. A phenomenon $\Phi \subset W$ where W is the world is composed of a set of elements $\{\varphi_1, \varphi_2, ..., \varphi_n \in \Phi\}$ of various types, including relations $\mathfrak{R}_\Phi$ that bind elements of Φ with each other and with elements of other phenomena. One important type of relations are causal relationships, but other types can be considered too, for example mereological relationships. These relations can be partitioned in two sets, the set of *inward facing* relations $\mathfrak{R}_\Phi^{in} = \mathfrak{R}_\Phi \cap (2^\Phi \times 2^\Phi)$ and the set of *outward facing* relations $\mathfrak{R}_\Phi^{out} = \mathfrak{R}_\Phi \setminus \mathfrak{R}_\Phi^{in}$. An agent understanding only $\mathfrak{R}_\Phi^{in}$ can be said to understand the phenomenon Φ but not its relation to other phenomena, while an agent understanding only $\mathfrak{R}_\Phi^{out}$ understands the phenomenon's relations to other phenomena but is unable to understand its inner workings (Thórisson, Kremelberg, et al., 2016).

An intelligent system's understanding of a phenomenon comes about by its creation of models of the phenomenon. For the purposes of this discussion, we assume that these models could be the kind of bi-directional models discussed above in Section 2.1. A set of models M_Φ for a phenomenon Φ consists in information structures that can be used to (1) predict Φ, (2) produce effective plans to achieve goals with respect to Φ, (3) explain Φ and (4) re-create Φ. The better these models represent elements $\varphi \in \Phi$ including their relationships $\mathfrak{R}_\Phi$, the greater is the accuracy of M_Φ with respect to Φ (Thórisson, Kremelberg, et al., 2016).

Therefore, considering an agent A's knowledge to be a set of models M, Thórisson, Kremelberg, et al. (2016) define understanding as:

Definition 2.2.1 (Understanding)**.** An agent's A understanding of phenomenon Φ depends on the accuracy of M with respect to Φ, M_Φ. Understanding is a (multidimensional) gradient from low to high levels, determined by the quality (correctness) of representation of two main factors in M_Φ:

U1 The completeness of the set of elements $\varphi \in \Phi$ represented by M_Φ.

U2 The accuracy of the relevant elements φ represented by M_Φ.

The understanding of a phenomenon Φ is evaluated by testing on, at least, the following four capabilities of the understander, ordered by the increasing level of understanding required to master each: prediction, goal achievement, explanation and re-creation in the context of Φ. Each of these capabilities can be evaluated in a range between zero and one as a function of **U1** and **U2** (Thórisson, Kremelberg, et al., 2016).

Prediction in the context of a phenomenon consists in inferring values of variables given the values of other variables. These predictions need not necessarily be forward in time, but may be about the current time step or even about past values. In fact, predictions don't even need to follow the causal direction and don't necessarily require an understanding of causal relationships. Predictions may be computed by associations alone, because the state of a variable can already be inferred by reasoning on the co-occurrence of the state of other variables[3] (Thórisson, Kremelberg, et al., 2016). Predictions, as understood in this theory of understanding, mirror Pearl's observational queries in his characterization of causation as a ladder, and both are the lowest level of their respective hierarchy, as they are the simplest form of intelligent behavior. As previously mentioned in the discussion about Pearl's work, from animals and humans to machine learning algorithms, most if not all intelligent systems show a high degree of proficiency in predicting events and states. In fact, even non-intelligent systems such as linear regression or Bayesian probability are good at such predictions, and it may be argued that intelligence is not a requirement at all for building good predictors. The predictive ability of a system can be evaluated by giving the system a set of variables with their respective values, along with the time at which those values were sampled, and then asking different types of questions. A type of question might show the system another set of variables with their values and a certain time t, and would correspond to asking the system whether the variables shown will take the given values at the time t. Another type of question might omit the values of the variables and the query would be asking the system what values would those variables jointly have at time t. In a similar way, omitting the time would result in asking at what time would the variables obtain their values, if any such thing is possible at all. Even the variables can be omitted, and it would correspond to asking which variables can take the given values at the given time t. Furthermore, questions could be formulated by omitting a combination of the variables, values and possibly the time, or instead of giving precise values a range of values could be used as a way to partially omit values. The answers to any of these queries might not be unique, and it could be expected of the system to produce all of them, possibly with a confidence value for each of them (Bieger and Thórisson, 2017).

Achieving goals in the context of a phenomenon is a further step above predictions, because it necessarily requires an understanding of causal relations, in particular of the causal relations that relate variables under the intelligent system's control (the manipulatable variables) to its goals. For this end, models that capture the interaction of the system with the world become absolutely necessary (Thórisson, Kremelberg, et al., 2016). To evaluate goal achievement the system is assigned a task that is related to the phenomenon in some way. By task, or rather task-environment, is meant an activity with an initial state (the variables' values at the initial time step) and a goal which is the target set of values for a subset of the variables of the task. A task might possibly also have a set of constraints that if violated result in instant failure, an allotted energy and time requirement for the system to conclude the task (a must for tasks in the physical world) and some additional information. Since the body of the system is part of the task-environment, some of the variables of a task are manipulatable (otherwise the system cannot possibly do anything), while some others might only be observable (through the sensors on the body of the system). Additionally, at least a subset of the manipulatable variables must in some way be causally related to the goal variables. The ability of a system to achieve its goals can be evaluated by testing its ability to produce effective plans for executing the assigned task, where by effective plan is meant that the plan can be proven to be

[3]It is not excluded that understanding of causal relations might indeed help anyway.

useful, efficient, effective, and correct, through implementation. By evaluating the production of plans instead of goal achievement directly, the issue of dealing with task-specific interfaces (i.e. a particular body for the system) is avoided (Bieger and Thórisson, 2017). The achievement of goals can be said to correspond to the ability to answer interventional questions in the second rung of Pearl's ladder of causation, as in both cases the intelligent agent is required to understand the consequences of their (or someone else's) actions in the world, and for that causal relations have primary importance. The passive observation of the world and its associations, which suffices for prediction/observational queries, is inadequate for reaching a task's goal state, or to correctly predict the effects of actions on variables.

Explanation of phenomena is a further step in demonstrating understanding. While even goals in some cases might be achieved through some carefully executed tricks without using models, explanations require a correct causal model of the phenomenon's inner and outer relations. Explanation of a phenomenon consists in finding the necessary and sufficient causes for the event the intelligent system is being asked about; furthermore the system must be able to highlight what causes are salient, that is, which variables are the most prominent among all the identified causes (cf. actual causes by Pearl) (Thórisson, Kremelberg, et al., 2016). To evaluate an intelligent system's explanation capabilities, it would be useful to ask explanations at different levels of abstraction, from the lowest to the highest level of detail. Another way would be to introduce novel modifications to the phenomenon and see whether the system is still able to produce meaningful explanations. This makes not only possible to verify the accuracy and completeness of the understander's causal models, but also a way to find what are their boundaries. As of today, practically no AI system is capable of providing any such explanation, neither in natural language or in any other lower level machine language (Bieger and Thórisson, 2017).

Lastly, the **(re)creation** of a phenomenon is the strongest form of evidence of an intelligent system's understanding of it. Creating or re-creating a phenomenon involves the production of models that expose its necessary and sufficient features, in the same that scientists today are able to produce models of the universe or of the human body (Thórisson, Kremelberg, et al., 2016). This ability can be evaluated by giving a system some of type of materials or components and then asking it to combine them to recreate the phenomenon being tested, or at least to imitate it in the best possible way. As for explanation, there are virtually no AI-systems today able to produce models that can be understood and interpreted by us (Bieger and Thórisson, 2017).

Parallels can be traced between explanation and (re)creation of phenomena with Pearl's third rung of the ladder of causation, counterfactual reasoning. Counterfactual reasoning allows an intelligent system to reason about what could have happened had something occurred differently. It requires a model of the world, or of a phenomenon, that is accurate enough to make the agent able to find out the reason why things turned out the way they did. It is therefore a model precise enough to be in fact a re-creation of the phenomenon at hand. Furthermore by having such a model the agent can compute the sufficient and necessary causes of events and sub-parts of the phenomenon, and would therefore also show a considerable skill in providing human understandable explanations.

Having introduced this theory of pragmatic understanding, now it is possible to discuss the concept of *meaning*. A causal event x acquires meaning for some agent A when x has potential to influence something of relevance to one or more of the agent's goals G. Given some event x that might be relevant for A, the agent can compute its meaning with respect to any of its goals in the current situation, where the current situation consists in the actual values of some subset of the world's variables. The consequence of also considering the current situation when assigning some meaning to an event is that meaning depends on the context, and thus also on time. Therefore, for any agent, something holds any meaning insofar as it is related to some time-constrained goal. The meaning of any given datum (where a datum could be an event, a perception from a sensor, or even the result of an internal reasoning process) consists in its set of implications in the current (time-dependent) context for some active goal of the agent. An agent should seek to compute the implications that are relevant for its goals, giving precedence to those that are more temporally salient. An intuitive way to characterize what data might provide relevant implications for a goal is to take the point of view of causal diagrams. If a datum is generated from a variable that is part of a causal path that leads to the goal variable, then there's good reason to believe that it might provide the agent with relevant implications (Thórisson, Kremelberg, et al., 2016).

2.3 Causality

Having described *causal-relational models* and *understanding* in the previous sections, we can now introduce Pearl's (Pearl, 1988; Pearl, 2009; Pearl and Bareinboim, 2014) approach to causality. The most important type of relationships that a learner needs to understand to execute a task are the causal relationships that relate its actions and observations to the task's goals. The objective of this section then is to clarify what are causal relationships, why are they needed and how they can be formalized in a mathematical language. The concept of causal diagram is also presented as it will be the basis for the representation of tasks in Chapter 3. The contents and level of detail of this section is higher than what is strictly necessary for the subsequent discussion but we included it here anyway for the benefit of future readers who might find useful to see a compact overview of some of Pearl's work on causation.

Pearl and Mackenzie's account of causality is introduced by the useful metaphor of a three step ladder corresponding to to three levels of causal reasoning: association, intervention and counterfactuals (Pearl and Mackenzie, 2018, pp. 27-43). These levels describe a progression in cognitive abilities from the lowest level of reasoning by association to the highest level of counterfactual reasoning.

The first rung of the ladder is concerned with predictions based on passive observations of the world. Reasoning is performed by looking at associations in the data, that is, by looking at whether the occurrence of some event results in a greater (or lesser) likelihood to observe another event. Measures of association, of which correlation is an example, cannot discriminate causes from effect and neither can tell if any causal relationship is present at all; nevertheless good predictions are possible even without causal knowledge. The types of questions that exemplify this level are concerned with estimating values of variables given the observation of other variables. Besides animals, Pearl and Mackenzie also claim that most learning machines are also stuck at this level of cognition. Machine learning and deep learning algorithms, as cleverly designed and finely tuned for their tasks might be, only work with pure data and do not incorporate causal knowledge, and therefore are limited to work by association of observations (Pearl and Mackenzie, 2018).

At the second step of the ladder the concern is shifted from the observation of the world to the active intervention in it. An agent that is able to reason about interventions can predict the consequences of its and other agents' actions and is able to come up with plans to bring about desirable states. In order to do so, data is not enough and the agent needs a (causal) model of the world. Having such models essentially enables the agent to act in a goal-driven manner by backwards chaining from some goal state all the way back to its range of possible actions. The authors put babies and early hominids as examples of organisms that perform this type of goal-driven reasoning (Pearl and Mackenzie, 2018).

The third and last step is about counterfactuals, reasoning about past states that never were but could have been. This type of reasoning allows an agent to reflect on its past actions and learn to do better a future time, or even to learn from the experience of others. This type of reasoning also allows to find the reason why things turned out the way they did, or why they did not. According to Pearl and Mackenzie only modern humans are capable of this, and the break with early hominids who could not that happened sometime about 40,000 years ago marked a change in human cognitive abilities, which eventually resulted in the series of technological advancements that brought about modern civilization. Endowing an agent with this capability requires a model that encompasses the underlying causal processes, in other words, knowledge of the laws that regulate the world the agent lives in. Having such a theory of the world allows reasoning about not only about situations that could have been, but also about total impossibilities. It is precisely this capacity of thinking about things that never existed that have driven the human development of philosophy, science and technology (Pearl and Mackenzie, 2018).

Following this introduction, we can introduce the definition of structural causal model.

Definition 2.3.1 (Causal Model, Structural Causal Model (Pearl, 2009, p. 203)**).** A causal model is a triple:

$$M = \langle U, V, F \rangle$$

where:

i) U is a set of background variables that are determined by factors outside the model;

ii) V is a set $\{v_1, v_2, ..., v_n\}$ of variables, called endogenous, that are determined by variables in the model, i.e. $U \cup V$;

iii) F is a set of functions $\{f_1, f_2, ..., f_n\}$ such that each f_i is a mapping from (the respective domains of) $U_i \cup PA_i$ to V_i, where $U_i \subseteq U$ and $PA_i \subseteq V \setminus V_i$ and the entire set F forms a mapping from U to V. Or equivalently, each f_i in:

$$v_i = f_i(PA_i, U_i), \qquad i = 1, ..., n \tag{2.1}$$

assigns a value to v_i that depends on the values of a select set of variables in $U \cup V$, and the entire set F has a unique solution $V(u)$.

Each equation in a structural causal model has two peculiar characteristics: it represents an *autonomous mechanism* (Pearl, 2009, p. 27) and the equality sign is asymmetrical (Pearl, 2009, pp. 159-162). By autonomous mechanism is meant that each equation is not affected by changes in other equations, and therefore an intervention that targets one variable v_i leaves all other equations in place for any v_j with $j \neq i$. The equality sign is asymmetric in the sense that when dealing with interventions the equations cannot be reversed to determine any other variable than v_i. For example, if the structural causal model includes the equation $v_i = 5 \cdot v_j + u$, then the equation $v_j = \frac{v_i - u}{5}$ does not necessarily hold. Both equations hold if they are relating observations of variables, but when dealing with interventions the latter cannot be used to determine the value of v_j. Intuitively, if the value of v_i is determined by v_j because it is featured on the right hand side of the structural equation of v_i, then a change in v_j influences the value of v_i, but the opposite does not hold unless, in fact, v_i is also present on the right hand side of the structural equation of v_j too. Therefore the equality sign in structural equations behave as in standard algebra only when dealing with observations, because then the equations just express the observed relationships between variables in the model. This different interpretation of equalities is further clarified by the operational definition of structural equations given by (Pearl, 2009, p. 160).

Definition 2.3.2 (Structural Equations (Pearl, 2009, p. 160)**).** An equation $y = f(x) + \epsilon$ is said to be structural if it is to be interpreted as follows: in an ideal experiment where the value of X is set to x and any other set Z of variables (not containing either X or Y) is set to some value z, the value y of Y is given by $f(x) + \epsilon$, where ϵ is not influenced by either x or z.

The consequence of this definition is that it is only concerned with the value of y: nowhere it states anything about what values x or ϵ can take when controlling for Y.

2.3.1 Causal diagrams

In this section we provide the necessary graph-theoretic terminology that will be used in the rest of this thesis, and then we introduce the concept of causal diagram.

A **graph** is a pair $G = (V, E)$ that consists in a set of vertices, also called nodes, V and a set[4] of unordered pairs of vertices, called edges, $E \subseteq V \times V$. A graph is **directed** if the set of edges E consists in ordered pairs of vertices $(i, j) \in E$, where the first index marks the **source node** and the second index marks the **destination node**, and each edge is rendered as $i \rightarrow j$. Edges of this type can be properly called directed edges or arrows.

Two nodes i and j are considered **adjacent** if either $(i, j) \in E$ or $(j, i) \in E$, and a graph G is **fully connected** if all nodes are adjacent with each other. The **in-degree** of a node is the number of incoming directed edges, while the **out-degree** of a node is the number of outgoing directed edges. A node i is a **parent** of a node j if $i, j \in E$ but $j, i \notin E$; in such case j is also called a **child** of node i. The set of parents of a node j and the set of children of a node i are denoted $\mathbf{PA}_j$ and $\mathbf{CH}_i$ respectively. A **path** is a sequence of edges joining a sequence of adjacent nodes, with all edges and nodes distinct. A **directed path** is a path such that the destination node of each edge in the path is the source node of the following edge in the path. If there exists a directed path from any two nodes i and j, i is called an **ancestor** of j and j is a **descendant** of i.

For the purpose of this thesis we are mostly interested in directed acyclic graphs. A **directed acyclic graph**, or DAG for short, is a directed graph with no directed cycles, that is, for any two nodes i and j, either there is a directed path from i to j, from j to i, or neither of the two. Consequently, if $(i, j) \in E$ then $(j, i) \notin E$.

Causal diagram Any structural causal model has an associated graph where each vertex corresponds to a variable v_i and a directed edge is drawn from each $pa \in PA_i$ to v_i (Peters, Janzing, and Schölkopf, 2017). In other words, a directed edge is drawn from any variables occurring on the right hand side of each equation (2.1) to the vertex occurring on the left hand side. The resulting graph is assumed to be a directed acyclic graph.

It is now possible to introduce Pearl's definition of **d-separation** (Pearl, 1988). The d-separation criterion applied to any two disjoint sets of variables X and Y allows the identification of another disjoint set of variables Z which makes X and Y independent of each other when Z is controlled for. For the application of the d-Separation criterion the variables need to be represented as nodes of a directed acyclic graph (DAG) whose arrows, correctly represent their causal relationships(Pearl, 2009, p. 16).

Definition 2.3.3 (d-Separation (Pearl, 2009, pp. 16-17)**).** A path p is said to be d-separated (or blocked) by a set of nodes Z if and only if at least one of the following is true:

[4]Therefore loops are not allowed.

1. p contains a chain $i \rightarrow m \rightarrow j$ or a fork $i \leftarrow m \rightarrow j$ and $m \in Z$;
2. p contains a collider $i \rightarrow m \leftarrow j$ such that $m \notin Z$ and neither does any of his descendants.

A set of nodes Z d-separates X from Y if and only if Z blocks all paths from any node in X to any node in Y.

The concept of d-separation can be better clarified when giving causal meaning to the arrows of the graph.

A **chain** $i \rightarrow m \rightarrow j$ is a typical case of mediation, where the effect of i on j comes forth by the influence of the mediating variable m. An example of such a chain is

coffee consumption → caffeine intake → wakefulness

where the effect of the consumption of coffee on the wakefulness of an individual is mediated by the actual caffeine intake. The variable caffeine intake d-separates coffee consumption and wakefulness, therefore adjusting for it would make the other two variables independent of each other. It can be imagined that when only looking at data for a certain value of caffeine intake, the consumption of coffee need not be correlated with wakefulness at all; sometimes people get their caffeine kick from tea, or they just drink lots of decaf.

A **fork** $i \leftarrow m \rightarrow j$ can be described as the case of a common cause m causing both i and j. In this case m can be properly called a confounder of the other two variables, because i and j would appear correlated even though there's no causal relationship between the two. An example of a fork is

ice cream sales ← weather → crime rate

where a warmer weather, e.g. in summer, increases both ice cream sales and the crime rate, which appear to be associated despite having no causal link. This association disappears (the relationship is deconfounded) when controlling for the weather and adding weather to Z d-separates ice cream sales and the crime rate.

A **collider** $i \rightarrow m \leftarrow j$ represents a situation where a variable of interest m is influenced by two different causes i and j. An example is described by Sackett (1979) of a purpoted association between locomotor disease and respiratory disease in hospitalized patients. The collider can be visualized as

locomotor disease → hospitalization ← respiratory disease

Examining only hospitalized patients is equivalent of conditioning on the hospitalization variable, therefore locomotor disease and respiratory disease appear to be correlated even though in the general population (i.e. when not conditioning on hospitalization) they are not. This phenomenon is known as selection bias or Berkson's paradox(Berkson, 1946). A similar phenomenon is observed in the following example presented by Kim and Pearl (1983)

earthquake → alarm ← attempted burglary

in this case conditioning only on the event where the alarm goes off, a negative association is observed between the earthquake and attempted burglary variables: usually either of the two is enough for the alarm to ring and it is very unlikely for the two to occur together. This phenomenon is known as the "explain away" effect (Kim and Pearl, 1983), because the occurrence of one event explains away the absence of the other event. In both the previous examples the variables on the edges are d-separated only when the middle variable is not added to Z.

2.3.2 Representing time in causal diagrams

Causal diagrams can be extended to model time series data. Reasoning about causation on variables that refer to different moments in time is actually easier than timeless data, because causation can occur only forward in time (Peters, Janzing, and Schölkopf, 2017).

Causal diagrams can now include an infinite number of nodes as they stretch indefinitely in the future. The nodes of the causal diagram are denoted by X_t^j where $j \in \{1, \dots, d\}$ is the index denoting a variable in the d-dimensional vector $\mathbf{X_t}$ and $t \in \mathbb{Z}$ is the temporal index of the variable. Since causation cannot occur from the future to the past, there are only two types of causal relations to describe: from the past to future and from the same time step (Peters, Janzing, and Schölkopf, 2017).

A causal diagram containing arrows only from X_t^j to X_s^k for $t < s$ and not for $t = s$ is said to be without instantaneous effects, as causal links only occur from a past time step t to a more recent time s. On the other hand, a causal diagram which also contains arrows from X_t^j to X_t^k for some j and k is said to feature instantaneous effects. A causal diagram is said to be purely instantaneous if it contains arrows from variables X_t^j to X_t^k and X_s^j but not X_s^k, for $j \neq k$ and $t < s$ (Peters, Janzing, and Schölkopf, 2017).

Furthermore, we can distinguish the full time graph from the summary graph. The full time graph is a complete representation of the causal diagram with an infinite number of nodes X_t^i. The summary graph is a more compact representation, with nodes X^i for $i \in \{1, \dots, d\}$ and arrows from X^j to X^k for $j \neq k$ whenever in the full time graph there is an arrow from X_t^j to X_s^k for $t \leq s$. While the summary graph might contains cycles, the full time graph is assumed to be acyclic (Peters, Janzing, and Schölkopf, 2017).

The structural causal model associated with the full time graph can include at most the past q values of all variables in a given structural assignment. The structural assignments are of the form:

$$X_t^j = f^j \left((PA_q^j)_{t-q}, (PA_{q-1}^j)_{t-q+1}, \dots, (PA_0^j)_t, U_t^j \right)$$

which means that a given variable X^j at time t can be affected by the set of its parents from time t to time $t - q$ for some fixed q and the set of endogenous variables U^j at time t. Of course, PA_{t-s}^j does not contain X_{t-s}^j for $s = 0$, but it may for any $s > 0$ (Peters, Janzing, and Schölkopf, 2017).

2.3.3 Association and correlation

Every student of statistics knows that correlation does not imply causation, but, as Pearl and Mackenzie put it, "some correlations do imply causation"(Pearl and Mackenzie, 2018, p. 77).

The concept of correlation can be traced back to Sir Francis Galton in 1888, when he compared forearm and height measurements and realized that the predictions of the value of a variable given the other were always points on the same line, the regression line(Galton, 1888). The formula for correlation was later introduced by Karl Pearson, and the slope of the regression line was called the correlation coefficient. Correlation is just a particular type of association, specifically, a linear association between variables.

The first problem when considering associations is that, intuitively, not all associations are meaningful. Compare, for example, the association that exists between ice cream sales and murders with the association that exists between clouds and rainfall. Both associations might well be very strong, but the former just seems spurious, as some associations are rightly called. The reason is that ice cream sales and crime are not causally related at all, but are associated because of an hidden common cause, a confounder (e.g. weather, season and so on). The association between clouds and rainfall, on the other hand, seems an obvious one as it is the result of the presence of causal relationship between the two, because clouds cause rainfall.

As the following section will show it is not possible to discriminate which associations are meaningful and which are not without referring to the concept of causation (Pearl and Mackenzie, 2018, p. 72).

2.3.4 Linking association with causation: the Common Cause Principle

The Common Cause Principle was introduced by Reichenbach (1956) to explain the occurrence of statistical dependencies (associations) in data as the result of (possibly unknown) causal links.

Definition 2.3.4 (**Reichenbach's Common Cause Principle** (Reichenbach, 1956))**.** Assume that two random variables A and B are not statistically indepdendent, that is $A \perp\!\!\!\perp B$. Then any combination of the following must be true:

- A causes B
- B causes A
- A third variable C is a (Reichenbachian) common cause of both A and B

In short, what the Common Cause Principle says is that for any association found in the data, there must be a causal explanation for it. While this statement might be true in most cases, there are some known exceptions. Firstly, association between variables might arise as a consequence of selection bias (see the discussion on colliders in Section 2.3.1). Second, spurious associations might arise when looking at time-series data of phenomena that are both developing over time. And finally, some associations appear solely due to random chance, due to the multiple comparisons problem. If none of these considerations apply, the Common Cause Principle provides the only possible explanation for the observed dependence (Peters, Janzing, and Schölkopf, 2017, p. 7).

2.3.5 The manipulative approach to causation

The approach to causality that is followed in this work is the so-called "manipulative approach" (Pearl, 2009, pp. 223-228). In this description of reality, the physical world is made up of independent and invariant mechanisms, each of which constrain a finite set of variables. The interaction between mechanisms is usually carried out through shared variables (i.e. a variable might be part of multiple mechanisms), therefore to predict the causal consequences of an action being performed, which can be usually understood as setting a set of variables to some value, it is only necessary to set down the relevant mechanisms and then follow the ensuing interactions between mechanisms until a new steady state is achieved. The underlying assumption of this framework is that these actions are *local surgeries*. By locality, here is meant that actions are local in the space of mechanisms (instead, e.g., of variables or time). That is, an action only alters a specific mechanism while all other mechanisms remain in place. Pearl relates here a useful example of an array of domino tiles: tipping a tile is by no means a local action, as it affects the full array, yet it is local in the sense that it only affects the mechanism regulating that one single tile, which in this case are the physical forces that maintain it erect and still. Since all other mechanisms remain in place unaltered, any other mind possessing the causal model of this example is able to compute the consequences of this action. It is precisely in this sense that causal diagrams can answer such a huge number of queries about the effects of actions without having to fully specify all the possible outcomes beforehand (Pearl, 2009).

Pearl goes on to justify the notion of causation on the grounds that, except for the context of physics and engineering, mechanisms in everyday scenarios don't usually have names. Unlike electrical circuits where one only has to input different values in systems of well known equations, real life interventions are usually in the form of propositions (like "press the button", or, using Pearl's own terms, '$do(x)$', where x can be any proposition). This specification is nonetheless sufficient if the knowledge is organized causally, because each variable is determined by only one mechanism. Therefore intelligent beings are able to find out by themselves which is the mechanism to be perturbed to carry out the specified action and subsequently derive the consequences on the other mechanisms. This linguistic abbreviation, as Pearl puts it, defines a relation that is usually called 'causation'. The definition he provides is the following:

> Event A causes B if the perturbation needed for realizing A entails the realization of B. (Pearl, 2009, p. 225)

where by 'needed' is meant that there is a condition of minimality in the perturbation realizing A. Relationships between variables and the interaction of mechanisms are most commonly expressed in terms of this type of cause-effect relationships, instead of the mechanisms themselves (Pearl, 2009).

Using this approach to causality it is necessary that for each equation of a structural causal model, representing an individual mechanism, there is a single, special variable that is featured on the left-hand side and is specified as the dependent (output) variable. In general, though, mechanisms can also be specified as a function constraint in the form of:

$$G_k(x_1, x_2, \ldots, x_l, u_1, u_2, \ldots, u_m) = 0$$

In such cases, it is possible to wonder whether given a set of such functional constraints it is still possible to find the unique dependent variable for each mechanism. To do so, under the assumption of locality of actions, Pearl relates the following procedure: let A_k to be the set of actions affecting the equation G_k, and suppose that an action $a \in A_k$ was chosen and that the changes it effected to G_k altered the solutions of the system of equations compared to before the action was taken. Then if X is the set of variables featuring in G_k, check if there is some X_k variable that is responsible for the changes in all other solutions. If X_k remains the same for all choices of actions in A_k and u, then that is the dependent variable of G_k. Therefore it can be said that the changes effected by A_k can be ascribed to the changes in X_k, therefore X_k can be considered the 'representative' variable of mechanism G_k and saying that action a caused event $Y = y$ would be tantamount of saying that event $X_k = x_k$ caused $Y = y$, because of the invariance of X_k with respect to the choice of action. Because of that it is possible to write such action as $do(X_k = x_k)$ when in fact x_k can be best understood as the causal consequence of said action, and it is given a clear meaning to the notion of locality of actions. Such procedure requires knowledge of which variables are the background factors u, otherwise there can be many possible different representative variables, depending on which variables are selected as background factors. Furthermore a dependent variable in a model might become independent in another model, even if part of the same mechanism (Pearl, 2009).

This asymmetry of structural equations representing causal relationships does not conflict the usual symmetry that characterizes physical equations. Stating that X causes Y and not the other way around means that changing the mechanism of X (the mechanism where X is the dependent variable) has a different end result than changing the mechanism of Y. Pearl

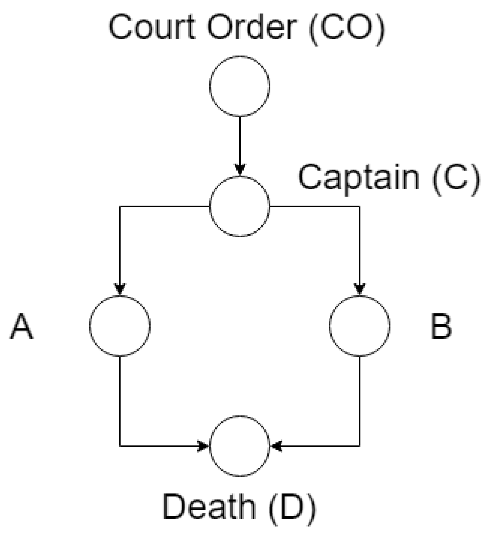


Figure 2.1: Causal diagram of the firing squad scenario.

claims that since these two mechanisms are different, the preceding causal statement is not at odds with the symmetry of equations in physics and engineering. In fact, the feasibility of ascertaining for each equation which variable is dependent means that the discovery of causal relationships is in effect equivalent to the discovery of physical laws by means of, e.g., experimentation. According to Pearl, the problem of causal induction can be thought of as equivalent to the better known problem of scientific induction (Pearl, 2009).

2.3.6 The mini-Turing test

Pearl and Mackenzie (2018, pp. 43-52) introduce the mini-Turing test as a test to verify if a machine can answer causal questions in the same way a human can. The machine would be given an appropriate encoding of a story and then asked some questions about it. It is important to note that the machine is given this data using the best possible representation for the machine and that all the information given is already correct; i.e. there's no need for the machine to learn anything more than it is given or to perform (a possibly very difficult) causal discovery to learn what relations are there between variables.

A proposed representation that machines should use to pass the test is a causal diagram as was defined in Section 2.3.1, which according to Pearl and Mackenzie is also the only model known to pass the test.

The example story that is described by Pearl and Mackenzie (2018, pp. 46-47) is about a firing squad, where a Court (CO) orders the execution, the order is relayed to a Captain (C) which in turn gives his command to Soldier A and Soldier B. Both soldiers always follow orders and always hit the target, so anyone of them shooting is enough to cause the prisoner's Death (D). This story is represented by the causal diagram shown in Figure 2.1. All the variables in the diagram can be understood as Boolean variables where a value of 1 means the event has happened (e.g. $C = 1$ means that the Captain gave his command) and a value of 0 means it has not (e.g. $A = 0$ means Soldier A didn't fire).

To illustrate this example, the firing squad story was presented to the Non-Axiomatic Reasoning System (NARS) using its native narsese language. The narsese code is shown in Listing 2.1. The causal relationships are encoded using the implication operator and by describing both the positive (when an event happens) and negative (when an event doesn't happen) cases for all the causal relationships of the diagram.

Listing 2.1: Firing squad example in narsese

```
<<(*,{CO},order) --> give> ==> <(*,{C},order) --> give>>.
<(--, <(*,{CO},order) --> give>) ==> (--, <(*,{C},order) -->
   give>)>.

<<(*,{C},order) --> give> ==> <(*,{A},{prisoner}) --> shoot>>.
<<(*,{C},order) --> give> ==> <(*,{B},{prisoner}) --> shoot>>.
<(--, <(*,{C},order) --> give>) ==> (--, <(*,{A},{prisoner})
   --> shoot>)>.
<(--, <(*,{C},order) --> give>) ==> (--, <(*,{B},{prisoner})
   --> shoot>)>.

<<(*,{A},{prisoner}) --> shoot> ==> <{prisoner} --> [dead]>>.
<<(*,{B},{prisoner}) --> shoot> ==> <{prisoner} --> [dead]>>.
<(&&, (--, <(*,{A},{prisoner}) --> shoot>), (--,
   <(*,{B},{prisoner}) --> shoot>) ) ==> <{prisoner} -->
   [alive]>>.
```

The test now entails asking different questions from the three rungs of the ladder of causation. The easiest queries are those that inquire about associations, which can answered by looking at associations alone. Examples of this types of queries are: "The Court order was given, what is the status of the prisoner?", "The prisoner is alive, was the Court order given?" and "Soldier A fired, did Soldier B fire?". These questions correspond to the following probability statements: $P(D \mid CO = 1)$, $P(CO \mid D = 0)$ and $P(B \mid A = 1)$; the conditional probabilities using data generated by the firing squad system are enough to give an answer without even using any causal model.

This type of queries, that can be answered by looking at associations in observed data, are also the only type of queries that current machine learning algorithms can hope to answer in the mini-Turing test. Machine learning systems, including deep learning neural networks and the like, work by fitting some function to the observed data, not very differently than what is also done in statistics (Pearl and Mackenzie, 2018). These AI systems do not use causal models and only receive observational data for some phenomenon, therefore are unable to answer queries about interventions or counterfactual statements.

Obviously, NARS is very well capable of answering these types of questions too, and its responses are shown in Listing 2.2, 2.3 and 2.4.

Listing 2.2: Q1: The Court order was given, what is the status of the prisoner?

```
<(*,{CO},order) --> give>. :|:
<{prisoner} --> [?1]>?
Answer <{prisoner} --> [dead]>. %1.00;0.38%
Answer <{prisoner} --> [dead]>. %1.00;0.40%
```

Listing 2.3: Q2: The prisoner is alive, was the Court order given?

```
<{prisoner} --> [alive]>. :|:
<(*,{CO},order) --> give>?
Answer <(*,{CO},order) --> give>. %0.00;0.15%
Answer <(*,{CO},order) --> give>. :-264: %0.00;0.25%
```

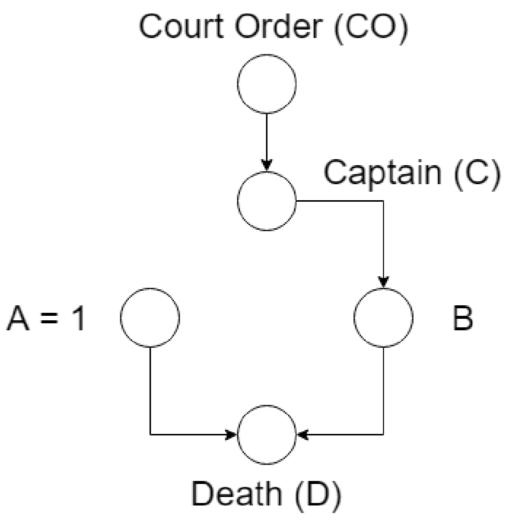


Figure 2.2: Causal diagram of the firing squad scenario after setting $A = 1$.

Listing 2.4: Q3: Soldier A fired, did Soldier B fire?

```
<(*,{A},{prisoner}) --> shoot>. :|:
<(*,{B},{prisoner}) --> shoot>?
Answer <(*,{B},{prisoner}) --> shoot>. %1.00;0.21%
Answer <(*,{B},{prisoner}) --> shoot>. %1.00;0.28%
Answer <(*,{B},{prisoner}) --> shoot>. %1.00;0.28%
Answer <(*,{B},{prisoner}) --> shoot>. %1.00;0.29%
Answer <(*,{B},{prisoner}) --> shoot>. %1.00;0.29%
Answer <(*,{B},{prisoner}) --> shoot>. :-7899: %1.00;0.56%
```

Next, the test can include questions about interventions on the causal diagram of Figure 2.1. Examples of such queries are: "If Soldier A fires on its own initiative, what will be the status of the prisoner?", "If the Captain refuses to give his command, what will soldier B do?" and "If Soldier B decides what to do on a whim, what will be the status of the prisoner?". These questions can be expressed in probability statements thanks to Pearl's do operator: $P(D \mid do(A = 1))$, $P(B \mid do(C = 0))$ and $P(D \mid do(B = \mathcal{U}\{0,1\})$ respectively. An intervention changes the underlying causal diagram by erasing all incoming arrows of the affected node(s) and by setting the value of the node's variable as specified by the do operator. For example, the intervention that sets the value of A to 1 (the intervention expressed by the first query above) results in the causal diagram shown in Figure 2.2. It can be noted that these questions are very easy to answer, but that is because humans are very good at causal reasoning (Pearl and Mackenzie, 2018). When it comes to trying to ask these questions to NARS, it turns out that there's a lack of proper terms in narsese to even pose them in the first place. That is because to ask questions about interventions, the machine needs to be able to break the rules, by blocking incoming influences on the affected variable and then setting it to some arbitrary value (Pearl and Mackenzie, 2018), which is not something that can be encoded in narsese for the time being.

One question that can be raised at this point is: "Isn't the status of the prisoner the same whether Soldier A is ordered to shoot or Soldier A just fires on its own initiative?" In other words, what is the difference between seeing and doing? Is $P(D \mid A = 1) = P(D \mid do(A = 1)$? That depends. The equality holds if there is no confounding between the variables involved. Using the firing squad example, seeing the Captain give his command and making

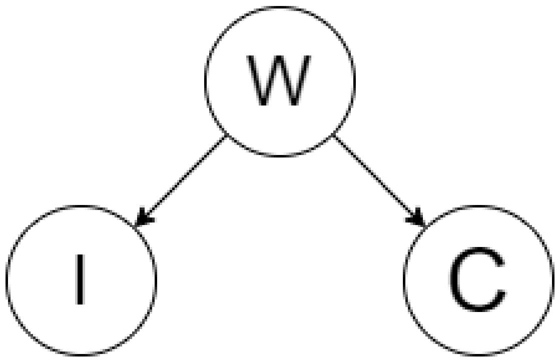


Figure 2.3: Causal diagram explaining the correlation of ice cream sales I with the crime rate C through the confounding factor W.

him give is command is the same for the prisoner's Death, because there are only causal paths from C to D and no confounding variables. This is different in the case of Soldier A, because there is a back-door path A-C-B-D, with C and B acting as confounders. Obviously, if A is observed to be one or if A is made to be one the prisoner is dead in both cases, but on the other hand, while seeing A to be zero means that the prisoner will be alive (because B is zero as well), setting A to zero does not tell that the prisoner will be alive with certainty, because possibly the Court order was given and Soldier B shot him. To clarify this discussion, consider the causal diagram in Figure 2.3. This diagram represents a famous example of confounding, where a strong association between Ice cream sales (I) and the Crime rate (C) was found. Such association can be explained, in a simplified way, by considering the Weather (W) as a common cause of both. When observing a high volume of Ice cream sales a simple application of regression can return the value of the Crime rate. But, on the other hand, setting the value of the ice cream sales volume to some arbitrary amount, e.g. 0,[5] doesn't not allow to infer anything about the Crime rate, which does not depend at all on Ice cream sales but instead is a function of the Weather. Note that even if the value of a variable is set by someone else, or even by something not human, it is still an intervention as long as the variable being set is shielded by any causal influence from other variables that would otherwise affect it, which is evident by the definition of intervention itself.

Lastly, the test can include counterfactual questions, for example: "Suppose that the prisoner has been shot, would he or she be alive had Soldier A not fired?" which corresponds to the probability $P(D \mid CO = 1, C = 1, A = 1, B = 1, D = 1, do(A = 0)$. A counterfactual question involves the observation of a world where all variables have been set ("Suppose that the prisoner has been shot,"), a question about the value of a variable ("would he or she be alive") and a statement about a different, fictional world where something happened differently ("had Soldier A not fired?"). If in the observed world the prisoner has been shot, all variable values are known by the rules of the described system: all variables are set to one. Then by means of the do operator the value of A is set to zero and now the question asks about what would the value of D be under these assumptions. It follow that Soldier A not shooting the prisoner would still result in Soldier B shooting, and therefore the prisoner's Death would still occur. This counterfactual question, just like for the intervention's case, cannot be posed to NARS because of the current impossibility in expressing the do operator in narsese.

[5]For example, a policy maker banning the sale of ice cream

CHAPTER 3

Task theory

In this chapter we introduce our work on task theory. Our main contributions towards this includes: An **extension of causal diagrams** to accommodate the various features of task-environments, including partial observability and manipulability of variables (Section 3.1); a series of **foundational principles** for moving towards the task theory we envision (Section 3.2); a **measure of a task's 'complicatedness'** that is objective and independent of any specific learner (Section 3.3) and (Section 3.4) a **measure of difficulty of a task's execution** that is dependant on both the task's intricacy and the performing controller. Sections 3.4.2 and 3.4.3 describe a series of factors that are independent of a task's intricacy and that can affect the difficulty of learning and performing a task by a controller. In the last sections we relate some ideas on how to decompose a task into sub-tasks (Section 3.5), how to compose multiple tasks into a single task (Section 3.6) and a task-theoretic point of view on the learning process of an agent (Section 3.7).

Before we proceed, a clarification is in order about the two points of view of a task, the *designer's* perspective and the *agent's* (learner's) perspective. A task can be viewed from the designer's point of view, in which case everything about the task is completely known and specified. By 'designer' we mean the agent (typically human) that decides what the task is. The 'agent' is the learner that is supposed to learn and perform the task. The designer specifies the variables, the mechanisms and the various goals and constraints, and therefore, just like a divinity, it can be assumed that whatever is being defined is an objective and incontrovertible fact about the real world. It is up to the designer, then, to ensure that their definition of the task matches the actual world they inhabit, and that whatever choices they made in the task description are valid and reasonable representations of real world entities. The other point of view is that of the agent that actually has to carry out the task. The agent has only a limited perception of the world described by the task, as it is only able to perceive the observable variables. In this case, whatever knowledge the agent may obtain through the process of learning the task has no certainty of being correct: it is in effect defeasible knowledge, because at any time, any learned fact could be proven wrong by additional experience. In the rest of this section it is always assumed the designer's point of view of the task.

3.1 An extension of structural causal diagrams

Owing to the AI background of Pearl, he envisioned the structural causal models described in the previous chapter as a useful basis for the development of generally intelligent machines capable of causal reasoning (Pearl and Mackenzie, 2018), in particular by means of a causal inference engine module (Pearl and Bareinboim, 2014, Figure 1). While AGI-aspiring systems following this approach have yet to surface, Peters, Janzing, and Schölkopf (2017) describe various applications of SCMs in the fields of machine learning and deep learning. This thesis is not concerned with the applications of causal models for the development of intelligent systems; rather, with their use in modeling tasks in a way that is independent of any specific learner. For this purpose the structural causal models and related causal diagrams described by Pearl (2009) and Peters, Janzing, and Schölkopf (2017) need to be extended in a number of ways to be useful in a task theory for artificial intelligence.

The foremost issue with causal calculus, which would come handy when trying to derive causal effects in a causal diagram, is that it requires huge amounts of data, possibly even infinite amounts, to be applied effectively. Such abundance of data is rarely the case for real world scenarios, as learners are expected to work under the assumption of insufficient knowledge and resources (AIKR) (Wang, 2012). Another issue with the approach of causal calculus is that the temporal component of causation is overlooked altogether. This again clashes with AIKR, as time is a fundamental resource for any concrete task to be executed in the physical world. For these reasons, the approach of causal calculus seems ill-suited to be used as an effective reasoning tool in any real time physical world scenario.

There are also a number of features of tasks we would like to represent in causal diagrams. The observability of some variable by the controller might be influenced by a number of factors, both inside and outside the task. These factors might act in different ways across time, and therefore the observations of a particular variable might improve or worsen in quality over time, possibly even as consequence of the controller's action. A variable might even become unobservable at some point, for example if the actions of the controller bring it outside the field of view of its sensors.

A similar argument can be made for the manipulation of variables. Unlike causal calculus, where a do operator can be applied on any variable, in the world of physical tasks controllers are usually constrained on some relatively small subset of variables that can be arbitrarily set to values in their domain. Furthermore, the effect of a manipulation might be different across time, depending on the current context of the manipulated variable and its neighbours. For example, a manipulation aimed to change the position of an actuator might have a different outcome if such actuator is already constrained in some position or not.

A task is considered to be finished when the goal variables assume the appropriate values as specified. Therefore causal diagrams should include a way to specify which variables are the goals, and how such goals can be verified by the executing controller.

The execution of a task implies the depletion of a certain amount of resources. Time, as already mentioned, is an important resource, but not the only one. Energy would be another resource which we would want to model into causal diagrams representing tasks. Every manipulation and observation in a task can be thought of as costing a certain amount of energy.

3.1.1 Variable attributes

The variables that make up a structural causal model represent different quantities of interest of the phenomenon being modelled. Apart from that, they behave essentially the same: any of them can be acted upon using the do-operator and be observed, and this is true at any time step. When modelling a task, structural causal models have to extended to allow for time-dependent partial observability and manipulability of variables.

Therefore three types, or rather attributes, of variables are to be included into causal diagrams: *manipulatable* variables, *observable* variables and *goal* variables, as introduced by Thórisson and Talbot (2018b). Every variable of an SCM can be assigned to any one of these types, or none of them, in which case the variable is generally referred to as *factor*.

Manipulatable variables

A manipulatable variable, or 'manipulatable' for short, is a variable that a controller can act upon. Using Pearl's terminology, that would mean that the controller can apply the do-operator to it and effectively set its value to any element of the variable domain. For a physical task a manipulatable variable is usually the physical interface between the mind of the agent and the actuator carrying out the issued commands. For example, a very simple robotic arm actuator could be modelled by a manipulatable variable whose domain would include $\{Move\ up, Move\ down, Move\ left, Move\ right\}$. The domain of a manipulatable variable could be as simple and as complex as the physical actuator can allow, and nothing prevents an actuator to be actually controlled by multiple manipulatables, each setting a particular parameter for the command to be performed.

It is important to not conflate the manipulatable variable with the actuator it affects; the signal sent to a finger to control its movement is different from the finger itself: while the signal carries specific information about a possibly fine-grained command the finger has to perform, the finger is characterized by an own set of variables specifying, among other things, its position in space, which cannot be arbitrarily set but follows certain laws of physics and constraints, including time and energy. To carry out the command the finger spends a certain amount of energy and takes a certain amount of time, therefore it isn't possible to set those values arbitrarily as it is the case with a manipulatable variable.

Manipulatables need not necessarily be tied to a controller's actuators. In fact, their usage as command interfaces to actuators is their finest possible level of detail, but manipulatables can also be used model more complex, high-level operations which may involve multiple actuators and a great many deal of other variables and mechanisms. They can be used to represent entire sub-tasks and sub-goals, precisely, those sub-elements of the task that a controller already knows before the task and can perform to an arbitrary level of precision. For example, a controller which knows how to open and close doors, might have a manipulatable variable with domain $\{Open\ door, Close\ door\}$ directly affecting the variables describing a door. Any part of the task that a controller already knows should not be part of the task proper and can be modelled with an appropriate manipulatable. In the case where a controller knows nothing about a task, it is at least assumed that it knows how to control its actuators, and therefore in those cases the manipulatables correspond exactly to the actuator interfaces.

It could be objected that by doing so, the objectivity of the task description is lost, because now the task depends somehow on what the learner knows how to do, which could be something that depends on the specific controller and its experience. To overcome this issue, it must be considered that it is always possible to describe a task by only using the actuator command interface, which depends solely on the body of the controller and thus on the task. Tasks whose manipulatables describe higher level action can always be re-specified to only include the low level actions made available by the actuators to the controller.

In the causal diagram, a manipulatable variable has at least one outgoing arrow to another (non-manipulatable) variable that represents the physical entity it is affecting. The affected variable might be further influenced by other variables in the task, which might constrain the degree of manipulation originating from the manipulatable. For example a robotic leg controlled by one or more manipulatable variables might not be able to carry out the issue commands if an obstacle is constraining the leg's movements. This approach allows the partial manipulability of variables: the manipulability of variables is time-dependent and also depends on the current state of the task.

Manipulatable variables have no incoming arrows in the SCM. Indeed, only the controller can affect these variables and the controller is never part of the task.

Observable variables

An observable variable, or 'observable' for short, is a variable whose value can be accessed by the controller. In other words, the values that the observables hold at a given time step are the same values that the controller receives in input at that time from its physical sensors. It is important to stress here that the observable variable does not correspond with the entity that is observed: the former is the result of a measurement process performed by a physical sensor and the latter is the actual physical entity subject to measurement. For example, the observations we perform with our eyes (the sensor) are of quite a different nature from the actual object that is being observed by measurement of the light that bounced off from it. The brain further processes data received from the optic nerve (without such additional processing, we would see the world 'upside-down') before our consciousness receives the familiar visual representation of the outside world. But, as this process implies, what is being seen does not correspond at all with what is out there, even though it should be as precise an approximation as it can be.

An observable variable therefore is a representation of a real world entity that comes into being by means of the sensor(s) measuring it. A very good sensor can guarantee a certain fidelity of representation of what it measures and possibly provide some error bounds which might or might not be known to the controller, while a not as good sensor might only provide a coarse or even a very far off measurement. Furthermore, the precision of a sensor might vary over time, depending on factors both inside and outside of the task.

This approach to observability is in effect a form of epistemological solipsism, as the only information available to the controller is that which is sensed while the physical phenomena actually described in the task are forever out of reach. An objection to this approach might be that if a controller can only ever perceive the world through its sensors, how can it affect it and verify the consequences of its actions? But this objection poses no problem at all: the controller only ever needs to affect its manipulatable variables (which are not observable, in the same way that a human does not have to observe the signals the brain is sending to their muscles to effectively control them) and verify their effect through its observables, which of course also include sensory information about the actuators themselves.

In the causal diagram, an observable variable has at least one incoming arrow from the (non-observable) variable representing the physical entity it is observing. Further incoming arrows can be drawn from other non-observable variables representing other factors that may affect the resulting observation by the sensor. These additional factors allow the partial observability of variables: they might improve or impair the quality of the observation, or even disallow it completely. This way the observability of a variable is time-dependent and might change over time during the execution of the task.

Observable variables have no outgoing arrows in the SCM. As a simplifying assumption, observation does not affect in any way the entity being observed nor any other variable. (The only context where this assumption is obviously violated is at the quantum level, which is out of scope for a task theory of practical use.)

Goal variables

A goal variable is a variable whose value needs to be in a certain range for the task to be considered successfully completed. Likewise, a negative goal variable is a variable whose value needs to be in a certain range for the task to be considered failed. A task may include any number of goal variables, and a task is successfully completed when all goal variables present the appropriate values. In the same way, a task might include any number of negative goal variables and the task is considered failed if any negative goal variable presents its value in the specified range.

3.1.2 Time, energy and other resources

Since our task theory is primarily concerned with tasks in the physical world, the scarcity of resources must always be taken into account. For every physical task, these resources always include, at least, time and energy.

Structural causal models and the related causal diagrams can already be used to model time series data (see Section 2.3.2). In particular, any causal diagram can be represented either as the full time graph or as the summary graph, and from now on every causal diagram presented in this thesis will fall in either of these two categories. Therefore time is explicitly modelled as part of the task itself.

The causal models that explicitly deal with time include the possibility of instantaneous effects. While in general instantaneous effects are a physical impossibility (because any causal effect necessarily takes some time $t > 0$), they can be considered if the time required for the effect to take place is lower than the shortest detectable time frame by the controller's sensors (which are part of the task and therefore it is purely a physical consideration independent of any specific controller).

Energy, on the other hand, can be understood as a currency spent to set the manipulatable variables. Whenever a controller wants to set the value of a manipulatable, a certain amount of energy can be detracted from the currently available energy. Furthermore, at every time step energy can be deducted as expense for the observations or just the maintenance of the controller and its body. A similar approach can be followed for any other type of resource that take part in the task.

3.1.3 Example

This general example (Figure 3.1) shows how our extension of causal diagrams can be used to model a task. For simplicity we show the diagram in the summary graph representation (see Section 2.3.2). The manipulatable variables A, B and C are marked in blue and directly affect the variables I and J of the task. Note that variable I is also directly affected by J: therefore if I represents a particular physical property of an actuator, specifying for example the angle or speed of movement, the influence of J might affect it despite J not being a manipulatable itself. Variables I to N are unobservable, not manipulatable and non-goal variables, which represent different and various properties of interest of physical entities in the task. Variables X and Y, marked in red, still represent properties of physical entities but are also goal variables, as their values need to be (or not to be) in a certain range for the task completion (failure). Variables marked in green are instead the observable variables, and their values are the sensory inputs of the controller assigned to execute the task. Notice that, for example, variable I' is not only determined by the variable that is the measurement of (I), but is further affected by K, which might influence the quality of sensory observation. The

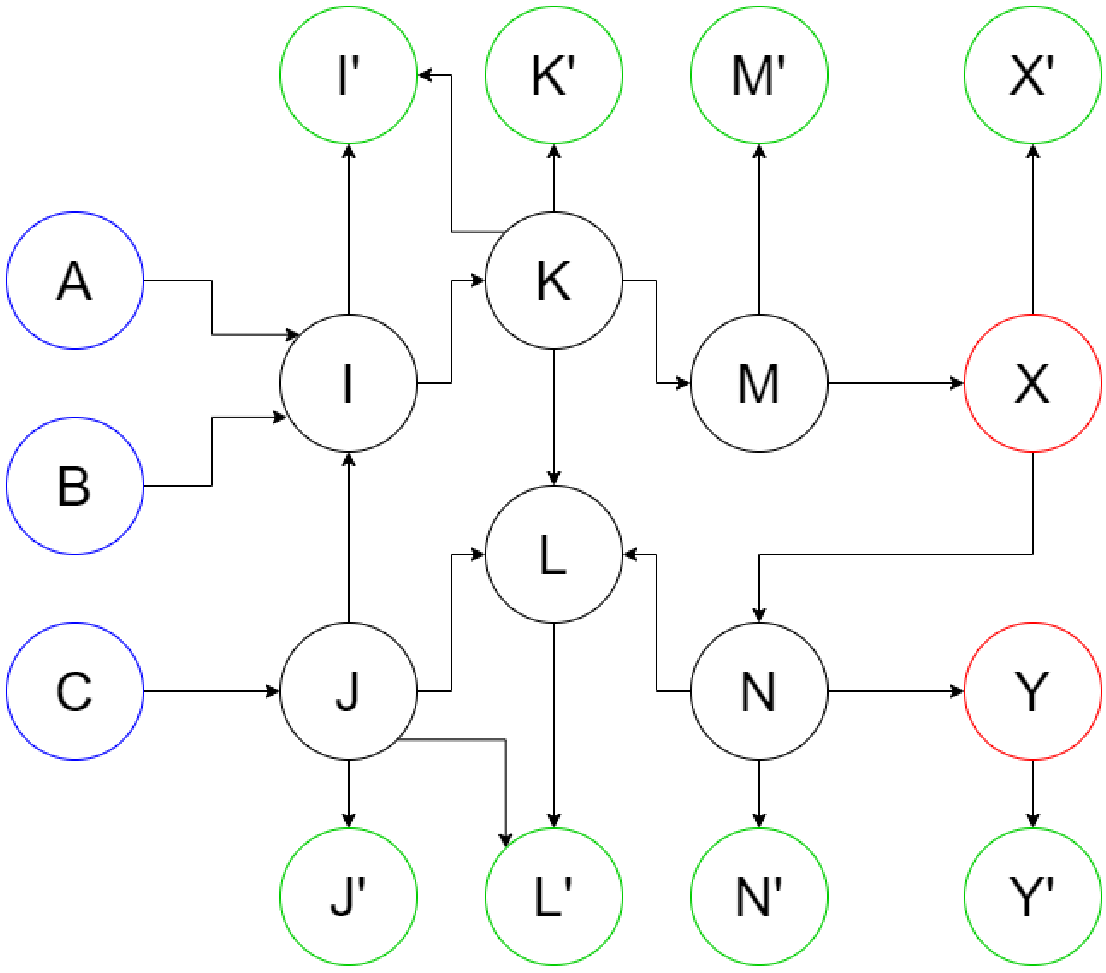


Figure 3.1: An example of summary graph representing a task. Blue nodes represent the manipulatable variables, green nodes the observables and red nodes the goal variables. All other nodes are factors.

remaining variables, marked in black, are the unobservable, non-manipulatable factors that make up the physics of the task.

3.1.4 The cart-pole task

This concrete example shows how the cart-pole task (Barto, Sutton, and Anderson, 1983) can be modelled using our extension of causal diagrams. In this task a cart has a rigid pole attached with a hinge mechanism. The cart is free to move on the x-axis along a one-dimensional track, while the pole can swing back and forth vertically on the cart. The goal of this task is to prevent the pole from falling over, and the only available action for the performing controller is to apply a force to push the cart either left or right. The cart-pole setup is visualized in Figure 3.2.

The summary graph for this task is shown in Figure 3.3, while the full time graph is shown in Figure 3.4. The actionable variable *Action* can be set to either -10 or $+10$, to push the cart left or right respectively. The observable variables are:

- x': the position of the cart on the track;

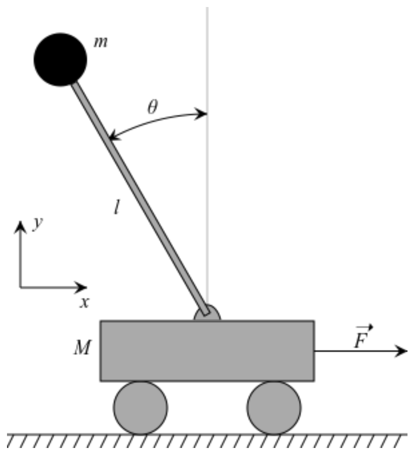


Figure 3.2: A schematic of the cart-pole task (Krishnavedala, 2012).

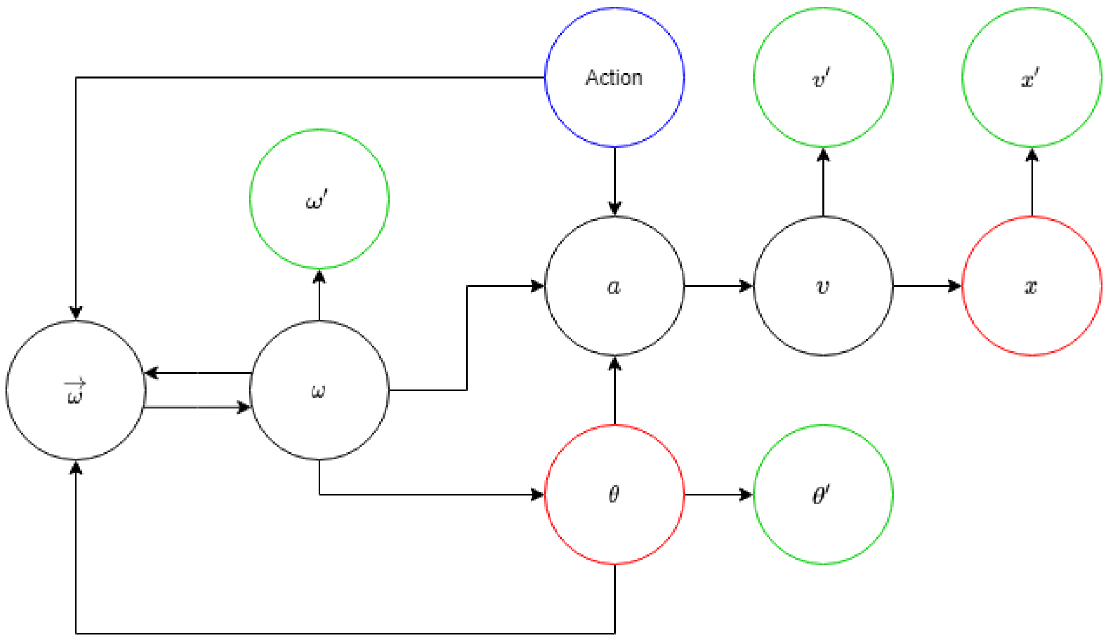


Figure 3.3: Summary graph of the cart-pole task. Note that cycles are allowed in summary graphs, but the corresponding full-time graph is acyclic.

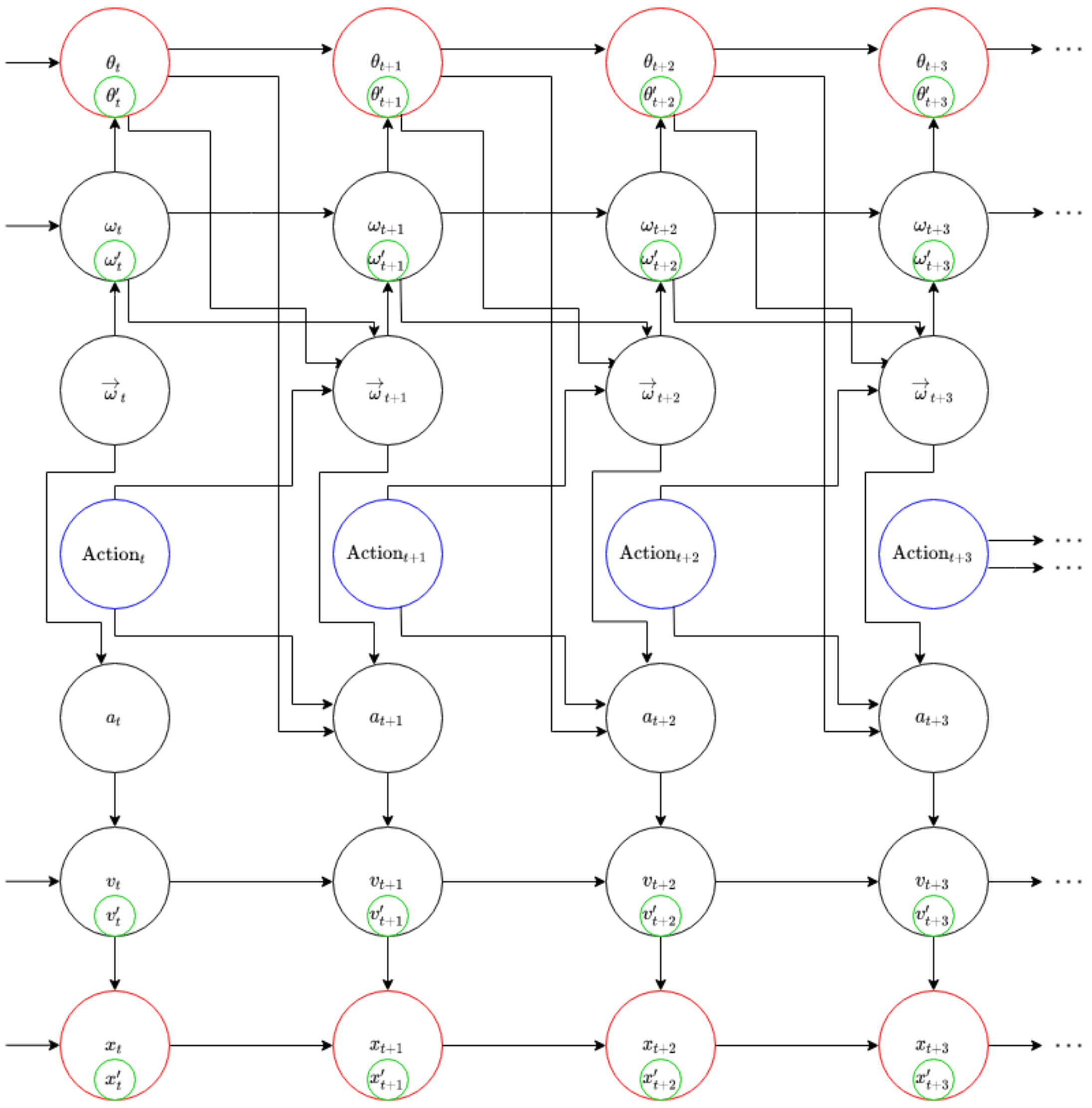


Figure 3.4: Full-time graph of the cart-pole task. To avoid unnecessary clutter, the observable variables are shown as smaller circles inside the variable node they observe.

- v': the velocity of the cart;
- θ': the angle of the pole on the cart;
- ω': the rate of change of the angle of the pole.

The goal variables are x and θ: the cart must stay within 2.4 units of the initial position and the angle of the pole must stay within 12 radians of the perfectly vertical position. Other constants in the task are the gravitational force $g = 9.81$, the mass of the cart $M = 1.0$, the mass of the pole $m = 0.1$ and the length of the pole $l = 0.5$.

The mechanisms at play in the task are described by the following structural equations:

$$\begin{aligned}
\overrightarrow{\omega}_t &= \frac{((M+m)\cdot g \cdot \sin\theta_{t-1} - (\cos\theta_{t-1} \cdot (\text{Action}_{t-1} + m \cdot l \cdot \omega_{t-1}^2 \cdot \sin\theta_{t-1})))}{\frac{4}{3}\cdot (M+m)\cdot l - m \cdot l \cdot \cos^2\theta_{t-1}} \\
a_t &= \frac{(\text{Action}_{t-1} + m \cdot l \cdot (\omega_{t-1}^2 \cdot \sin\theta_{t-1} - \overrightarrow{\omega}_t \cdot \cos\theta_{t-1}))}{M+m} \\
v_t &= v_{t-1} + a_t \cdot dt \\
x_t &= x_{t-1} + v_t \cdot dt \\
\omega_t &= \omega_{t-1} + \overrightarrow{\omega}_t \cdot dt \\
\theta_t &= (((\theta_{t-1} + \omega_t \cdot dt) + \pi) \mod (2\cdot\pi)) - \pi
\end{aligned}$$

3.2 Principles for a task theory

In the previous section we described how causal diagrams can be used to represent tasks. For the purpose of specifying a task theory, in this section we can now lay out some foundational principles of the tasks we want to represent and on which our task theory will rest on.

A problem is specified by the initial state, goal states and failure states, and by task is meant a problem assigned to a particular agent (Thórisson, Bieger, et al., 2016). Colloquially, a task is usually identified by its goals. The task of "doing the dishes" is usually thought of in terms of the end result — whether the dishes are finally clean or not is all there is to the task. This familiar definition is lacking first and foremost in forgoing any specification of the environment, that is the particular setting and context where the task is to be executed. Doing the dishes at home is one thing, but doing the dishes in space without gravity can be an entirely different experience.[1] In fact, as previously stated, the task and the surrounding environment are so tightly coupled that drawing a distinction between the two is arbitrary at best and impossible at worst (Thórisson, Bieger, et al., 2016; Bieger and Thórisson, 2017). For this reason, it is more proper to talk about a task-environment, and that is what is really meant in this thesis when the term task is used (see Glossary). Furthermore, the body of the controller has a special status because it acts as the interface between the controller and the environment. The body can be understood as a set of sensors providing sensory inputs and actuators executing the controller's commands. A task can change considerably according to the body that is provided to the controller.

1. ▷ *The environment, including the body of the controller, is part of the task.*

A possibly non-obvious consequence of considering the environment as part of the task is that the controller's body, and thus its own actuators and sensors, are part of the task too. Therefore, maintaining the running example of doing the dishes, the task can change considerably

[1]Example from K. R. Thórisson, personal communication, 2020.

if instead of being able to see the dishes the controller can only rely on tactile information. Likewise, doing the dishes using robotic arms is one thing while achieving the same objective by controlling a swarm of thousands of nano-robots is a totally different thing.

A more subtle factor to consider is the level of detail of a task. Any phenomenon in the world can be described at different levels of detail, from highly detailed fine-grained descriptions to very abstract coarse grained ones, and tasks are no exception. Task descriptions can be made arbitrarily more precise, going from the familiar human-friendly descriptions that intuitively come to mind for most tasks to overly complex descriptions at the atomic or even sub-atomic level.

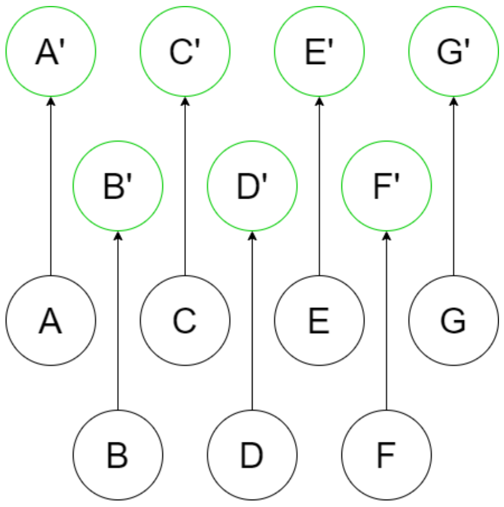
2. ▷ *The level of detail is part of the task.*

In other words, any task is limited to its level of detail, and even if the same task is presented at another level of detail, it is not the same task. For example, an electronic circuit implementing a logic gates task can be described at the level of its electronic components, at the even lower level of the chemical reactions in its circuits or at the higher level of the implemented logic circuit. The task of obtaining some output in such circuit is very different according to the level of description being used, because effectively the variables and the mechanisms changed together with the level of detail. Therefore variations in the level of detail effectively result in *different* tasks.

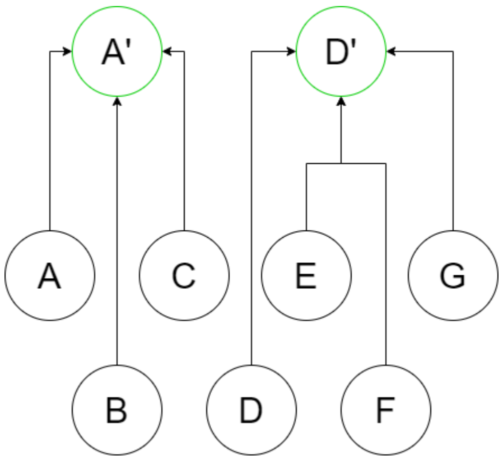
Given the various choices of different levels of detail, which level of detail is appropriate for some learner in some context? For a human learner, what is it that makes a level of detail more 'human-friendly' than alternatives? Since the body of the controller is part of the task, the lower bound on the level of detail of a task would be exactly that which is induced by the available sensors and actuators. The actuators define the granularity of what can be manipulated while the sensors define the granularity of what can be measured.

3. ▷ *A controller's sensors and actuators define the limits of relevant spatio-temporal task detail.*

Therefore, the finest possible level of detail for a task depends *necessarily on what the body allows the controller to observe and manipulate*, and tasks described at more fine-grained levels of detail than what the controller's body allow would be experienced by the controller at coarser level of detail, in accordance with what is made possible by its body. Since, in the human adult case, the sensors include the eyes (providing the visual information) and skin (providing the tactile information) and the actuators include arms and hands, it is intuitive to think of doing the dishes in a level of detail that matches their capabilities. Humans cannot naturally perceive individual atoms and even less manipulate them, therefore doing the dishes is not usually considered at any finer level of detail than that made possible by the human body. Of course physicists are well able to see and manipulate atoms with the appropriate instrumentation, therefore there might well be tasks naturally described at the atomic or subatomic level of detail; doing the dishes by a normal human body is just not one of them. Furthermore, considering the way observables are dealt with in this framework (see Section 3.1.1), a task described at such fine grained level of detail such that it is effectively out of reach for the controller's sensors would indeed be treated by the controller as if described at a coarser level of detail. Even if there were thousands of variables describing some entity, a sensor might only be able to observe such entity in terms of a handful of them, and for the controller receiving such sensory information that is all there is of the task, being forever locked out of observing the true essence of the task which lays beneath what is observable by its own body. This situation is visualized in Figure 3.5. The same variables can be perceived



(a) A situation where variables A to G have corresponding observables in A' to G', because the appropriate sensors are available for their observation.



(b) A situation where variables A to G cannot be appropriately distinguished by the available sensors, and therefore there are only two observables perceived by the controller.

Figure 3.5: An example where (a) the sensors are able to provide observables for all the variables and (b) the sensors are unable to provide observables for all the variables.

individually by corresponding observables if the sensors allow (Figure 3.5a) or they could be coalesced in a smaller number of observables if the sensors are unable to distinguish them at such degree of granularity (Figure 3.5b). Similar considerations also apply to manipulatable variables.

It is always possible to add to the description of the task by inserting new variables and mechanisms to enlarge the scope of the task description. The variables and mechanisms included in the description of the "doing the dishes" task, for example, might include a complete specification of the kitchen, or the home, and in principle of anything that exists in the universe. All this additional information seems useless, and in fact it is, for there certainly is no need to factor in the movement of celestial bodies or what is happening outside on the street to do the dishes. But humans instinctively know all of this because they are already familiar with this task and with most other tasks that exists in the real world, and therefore know which variables are relevant and which are not. Such assumption would not hold for an AI system, and even less so would hold for an AGI-aspiring baby machine that knows nothing besides its seed.

4. ▷ *A task is unaffected by variables which do not constrain its solution space.*

Let's consider an example where such human intuition about the relevance of variables could not apply. The goal is to turn on a light by toggling the correct combination of three switches. If the task includes only three switches, it would therefore take at most eight tries to find such combination. Now consider this very same task, but also including the description of 997 more switches which have no causal relation to the light. Even though the actions to carry out the task are the same as before, the task now requires additional effort to be executed: now a controller would also need to find which are the relevant switches among a thousand possible candidates. Even if it were possible to see the wiring and recognize which switches are connected to the light and which aren't, the issue of finding the relevant switches is made easier but is nonetheless present. Only with time and experience a controller would eventually start to learn that some switches are, in fact, irrelevant for the task and in future situations where a similar task occurs it could use such experience to narrow down its attention to what it believes are the relevant variables. A general understanding of the world greatly helps reasoning about which variables are useful and which are not but such reasoning doesn't apply in the case of completely novel phenomena, or when there's no previous experience (and understanding) as in the case of (machine or human alike) babies. Nonetheless, from the designer's perspective the presence of such superfluous variables does not change the task, because they do not constrain the solution space of the task, and therefore are of no-importance to execute the task. These variables might make a task harder to learn for one controller (e.g. a controller unfamiliar with the task) but they might not affect the performance of another controller at all (e.g. a controller familiar with the task which can thus recognize superfluous elements and ignore them). The set of variables and causal relationships whose *understanding* is required to carry out the task remains the same regardless of how many additional elements are added to the task description.

3.3 Intricacy

Having set forth the foundational principles, we can now turn to the issue of defining a measure that relates the complexity of the task in an objective way, that is in a way that does not depend on any specific learner to which the task may be assigned. By "complexity" here we do not refer to the classical computer science notion of the term, but rather to the intuitive concept of "complicatedness," "complexness" or "convolutedness." This measure should be grounded in the physical properties of the task, and in particular we chose to ground it in measurable causal relations/mechanisms, because it is through such mechanisms that actions have effects, things get done, and thus tasks can be executed. From this measure it will then be possible to talk about the subjective difficulty of a task for some specific controller (Section 3.4).

Definition 3.3.1 (**Intricacy** of a task T)**.** The intricacy of a task T is defined as the measure of a task's "complexity" based purely on physical, measurable parameters. It can be measured in either of the following ways:

1. The minimal number of relational models required to capture the subset of $\mathfrak{R}_T^{in}$ which includes only relations on the causal path to some goal.[2]
2. The number, length and type of mechanisms of causal chains that affect observable variables on a causal path to at least one goal.
3. The size of the smallest solution tree that can be constructed where all nodes are manipulatable variables.
4. Size of the solution space relative to the number of possible action sequences.

Intricacy thus defined is a property that relates to the physics of the task. Intuitively it is a measure of what physical mechanisms are in place that need to be known by any intelligent being, whatever is its architecture, knowledge or capabilities to perform the task in the given environment (inclusive of the controller's body). The task's intricacy is invariant on the initial values of the task's variables.

[2]The models referred by this definition are of the type described in Section 2.1 and by $\mathfrak{R}_T^{in}$ is meant the set of inward facing (causal) relations of the task (see Section 2.2).

The **preferred measure**[3] of intricacy is the one given by (1). The models referred by this definition are of the type described in Section 2.1 and by $\mathfrak{R}_T^{in}$ is meant the set of inward facing (causal) relations of the task (see Section 2.2). This minimal number of models is an objective measure which depends solely on the particular specification of the task (inclusive of the controller's body), and captures all the relevant parts of the task proper, leaving out possibly unnecessary details and relations that are superfluous for the task. This means that, for example, tasks which contain superfluous variables and relations (as described in Section 3.2) have the same intricacy of the same task abridged of all superfluities. The steps to obtain this minimal number of models entail the identification of all the relevant causal chains, turning them into relational models and then quantify them. A process for generating these models could for instance be the modelling process used by AERA (Nivel, Thórisson, et al., 2013), which can produce them in an unsupervised learning mode when given only the top-level goal(s) of a task.

The **second measure** comes directly from the level of description of causal diagrams. In this case we take into consideration the concept of causal chain (or path), that is a sequence of variables connected by arrows such that for each pair of subsequent variables (V_i, V_j) in the sequence there is an arrow starting in V_i and ending in V_j. Intricacy can then be measure by taking in consideration the observable variables of the task that lay on causal chain that contains any observable goal variable. The three parameters of concern to this definition are (a) how many distinct such causal chains there are, (b) how long are they in terms of number of variables involved and, (c) the complexity of the functions that define the mechanisms on these causal chains. A possible way to characterize the complexity of these functions is to use the Vapnik-Chervonenkis dimensions (VC-dimension). The VC-dimension associates a value to different types of function classes, for example, constants have a VC-dimension of 0, step functions have a VC-dimension of 1, linear functions in d variables have VC-dimension of $d + 1$, and generally other non-linear 'complex' functions have higher values associated than the 'simpler' linear or step functions. An even better measure for function complexity would be the Rademacher complexity, which includes the Vapnik-Chervonenkis dimension bound. A class of functions with a lower Rademacher Complexity can be understood to be easier to learn (in the context of statistical learning)[4].

The **third measure** considers the size of the smallest tree of atomic actions, that is, the values being set in the manipulatable variables, that leads to state where the task in successfully finished. The obvious drawback of this approach is that, unlike other approaches, the size of this tree depends on the specific instance of the task, that is it depends on the actual initial values of the variables that make up the task. Therefore this measure of intricacy can be used following the usual approach in computer science of asymptotic analysis considering the best, average and worst case scenarios.

The **fourth measure** of intricacy was initially proposed by Thórisson, Bieger, et al. (2016). By size of the solution space is meant the total number of task states in the solution space, with each state identified by distinct value assignment to all of the task variables. The number of possible action sequences would be the total number of possible sequences of atomic actions. It is related by Thórisson, Bieger, et al. (2016) that such a measure would at least convey the rate of success of a random performer on the task, or in simpler terms, allowing prediction of how likely any controller is to fail the task.

[3]By "preferred measure" is meant that is it the measure of intricacy that we are impartial to. The provided measures of intricacy are ordered by how powerful we think they are according to our research.

[4]For more information on the Rademacher complexity, see `http://www.cs.cmu.edu/~ninamf/ML11/lect1117.pdf` and `https://www.cs.princeton.edu/courses/archive/spring13/cos511/scribe_notes/0305.pdf`

3.3.1 Some properties of intricacy

We can now look at some implications of an intricacy measure, as discussed above, in light of its target application and domain of use.

Controllers have an upper bound on the amount of intricacy they can handle in a task Under the assumption of insufficient knowledge and resources, highly intricate tasks might be outside the capabilities of some controller to be learned, performed or both. The more intricate a task is, the more models are necessary to execute it; therefore if a controller does not possess the appropriate cognitive capabilities to handle the intricacy of the task (e.g. it cannot handle any more than X models at the same time in its working memory), it would not be able to learn the task without some outside support.

Intricacy is inversely proportional to the size of the solution space An highly intricate task should necessarily have a restricted solution space relative to all the possible action sequences. Otherwise it could counter-intuitively be possible for a random performer to finish the task more quickly (and easily) than an agent actually learning the task.

Intricacy is invariant under the initial conditions of the task The intricacy of a task remains the same across the full spectrum of possible initial values for the task; in other words, it is invariant across the possible instantiation values of a task family. This follows directly from the fact that intricacy is only concerned about the (causal) relationships in the task and not about specific variables and their values.

The lower the level of detail of a task, the higher its intricacy becomes. Tasks described at a very fine-grained level of detail are necessarily more intricate than tasks described at more coarse-grained levels of detail, because they would include a greater number of causal relationships. For example, a task described at the atomic or sub-atomic levels of description would be extremely intricate, as it would include the specification of all the involved mechanisms for each particle. The causal diagram for such a task would be likewise intractable in its complexity, for the number of nodes and arrows between them. Despite the fact that two tasks might indeed have the same macro-level goal, a description in a more coarse-grained level of detail can make all the difference in the tractability of the task for a controller.

3.4 Difficulty

From the foundational principles, in particular the principle that related the effect of superfluous variables on the task, it follows that the difficulty of executing any particular task is not uniquely determined by the task itself (i.e. its intricacy), but also depends on the performing agent. Besides previous experience with the task, some controller might be (cognitively) better or worse suited to perform the task for a plethora of reasons: it could have trained on similar tasks or on tasks which share some of the variables and relationships with this task, it could be quicker (or slower) at learning associations and cause-and-effect relationships, it could have a better (or worse) precision in controlling its actuators, and so on. Controllers, and by controller we mean effectively the mind of the intelligent system, might have either the experience or the architecture that is particularly well-suited (or ill-suited) for the task at hand, or for a type of tasks in general, or for any task at all, for reasons completely independent of the tasks themselves. Difficulty must therefore be a cross product of a task and a controller.

Given a task and an intelligent controller assigned to execute it, it is possible to talk about a measure of how difficult that task is for that specific controller.

Definition 3.4.1 (**Difficulty** of a task T for a controller C)**.** The difficulty of a task T assigned to a controller C is defined as the cross product of the task's intricacy and the level of understanding of the performing controller: $\{T \times C\}$.

This notion of difficulty can be considered to be the *effective intricacy* of a task for a specific learner. The effective intricacy of a task only takes into consideration the sub-parts of the task that the learner does not already know; in other words it's the intricacy of the task as seen from the agent's perspective. By learning the task, the effective intricacy is progressively decreased as the controller becomes more familiar with it and obtains knowledge (understanding) of how its sub-tasks can be executed. When a sub-task is successfully learned by a controller, from the point of view of the agent it stops being part of the task and its intricacy needs not be considered anymore:[5] *what a controller knows how to do is not part of the task for that specific controller.* As understanding of the task is accumulated, the task becomes easier: the difficulty decreases and when it reaches zero the controller is able to perform the task repeatedly without error.

3.4.1 Task execution phases: Learning & doing

The execution of a task can be thought of in the terms of two, possibly overlapping,[6] phases: a *learning* phase and a *performing* phase. The difficulty of a task execution can then be understood as affecting both of these phases in different ways, as shown in Section 3.4.2 and 3.4.3.

The **learning phase** is characterized by the search for models that can provide a *satisficing* solution to the problem posed by the task, where by satisficing is meant that the solution need not be optimal, but should still suffice to achieve the goals set by the task. The set of models to be learned would be a superset of the minimal set of models which identify the task's intricacy, because this set might possibly include other, superfluous and/or incorrect, models that came about from the learning process which are not necessarily useful. From the point of view of our task theory, the search for these models involves looking for the relevant observable variables that have some causal relationship with the goal variables and growing the understanding of how these variables map onto the goal variables. In other words, learning can be thought of as the process of finding the appropriate mapping from manipulatables, the atomic actions, to the observable goals and the other observables that are found to be associated with the goals. The search for this mapping is most likely happening by looking at associations in the perceived data, and it leads to the production of (possibly useful) models. More generally, the production of useful models is guided by hints found in observations, of which associations and correlations are a particular case.

The **performing phase** is the actual execution of any solution found in the learning phase, and therefore it is all about reaching proficiency in performing at least one such solution. This proficiency is progressively improved by increasing the reliability of execution of the various steps of the solution, for example by reducing the rate of error.

[5]From the designer's point of view, the intricacy of the task is unaffected.

[6]For an AGI-aspiring system or a reinforcement learner these would be overlapping to various degrees, while in the case of other machine-learning approaches these may be distinct, sequential phases.

A controller's performance on a task can be evaluated as a function of its reliability, the capacity of the controller to keep performing the task over and over again, and its robustness to the initial conditions of the task, the capacity of the controller to perform the task in its average case scenario.

There are a number of factors that affect the difficulty of learning and performing a task that are unrelated to the task's intricacy. Some of these factors that have been identified are discussed in the following two sections.

3.4.2 Difficulty factors in the learning phase

In this section are described the various factors that affect the difficulty of some specific controller in learning a task. In the context of learning, the difficulty can be easily understood as the time required to learn the task: a task is said to be more difficult to learn than another task if it takes a longer time to learn it.

Constraints on solution quality — A task might have multiple solutions, some better than the others along some metric (for example, number of steps, time or energy requirements, and so on). Specifying some restrictive constraints on the quality of a task's solution can indeed affect the difficulty of learning the task (e.g. requiring optimal energy or time costs for the solution might very well make the task close to impossible to solve by a controller).

Spurious associations — Spurious associations, where variables appear associated due to a common cause affect the difficulty of the learning phase. The presence of these associations do not affect intricacy though, because as an objective measure based on physical mechanisms, it doesn't really matter if two variables appear associated or not. In fact, such associations might be observed by some learners while others might not, therefore the identification and the influence on the learning process of associations is dependent on the cumulative understanding of the performing controller.

Anticausal associations — For every causal relationship, there are observable anti-causal associations between the effects and their causes. By "anti-causal association" is meant a causal association where the causal variables and the effect variables are reversed, i.e. a relationship that supposes that the effects caused their causes. A controller that is able to detect and understand the direction of causation is able to learn the correct model of reality more quickly. In particular, controllers which understand the notion of time are greatly facilitated, as the causes always precede their effects.

Delays of observations — By delay in observations is meant the phenomenon where the observed value of a variable is experienced with a certain time lag t after its occurrence. While such phenomenon is an inevitable occurrence in the physical world, due to the fact that transmission of information can never be instantaneous in principle, it becomes particularly relevant if such delay is much larger than the controller's minimal amount of time needed to experience a sensory stimulus. Longer delays can impact the learning of cause-and-effect relationships, because it becomes harder for the controller to ascertain which action led to which effect (it is in effect an instance of the temporal credit assignment problem). Nevertheless, delays do not affect intricacy because it do not affect the number of causal mechanisms in place.

Distinguishability of signals — The state of an observable variable might not be distinguishable from other states. The distinguishability of signals depends intrinsically on the sensors performing the measurements, and if picking out (distinguishing) the signals of some variable is difficult, the creation of accurate models (i.e. learning the task) becomes more difficult as well. There might be other variables, more easily measured, that can act as predictors of the variable whose states are difficult to pick out.

Superfluous variables — The presence of superfluous variables in the task, or rather, the inability of a controller to employ an attention mechanism to restrict its own reasoning only on the salient elements and relations of the task, can greatly affect the speed of learning of the task.

Controllability — The degree of controllability of a task can directly affect the learning process. Some variables might not be controllable in a finite time. In other words, the state of some variables the agent is interested in might not behave in a way it desires by a particular deadline, which leads to failure in performing the task. Uncontrollability is influenced by such factors as constrained action (input) ranges, constrained value ranges of variables, the absence of causal chains from control inputs to particular variables in the state space, delay effects, the existence of confounding variables that influence the causal process in an unknown manner. Besides, while some variables are directly controllable and can be set to any value at any time, some are controllable only after spending sufficient energy (control effort) and time.

3.4.3 Difficulty factors in the performing phase

Various characteristics of a task's solution influence the difficulty of performing it. The length in number of steps and how heterogeneous are the steps can all influence the difficulty of actually carrying out the task, because they have the possibility of increasing the rate of errors of the performer. Another series of factors are concerned entirely with errors in execution, for example, whether a misstep in the solution invalidates the task completely, and if not, how many such missteps can occur until the task is considered to be failed. Time is also an important factor: solutions that take more time to execute are harder to perform, all other things being equal, because they would at least be more prone to unexpected occurrences hampering the controller's performance. A sequence of steps to be executed to conclude a task might be executed synchronously or asynchronously: in the former case the various steps are executed one after another following a fixed time frame, with the task figuratively waiting for the controller to do something (e.g. like a game of chess, where each player patiently waits for their opponent's move), while in the latter case there is not any such constraint; the difficulty of executing a solution can indeed be affected accordingly if in some parts must be executed either synchronously or asynchronously. A solution might require a certain precision of execution of the various actions, and it can be expected that the greater the required precision, the harder is the solution to perform. Lastly, time, energy and other resource constraints can greatly affect the difficulty of performing a solution: when such resources are scarce, i.e. close to the physical minimum required to perform the task, it was generally found that controller incur in many more difficulties compared to situations where there is a lot of leeway in terms of available resources.

In summary, the factors that affect the performing phase that we have identified are the following:

- Length of the solution (in # of steps)
- Length of the solution (time required to execute the steps)

- Gravity of mistakes (if there are certain actions that invalidate the rest of the solution, or if it is always possible to correct mistakes)
- Number of allowed mistakes before the task is considered to be failed
- Variety of actions in the solution
- Synchronicity or asynchronicity of execution
- Precision of actions in the solution
- Available time to execute the solution (compared against the shortest possible time required)
- Available energy to execute the solution (compared against the lowest possible amount of energy required)

3.5 Decomposition of tasks

From our causal diagram representation of tasks it is possible to derive a procedure for the decomposition of tasks into its component subgoals in an objective way. The causal diagram representation of a task allows a straightforward approach for such decomposition.

Let's take for example the task of unlocking a three digit bike lock, with the assumption that the code is known beforehand. This task requires the setting of three independent digits of the code, in any order. It appears therefore intuitive that for the final goal of opening the lock, the subgoals would be exactly three, each essentially specifying to set the appropriate digit of the code. But is such decomposition objective? Why not decompose the task into setting the first two digits at once and then the last digit?

The manipulative approach to causation upon which we based our task theory provides a simple answer to these questions. Each digit of the code is set by an independent mechanism, which for example in the case of locks is a rotating gear responsible for only a single digit. This independent mechanism, which would be represented by a single node of the diagram at this level of detail, can be affected by a localized action of rotating its wheel. This action does not affect any other mechanism in the system, therefore it is local in the space of mechanisms. Compare this lock to another lock, where there are again three digits but the two left-most digits are set by a single rotating wheel which shows all possible combinations of 2-digit values $(00, 01, 02, \ldots, 98, 99)$. In this case, two digits are set by a single mechanism and a proper subgoal for this task would be: "Set the two left-most digits", instead of "Set the left-most digit" and "Set the middle digit".

Therefore the decomposition of tasks can follow the decomposition of causal diagrams into the individual nodes on the causal path to the goal variables. The order to follow when fulfilling the obtained subgoals is left to be decided by the controller.

3.6 Composition and sequencing of tasks

Likewise for decomposition, tasks can be composed by merging the corresponding causal diagrams. Endogenous variables shared by both diagrams can be merged into the same variable. Then, all is left to do is to check if any background variable u_i of any one diagram corresponds to an endogenous variable v_j in the other diagram. In such case an arrow can be drawn from

v_j to v_k, where v_k is the variable affected by u_i. After this process has been repeated for all the variables in the task, the merge is complete.

A particular type of composition is the sequential composition of tasks; many tasks are naturally thought of as a sequence of steps "Walk to the door, then open the door, then close the door". The sequential composition of tasks is a composition in which the sub-tasks being merged require a particular order for their execution, i.e. each task has some required pre-conditions that have to be met for the task to be attempted (One cannot open the door if they aren't in range for reaching the handle, so one first needs to complete the sub-task of getting there). The sequential composition of tasks can be implemented using the same rules as the classical composition, with the difference being only that some of the mechanisms in these tasks will depend on the values of some variables (representing the pre-conditional state).

3.7 Learning as a task

The body of the controller, in particular its sensors and actuators, belongs to the boundary between a task theory and a yet-to-be defined theory of controllers, a proper theory of learning. Having progressed thus far with a task theory, we are now in position to provide some insights about learning theory.

Learning itself can be considered as a task, as one of the goals in AGI research is to design reflective systems that are capable of recursive self-improvement of their learning and knowledge acquisition processes. Therefore, in the task of learning, we can consider the goal to be the production of a program with tolerance ranges across all key dimensions of the task being learned. Ideally, this program would use the *minimum* amount of time and energy, but rarely for real world tasks such aim is achievable. Instead, learners should aim for a *reasonable* cost in terms of time and energy. To define reasonability of the use of resources, we can first consider that for any task there must be a minimal number of causal relations that have to be modelled correctly by the controller. Then, a reasonable amount of time and energy would be whatever expenditure of these resources was incurred to act on this set of models.

If the task is being provided from another agent with a specification of all variables part of the task, learning then consists in finding which of these variables are relevant for achieving the goal. On the other hand, there is an additional degree of complexity if the task does not include such list, for example if this task only specifies the goal or has to be determined autonomously by the controller. In such cases, the controller also has to find which variables should it consider as part of the task in the first place, in other words, to employ an attention mechanism to home in on the set of possibly relevant variables.

CHAPTER 4

Conclusion

In this chapter we conclude this thesis by outlining some ideas for future work and answering to some criticisms that our ideas received. In Section 4.1 we describe possible approaches for testing our theory on AGI-aspiring systems and narrow-AI systems. In Section 4.2 Constructor Theory, developed by physicists David Deutsch and Chiara Marletto, is briefly introduced with the aim of drawing some connection to our work. In Section 4.3 we respond to some of our critics objections about our work and in Section 4.4 we describe some avenues for further research on this task theory.

4.1 Testing

The approach to task theory described in this work could be put to test in a number of interesting ways, each providing a variety of insights into the tasks and especially the learners assigned to them. The learners that could be used in this testing phase would include the Non-Axiomatic Reasoning System (NARS) and Autocatalytic Endogenous Reflective Architecture (AERA), which are among the foremost AGI-aspiring systems that exist today. For comparison, then, it would be interesting to compare their results with those of a reinforcement learner, e.g. an Actor-Critic Reinforcement Learner.

One way to go about testing would be to construct a number of well understood tasks, analyzed using the proposed task theory, and submit them to a number of different learners to verify whether there's any difference in the learner's capabilities and, most importantly, if the results match the expectations produced by their analysis according to the theory. Another way to go would be to construct a task such that its intricacy can be easily and intuitively increased, and see up until which point the various learners are able to cope with the increased complexity of the task.

Most interestingly, another way of testing the theory would be to construct two tasks that are overlapping in some of the variables and causal relationships between these variables. Then, by having the learners execute the first task and then the second, it would be possible to evaluate the degree of transfer learning that has occurred and the different degree of causal understanding achieved by the different learners.

4.2 Connections with Constructor Theory

Constructor Theory is a candidate "theory of everything" developed by physicists David Deutsch and Chiara Marletto. This theory aims to express all current scientific theories by defining which physical transformations, hereby called 'tasks', are *possible* and which are *impossible* (Deutsch, 2013). This stands in contrast with the usual interpretation of physics which aims to predict what will happen given some initial conditions and the laws of nature (Deutsch, 2013). A task is defined on some physical substrate as having a set of legitimate inputs for each of whom there exist one or more legitimate outputs (Deutsch, 2013), and consists in the transformation of the former into the latter. These tasks are carried out, or better yet these physical transformations are caused, by a *constructor*, which is defined as anything that can cause such transformation in the physical reality while retaining its ability to do so again (Deutsch, 2013). Therefore a constructor is capable of performing some task if whenever presented with the appropriate inputs of the task it will produce the appropriate output, all over again and a particular task is possible if it can be caused by any such constructor. (Deutsch, 2013). The notion of causation has been prominent in this short introduction to the theory, so it is interesting to note its definition:

> Since a catalyst changes only the rate of a reaction, not the position of equilibrium, it is sometimes deemed a mistake to regard catalysts as causing reactions. However, that argument would deny that anything causes anything. Even without a factory, the components of a car do spontaneously assemble themselves at a very low rate, due to Brownian motion, but this happens along with countless other competing processes, some of them (such as rusting away) much faster than that self-assembly, and all of them much slower than the assembly effected by the factory. Hence a car is overwhelmingly unlikely to appear unless a suitable constructor is present. So if causation is meaningful at all, catalysts and other constructors do indeed cause their characteristic constructions. (Deutsch, 2013, pp. 7-8)

From this point of view, we can consider something as causing something else if its action is instrumental in bringing about the result in a timely manner, a result which otherwise would not materialize because of other competing processes hampering the random and slow process of spontaneous motion of particles.

It easily follows from the definition of constructor that no such thing can exist in nature, both because of possible mistakes when performing a task and because everything in the physical reality is subject to decay and deterioration (Deutsch, 2013). The solution is to define some bounds on the energy used for the task and the error in input and output states, then defining tasks of the form "cause **C** to perform $\mathcal{A}_\epsilon$ and to remain capable of doing so again", where the constructor **C** is considered a substrate itself and $\mathcal{A}_\epsilon$ is some task with energy and error bounds defined (Deutsch, 2013). On the other hand, abstract constructors can be more familiar to us, and one such constructor is knowledge. Deutsch defines knowledge as:

> Knowledge is information which, once it is physically instantiated in a suitable environment, tends to cause itself to remain so: it survives criticism, testing, random noise, and error correction. (Here I am adopting Popper's (Popper, 1968) conception of knowledge, in which there need be no knowing subject.) For example, the knowledge encoded in an organism's DNA consists of abstract genes that cause the environment to transform raw materials into another instance of the organism, and thereby to keep those abstract genes, and not mutations or

other variants of them, physically instantiated, despite the mutation and natural selection that keep happening. (Deutsch, 2013, p. 24)

Knowledge is thus created and maintained by "intelligent beings" and the transformations described by physics are the result of the application of knowledge by some type of agent, humans for example (Deutsch, 2013).

It is interesting to note that in Constructor Theory tasks are considered with keen consideration about the available resources and their use, just like in AGI research. In (Deutsch, 2013, p. 26) a task of transmuting a mass m of hydrogen into at least a mass M gold using at most E energy is described. Using the physical law that relates mass to energy and having fixed a value for M it is possible to partition the space induced by E and m into the subspaces of possible and impossible tasks; the impossible subspace encompasses all combinations of E and m that in the physical world cannot be used to cause the transmutation of a mass M of gold, either because the limits imposed by relativity are violated or because of current limits in transmutation technology. The boundary between the possible and impossible subspaces represents the values for which the use of resources is optimal, i.e. for every value of hydrogen mass m the boundary is the minimal value of energy E to transmute it to the specified mass M of gold. The graph showing these spaces resembles very closely the graphs describing optimal time and energy usages for the tasks described in (Thorarensen et al., 2016, pp. 6-7).

It is evident that, in Constructor Theory, tasks are very similar to the concept of function, or rather, of physical mechanism, and the constructor is just the entity actually executing the function. Therefore a stark difference with our work is that in AI, tasks are a transformation from an initial state to a solution state, while in Constructor Theory they are about the conversion of input values to output values. In the above example is totally missing any mention of time, which makes sense for a function (since functions are usually considered to be timeless) but unacceptable for tasks in the physical world, which are subjected to the passage of time.

Linking with Computability Theory, programmable constructors are constructors that can load a program, i.e. knowledge, and execute the task that is encoded. It follows then that a universal constructor can exist, a programmable constructor whose repertoire, the set of tasks it can be programmed to perform, is that of all possible programmable constructors (cf. Universal Turing Machine). It seems spontaneous to ask whether people are such universal constructors. Deutsch argues in (Deutsch, 2013, pp. 38) that it is not the case, not because we might not be able to apply given knowledge (the program) to perform some task, but because as humans we might not be so inclined to carry it out again and again. Deutsch continues arguing that possibly we could be by being fooled into being destroyed after building another universal constructor, but he doesn't think that such thing can be done with high reliability. Therefore humans can at most be a rough approximation of a universal constructor.

The question that now arises is: would a truly AGI system be a universal constructor? Would we want it to be so, or would we rather want it to be more like us? Pearl argues (Pearl and Mackenzie, 2018, pp. 328-329) that robots would greatly benefit by having, just like us, the illusion of free will, as it might enhance communication among themselves and especially with humans. For the former, the argument goes that by thinking in terms of intents and willed choices it would be much easier for them to relay complex causal instructions while for the latter case the advantage is that it would make communication with us more natural, by virtue of sharing the same assumptions about the free will. Pearl relates that believing in their own free will would also help with the robot's reflective capabilities, by making it able to reason about its intents and possibly act differently.[1] If an AGI system is built to work under

[1]Therefore it would help for conterfactual reasoning, of which, as already argued, only modern humans are

the illusion of free will just like we are, it is not hard to see that it would too be only a rough approximation of a universal constructor, albeit better than any single human or humanity as a whole, provided that it is not programmed to have mechanisms such as boredom or rebelliousness.[2] Yet, all the aforementioned objections about humanity not being a universal constructor would apply to it as well.

4.3 Criticisms

In this section are presented some of the criticisms that were raised by other researchers that were introduced to this work while it was still in progress.

Subjectivity of causation Another objection sometimes raised is about the existence of causal relationships and causality, in the description of tasks. It is argued that for an intelligent system there is no use for the belief of objective causation, since an agent will never have the full knowledge of the environment or itself to derive any "true causal relationships" at all (Wang and Hammer, 2015). It is therefore much more useful for an agent to use a mechanism of temporal inference instead of causal inference, which would nonetheless result quite close to the typical notion of "causation" in everyday life (Wang and Hammer, 2015).

A: In our work, the usage of "objective causal relationships" is practically motivated and limited to the description of the tasks from the *designer's perspective*, which assumes that everything is known a-priori (p. 34). From the perspective of a learner, who does not have access to the designer's perspective, the concept of "objective causal relationships" is replaced by "useful causal model" (p. 48), and thus this criticism does not apply. When it comes to the learning and performing of tasks by some controller, no assumption of objectivity is made about the models or any other information structure used by the learner to describe hypothesized "actual, objective causal relationships" that lay at the heart of the task's physics.

Variables versus events An objection that was raised to our work is that we do not make clear the difference between a variable and an event. Variables are passive entities which hold values, while events, which include actions that are executed by the performing controller, are the actual bearers of change inside the task-environment. So how can variables, specifically the manipulatable variables, act in the same way as events?

A: The manipulatable variables are just variables indeed, but the controller can affect them directly by *setting* their value to something, in the same way that Pearl's do-operator intervenes on the variables of a causal model (p. 36). So, in this sense, there are still events, in the form of actions carried out by a controller, which by affecting the manipulatable variables bring about changes into the task-environment. Of course, any action in the physical world takes time and requires energy; a mountain climber may "set her energy" on the mountain top, but that doesn't bring her there instantaneously. In this way, targeting a particular value for the tension on a muscle may take a very short time, but making the muscle move the arm or leg to a target position takes longer. In both cases, however, the occurrences are events, one taking longer than the other. The purpose of task theory is to allow convenient analysis and dissection of complex tasks; how variables and events relate to each other remains to be further worked out.

capable of and would be immensely useful to have for AGI systems.

[2]Even if such things could be programmed, there is no good reason to endow a system with them.

Task versus problem One objection that was raised is that the distinction between 'task' as used here and the classical computer science notion of 'problem', which is typically defined as "a problem that can be solved by a computer," is not clarified.

A: Since in our work 'task' is defined as a problem assigned to an agent with variable bindings (including when it may start, how much time it may take, etc.) (see Glossary), this objection is resolved once this fact is taken into consideration.

4.4 Future work and open issues

In this section we outline some pointers for future work on task theory and some open issues that still need to be addressed.

Similarity An objective similarity measure between observable variables, in particular between the observable goal variables and the other observables, could be developed. This measure should be able to quantify how direct is the mapping between any two observables, i.e. it should quantify with a number the degree of association between variables. In the case of linear associations, this measure might correspond to the classic notion of correlation. As there exist spurious and non-spurious associations, we might talk of a relevant similarity which aids the controller in learning the task, and irrelevant similarity which doesn't. Sheikhlar, Thórisson, and Eberding (2020) define a number of similarity measures for variables, states, relations and transition functions, but for the similarity of variables their work is limited on considering the temporal and value similarity of the same variable.

Precision Another interesting measure that could be defined is precision, the measure of measurability and controllability of variables related to a (sub-)goal. It would be affected by the quality and resolution of observation provided by the sensors and quality and resolution of control provided by actuators.

Unlearnable tasks Some tasks can be defined in such a way that their successful execution depends on other variables that are unobservable, i.e. non-measurable knowledge. For example, the task of opening a lock without knowing the combination and with a limited number of tries can only be attempted by random guessing in absence of other hints in the task (e.g. is the combination written somewhere that can be observed?). If such is not the case, the successful execution of the task depends solely on the knowledge of the performing controller, which *could* know the code from other experience outside of the task. It should always be ascertained that if a task depends on such outside knowledge, unlearnable from the task itself, the learner possesses such knowledge.

4.5 Final remarks

A whole field of research could spring forth from this initial work on task theory. Even though not all of the requirements for a task theory (Section 1.2) might have been achieved, this work on a theory of tasks can still provide a starting point for future efforts by other researchers. In this sense, any type of advancement, or any subset of the aforementioned objectives being achieved in the future, is still a worthwhile endeavor that would bring us closer to a complete task theory, for which little to no work has been performed otherwise, which, once successfully specified, would be an immense boon for the field of AI.

Bibliography

Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson (1983). “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5, pp. 834–846. DOI: 10.1109/TSMC.1983.6313077.

Beishon, R. J. (1967). “Problems of Task Description in Process Control”. In: *Ergonomics* 10.2, pp. 177–186. DOI: 10.1080/00140136708930857. eprint: https://doi.org/10.1080/00140136708930857. URL: https://doi.org/10.1080/00140136708930857.

Berkson, Joseph (1946). “Limitations of the Application of Fourfold Table Analysis to Hospital Data”. In: *Biometrics Bulletin* 2.3, pp. 47–53. ISSN: 00994987. URL: http://www.jstor.org/stable/3002000.

Bieger, Jordi and Kristinn R. Thórisson (2017). “Evaluating Understanding”. In:

— (Aug. 2018). “Task Analysis for Teaching Cumulative Learners”. In:

Card, Stuart K., Allen Newell, and Thomas P. Moran (1983). *The Psychology of Human-Computer Interaction.* USA: L. Erlbaum Associates Inc. ISBN: 0898592437.

Clark, Richard et al. (Jan. 2008). “Cognitive task analysis”. In: *Handbook of Research on Educational Communications and Technology*, pp. 577–593.

Conant, Roger C. and W. Ross Ashby (1970). “Every good regulator of a system must be a model of that system”. In: *Intl. J. Systems Science*, pp. 89–97.

Crandall, Beth, Gary Klein, and Robert Hoffman (Jan. 2006). *Working Minds: A Practitioner’s Guide to Cognitive Task Analysis.* ISBN: 9780262270922. DOI: 10.7551/mitpress/7304.001.0001.

Deutsch, David (2013). *Constructor Theory.* arXiv: 1210.7439 [physics.hist-ph].

Drury, C.G. (1983). “Task analysis methods in industry”. In: *Applied Ergonomics* 14.1, pp. 19–28. ISSN: 0003-6870. DOI: https://doi.org/10.1016/0003-6870(83)90215-6. URL: https://www.sciencedirect.com/science/article/pii/0003687083902156.

Eberding, Leonard, Arash Sheikhlar, and Kristinn R. Thórisson (June 2020). “SAGE: Task-Environment Platform for Autonomy and Generality Evaluation”. In:

Galton, Francis (Jan. 1888). “Co-Relations and Their Measurement, Chiefly from Anthropometric Data”. In: *Proceedings of the Royal Society of London Series I* 45, pp. 135–145.

Garrett, Deon, Jordi Bieger, and Kristinn R. Thórisson (Dec. 2014). “Tunable and generic problem instance generation for multi-objective reinforcement learning”. In: pp. 1–8. DOI: 10.1109/ADPRL.2014.7010646.

Kim, Jin and Judea Pearl (1983). “A Computational Model for Causal and Diagnostic Reasoning in Inference Systems”. In: *IJCAI*.

Krishnavedala (2012). *Schematic drawing of an inverted pendulum on a cart.* File: `Cart-pendulum.svg`. URL: `https://commons.wikimedia.org/wiki/File:Cart-pendulum.svg`.

Kurke, Martin I. (1961). “Operational Sequence Diagrams in System Design”. In: *Human Factors* 3.1, pp. 66–73. DOI: `10.1177/001872086100300107`. eprint: `https://doi.org/10.1177/001872086100300107`. URL: `https://doi.org/10.1177/001872086100300107`.

Legg, Shane and Marcus Hutter (2007). *A Collection of Definitions of Intelligence.* arXiv: `0706.3639 [cs.AI]`.

Militello, Laura and Robert Hutton (Dec. 1998). “Applied Cognitive Task Analysis (ACTA): A Practitioner’s Toolkit for Understanding Cognitive Task Demands”. In: *Ergonomics* 41, pp. 1618–41. DOI: `10.1080/001401398186108`.

Miller, Robert (1953). “A METHOD FOR MAN-MACHINE TASK ANALYSIS”. In:

Nivel, Eric, Kristinn R. Thórisson, Bas Steunebrink, Haris Dindo, et al. (2014). “Bounded Seed-AGI”. In: *Artificial General Intelligence.* Ed. by Ben Goertzel, Laurent Orseau, and Javier Snaider. Cham: Springer International Publishing, pp. 85–96. ISBN: 978-3-319-09274-4.

Nivel, Eric, Kristinn R. Thórisson, Bas Steunebrink, and Jürgen Schmidhuber (2015). “Anytime Bounded Rationality”. In: *Artificial General Intelligence.* Ed. by Jordi Bieger, Ben Goertzel, and Alexey Potapov. Cham: Springer International Publishing, pp. 121–130. ISBN: 978-3-319-21365-1.

Nivel, Eric, Kristinn R. Thórisson, et al. (2013). *Bounded Recursive Self-Improvement.* arXiv: `1312.6764 [cs.AI]`.

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 1558604790.

— (2009). *Causality. Models, Reasoning, and Inference.* 2nd ed. Cambridge, UK: Cambridge University Press. ISBN: 978-0-521-89560-6. DOI: `10.1017/CBO9780511803161`.

Pearl, Judea and Elias Bareinboim (Nov. 2014). “External Validity: From Do-Calculus to Transportability Across Populations”. In: *Statistical Science* 29.4. ISSN: 0883-4237. DOI: `10.1214/14-sts486`. URL: `http://dx.doi.org/10.1214/14-STS486`.

Pearl, Judea and Dana Mackenzie (2018). *The Book of Why: The New Science of Cause and Effect.* 1st. USA: Basic Books, Inc. ISBN: 046509760X.

Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf (2017). *Elements of Causal Inference: Foundations and Learning Algorithms.* The MIT Press. ISBN: 0262037319.

Popper, Karl (1968). “Epistemology Without a Knowing Subject”. In: *Logic, Methodology and Philosophy of Science III.* Ed. by Bob Van Rootselaar and Johan Frederik Staal. Vol. 52. Studies in Logic and the Foundations of Mathematics. Elsevier. Chap. 3, pp. 333–373. DOI: `https://doi.org/10.1016/S0049-237X(08)71204-7`. URL: `https://www.sciencedirect.com/science/article/pii/S0049237X08712047`.

Reichenbach, Hans (1956). *The Direction of Time.* Dover Publications.

Riedl, Mark O. (2014). “The Lovelace 2.0 Test of Artificial Creativity and Intelligence”. In: *CoRR* abs/1410.6142. arXiv: `1410.6142`. URL: `http://arxiv.org/abs/1410.6142`.

Sackett, David L. (1979). “Bias in analytic research”. In: *Journal of Chronic Diseases* 32.1, pp. 51–63. ISSN: 0021-9681. DOI: `https://doi.org/10.1016/0021-9681(79)90012-2`. URL: `https://www.sciencedirect.com/science/article/pii/0021968179900122`.

Sheikhlar, Arash, Kristinn R. Thórisson, and Leonard Eberding (July 2020). "Autonomous Cumulative Transfer Learning". In: pp. 306–316. ISBN: 978-3-030-52151-6. DOI: 10.1007/978-3-030-52152-3_32.

Thorarensen, Thröstur et al. (Aug. 2016). "FraMoTEC: Modular Task-Environment Construction Framework for Evaluating Adaptive Control Systems". In:

Thórisson, Kristinn R. (2020a). "Discretionarily constrained adaptation under insufficient knowledge & resources". In: *Journal of Artificial General Intelligence* 11.2, pp. 7–12.

— (2020b). *Lecture notes in Advanced Topics in Artificial Intelligence.* http://cadia.ru.is/wiki/public:t720-atai-2012:what_is_agi. [Online; accessed 7-July-2021].

— (2020c). *Lecture notes in Advanced Topics in Artificial Intelligence.* http://cadia.ru.is/wiki/public:t-720-atai:atai-20:agents_and_control. [Online; accessed 7-July-2021].

— (2020d). *Lecture notes in Advanced Topics in Artificial Intelligence.* http://cadia.ru.is/wiki/public:t_720_atai:atai-20:knowledge_representation. [Online; accessed 29-June-2021].

Thórisson, Kristinn R., Jordi Bieger, et al. (July 2016). "Why Artificial Intelligence Needs a Task Theory – And What It Might Look Like". In: vol. 9782, pp. 118–128. ISBN: 978-3-319-41648-9. DOI: 10.1007/978-3-319-41649-6_12.

Thórisson, Kristinn R., David Kremelberg, et al. (July 2016). "About Understanding". In: ISBN: 978-3-319-41648-9. DOI: 10.1007/978-3-319-41649-6_11.

Thórisson, Kristinn R. and Arthur Talbot (July 2018a). "Abduction, Deduction & Causal-Relational Models". In:

— (2018b). "Cumulative Learning with Causal-Relational Models". In: *Artificial General Intelligence.* Ed. by Matthew Iklé et al. Cham: Springer International Publishing, pp. 227–237. ISBN: 978-3-319-97676-1.

Turing, Alan M. (Oct. 1950). "I.—Computing machinery and intelligence". In: *Mind* LIX.236, pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433.

Wang, Pei (2004). "Toward a Unified Artificial Intelligence". In: *AAAI Technical Report.*

— (Apr. 2012). "The assumptions on knowledge and resources in models of rationality". In: *International Journal of Machine Consciousness* 03. DOI: 10.1142/S1793843011000686.

— (2019). "On Defining Artificial Intelligence". In: *Journal of Artificial General Intelligence* 10.2, pp. 1–37. DOI: doi:10.2478/jagi-2019-0002. URL: https://doi.org/10.2478/jagi-2019-0002.

Wang, Pei and Patrick Hammer (July 2015). "Issues in Temporal and Causal Inference". In: pp. 208–217. ISBN: 978-3-319-21364-4. DOI: 10.1007/978-3-319-21365-1_22.