Modelling cognitive processes with probabilistic programming

Developing a generative process for the self-serving bias

Bachelor Thesis

Department of Cognitive and Differential Psychology
University of Mannheim

submitted by
Niklas Höpner - 1455147

March 27, 2020

Supervisor: Prof. Erdfelder

Abstract

A new framework for developing computational models of cognitive processes is introduced. The basis of the framework is formed by the assumption that humans structure their knowledge using abstract intuitive theories. Different classes of cognitive processes are framed as performing inference on an intuitive theory. Models of an intutive theory and inference procedures over these models can computationally be realized via probabilistic programming. The framework is explored by implementing a computational model of the self-serving bias, the phenomenon that people tend to attribute success to themselves and failure to external factors. For that an intuitive theory of performance situations is constructed and used as part of a process model of the self-serving bias. The process model is implemented in Pyro and evaluated on its ability to simulate experimental results. The main external-internal bias in attribution of success and failure can be simulated as well as other moderator effects.

Contents

1 Introduction

Humans are capable of navigating through a complicated world which they barely understand with amazing ease. Many people are able to use a computer, drive a car, use a washing-machine without being able to explain how any of these utilities work. The varieties of tasks a human can perform and especially learn to perform seems limitless. This also enables humans to adapt effortlessly to new environments. Take for example the rise of smartphones. Humans can navigate a smartphone using the touchscreen without ever having been told how a touchscreen even works.

The ability of humans to learn and reason over this world is unparalleled by the performance of any machine. The idea of a machine that can reason and understand the world on par with humans has always been at the heart of artificial intelligence (AI) and even at the heart of Computer Science. Turing offered the first definition of an 'intelligent' machine in form of the Turing test (Turing, 1950). A human holds a conversation with two entities via text at a computer. One of them is a machine while the other one is a person. The goal of the human is to ask questions and find out which of the entities is the person. Whether this definition of machine intelligence is suitable is debatable since the dialogue system ELIZA (Weizenbaum et al., 1966) already fooled a large fraction of humans that interacted with it. Knowing that ELIZA uses simple pattern matching to construct its dialogue, hardly anyone would call it an intelligent machine. Apart from that, the Turing test illustrates that the founding fathers of the field of Computer Science already thought about creating machines with human-level intelligence. Over the course of its history the field field of artificial intelligence (Nilsson, 2009) has produced remarkable results using a variety of techniques. Early computer algebra systems such as Macsyma (Martin & Fateman, 1971) used symbolic reasoning to manipulate mathematical equations similar to humans. In 1995 Gerry Tesauro developed a reinforcement learning algorithm that learned to play backgammon (Tesauro, 1995), surpassing human abilities. With the introduction of AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) the deep learning revolution started (Sejnowski, 2018). AlexNet is a deep convolutional neural net that enormously reduced the error rate in object-classification on the ImageNet datatset (Deng et al., 2009). Driven by large datasets and enormous computational resources deep learning managed to revolutionize many subdisciplines of AI such as computer vision, speech recognition and natural language processing. However having one model or system that exhibits intelligent behaviour in such a versatile way as humans do, still seems out of reach for now (Marcus, 2018). One example that illustrates the gap between human intelligence and machine intelligence is the ability to generalize from a few examples. Current deep learning architectures need million of datapoints to do well. For example, the state of the art language model BERT (Devlin, Chang, Lee, & Toutanova, 2018) is trained on the whole of wikipedia. In contrast to that human babys can learn the meaning of a single word from a single exposure (Carey & Bartlett, 1978).

One approach to close that gap proposed by cognitive scientist Josh Tennen-

baum and colleagues at MIT (Lake, Ullman, Tenenbaum, & Gershman, 2017) is to see cognitive processes as probabilistic inference over intuitive theories. It is assumed that humans structure their world knowledge of a domain in an intuitive theory. That intuitive theory summarizes the key entities of a domain and their causal relationships. The approach builds around the idea of using insights from cognitive science about the way human thinking works to construct machines that exhibit a similar versatile intelligence. The hope is that both cognitive science and artificial intelligence can benefit from each other. Artificial intelligence can use cognitive sciences to derive inspiration in constructing systems that learn as efficiently as humans and generalize to new tasks as easily as humans do. Cognitive science could benefit from any computational approaches that are developed for modelling purposes in AI and may obtain new tools to model cognitive processes.

One goal of this bachelor thesis is to explore the relationship between artificial intelligence and cognitive science. More concretely, the thesis explores a framework for modelling cognitive processes using techniques from probabilistic programming. Although the claim stands that this framework is general and can be used to model many different cognitive processes, a first attempt is undertaken by trying to model the self-serving bias. The second chapter gives an introduction into intuitive theories and how they can help in modelling cognitive processes. The third chapter introduces the self-serving bias as a classical cognitive bias. Here the focus lies on moderator variables and process theories that explain how the self-serving bias comes into existence. The fourth chapter introduces probabilistic programming which is a necessary tool to denote an intuitive theory and perform probabilistic inference on it. Probabilistic programming is an extremely exciting field in itself but only the tools necessary for modelling the self-serving bias will be introduced. In the following chapter a framework is provided with which a model of the self-serving bias can be implemented and tested. The goal is to design the framework as abstract as possible to be able to test any model of the self-serving and potentially even any model of any cognitive process. Different models of the self-serving biases are evaluated by how well they simulate experimental data. The conclusion discusses the results and potential difficulties and limitations of this modelling approach.

2 Intuitive Theories and Generative Processes

Intuitive theories offer a framework that can be used to understand higher-order cognitive processes such as reasoning and planning. This sections discusses the meaning of the term intuitive theory, outlines some properties of intuitive theories, discusses how an intuitive theory can be represented computationally and how a cognitive process can be seen as an inference procedure over an intuitive theory.

Over the course of their life, humans accumulate a wide variety of knowledge. This world knowledge is stored in the long-term memory. Different types of long-term memories exist for different types of knowledge. One usually differ-

entiates between the declarative memory and the procedural memory. Often the declarative memory is further divided into the semantic and episodic memory (Hoffmann & Engelkamp, 2016). Here a simplified perspective is taken and it is assumed that any knowledge a person has about a domain is stored in an intuitive theory. Intuitive theories consists of an ontology of concepts and laws that dictate how these concepts relate to each other (Gerstenberg & Tenenbaum, 2017). Take for example intuitive psychology. Humans use intuitive psychology to explain the actions of other people (Wright & Mischel, 1988). In a study of the mood effects on the fundamental attribution error participants were asked to read an essay and afterwards judge the characteristics of the essay writer on a number of scales, such as intelligence and likeability (Forgas, 1998). If the essay was favouring a position that the participant opposed, then the essay writer was seen as unintelligent. The specifics of the analyzed effects are not of interest but rather that participants explained the action (essay writing) using personality traits (intelligence).

The concepts that make up an intuitive theory can be very abstract such that the theory can be used for a large number of possible situations. Heider's study on apparent behaviour (Heider & Simmel, 1944) can be seen as a prime example on how abstract intuitive theories, here intuitive psychology, can be. During the experiment participants witness a short video clip in which geometric shapes such as a circle and a triangle move around. After the participants have watched the movie they are asked to describe and explain the behaviour of the geometric shapes. Interestingly, the participants assign human like notions such as aggressive or protecting towards these simple shapes just on the basis on how they move.

Intuitive theories must be formulated probabilistically to be able to deal with uncertainty (Gerstenberg & Tenenbaum, 2017). For example in most scenarios in which intuitive psychology is applied there is more than one explanations for the observed behaviour. Due to incomplete information all of the possible explanations may be valid. Assume a person observes another person going into a bank. Possible explanation for that behavior are that the person wants to withdraw cash or that he wants to set up a bank account or might even wants to cancel his bank account. In different situations different alternatives vary in their likelihood depending on the context. Any computational framework modelling intuitive theories must therefore be probabilistic.

The type of concepts that make up the ontology of an intuitive theory consists of observable variables and unobservable variables (Gerstenberg & Tenenbaum, 2017). The unobservable variables are used to explain observable variables. To illustrate that, take again intuitive psychology. Usually the action of a person is observed and any explanation of that action is done via unobservable mental states such as desires and beliefs or unobservable personality characteristics. In the example of a man walking into a bank, the action is observed and it is explained by the desire (to withdraw cash or open a bank account).

As mentioned before an intuitive theory not only consists of related concepts but these concepts are structured by some overaching principle. In intuitive psychology the important concepts are action, belief and desire and they are

structured by the principle of rational action (Gerstenberg & Tenenbaum, 2017). That is people perform actions that let them reach their desires in a maximal efficient way given their beliefs about the world. A person that walks into a bank with the desire to withdraw cash would not do so if he would not belief that he could do so at the bank. The abstractness and the strong structural relations between concepts of an intuitive theory enable people to learn and reason using only extremely sparse data. If a person observes an intentional action by another person for the first time, he still knows there must be some desire and belief that elicited the intention to perform the action. Looking at the context in which the person performed the action the observer may deduce a desire and belief without ever having been given an explicit example from which he can learn.

Intuitive theories are quite stable. Even if received information contradicts an intuitive theory this rather leads to the information being ignored than the theory being changed. This can be observed in effects such as the confirmation bias (Nickerson, 1998). The confirmation bias suggest that people rather make use of new information if it supports their prior belief/hypothesis about that topic. In a study from Lord, Ross Lepper (1979) people are first assessed on their opinion on capital punishment. Afterwards participants are presented with two studies dealing with the effect of capital punishment on crime rates. One study supports the hypothesis that capital punishment lowers crime rates while the other one presents evidence for the opposite viewpoint. After reading both studies participants favour the study supporting their prior opinion and tried to discredit the other study by pointing out weaknesses in the design although the methodology of both studies is the same. The idea is that intuitive theories change if the evidence against them becomes overwhelming (Gerstenberg & Tenenbaum, 2017).

To summarize an intuitive theory consists of interrelated concepts that are structured by an organizing principle. These theories are often formulated on an abstract level and can be applied to various situations. They are robust towards new evidence and probabilistic in nature.

The research on intuitive theories has often been focused around intuitive physics and intuitive psychology. Intuitive physics describes how humans understand the physical world around them. There has been little to none research on using intuitive theories as a more general guiding principle of human thinking. If one assumes that intuitive theories structure human thinking then there are many unaddressed questions. Is any form of knowledge structured using an intuitive theory or just causal knowledge? Take for example factual knowledge such as the fact that Germany consists of 16 states. One could incorporate such knowledge as seeing it as an attribute of the concept of Germany or as a relation between the concept of Germany and the concept of state. Further, it is unclear on which level of abstractness an intuitive theory is formulated and how it is applied to a concrete situation. As mentioned before the central concepts of intuitive psychology are action, desire and belief. However, given an observed action it is unclear with what content a human reasoning about that action fills the concepts of desire and belief. It becomes obvious that there are

many unanswered questions regarding intuitive theories as a core principal of human thinking. This also complicates the computational implementation of an intuitive theory. As will become apparent later, most computational modelling choices are guided by intuition rather than based on scientific evidence about intuitive theories. During the rest of the thesis I stay agnostic whether intuitive theories can be used to structure human thoughts and focus on modelling the self-serving bias using inspirations from intuitive psychology.

After looking at some of the key properties of intuitive theories, the next step is to ask what offers a good framework for modelling intuitive theories. An intuitive theory itself only determines the concepts that play a role and some notion of how those concepts are related. Take for example the situation of a student taking an exam. The student may have an intuitive theory regarding how the performance in the exam is determined. For example one intuitive theory could be that his skill, effort, the harshness of grading by the teacher and luck all influence his exam performance. A high skill and effort level leads to a good performance. If the grading of the teacher is demanding it leads to a worse performance. Luck just summarizes some factors which are beyond anyone's control such as how fit the student feels at the time of the exam. In figure 1 this intuitive theory is summarized by a directed graph. Each node describes a concept and the directed edges describe how these concepts are related. A graph itself is an intuitive choice for a data structure to represent a set of elements that are related to each other. A directed graph is useful in denoting the structure of the intuitive theory. In the next step one must decide on how to model the individual concepts and their relations. Since an intuitive theory must be able to express uncertainty about values of an unobserved variable or an outcome a natural choice is to model each concept as a random variable with a certain distribution. A relation between two concepts is then expressed by a conditional distribution. The fact that a high skill level leads to a good performance can be expressed by the fact that the conditional probability distribution $p(x_{performance}|x_{skill} = high)$ puts more probability mass on good performance than the probability distribution $p(x_{performance}|x_{skill} = low)$. Modelling the joint probability distribution of finitely many random variables and encoding conditional probability assumptions using graphs leads to probabilistic graphical models. A formal introduction into these types of models is given in section 4.

For now the goal is to see such a probabilistic graphical model as a generative process. This can be best understood as seeing the graphical model telling a story about how an outcome comes into existence. In the example displayed in figure 1 the idea is that first sample values of skill, effort, harshness of grading and luck are drawn from their respective distributions. Then the exam result is drawn from the conditional distribution $p(x_{perf.}|x_{skill}, x_{effort}, x_{grading}, x_{luck})$. Sampling the joint distribution of the random vector

$$X = (x_{skill}, x_{effort}, x_{grading}, x_{luck}, x_{performance})$$

in such a manner will in the rest of the thesis be referred to as sampling the

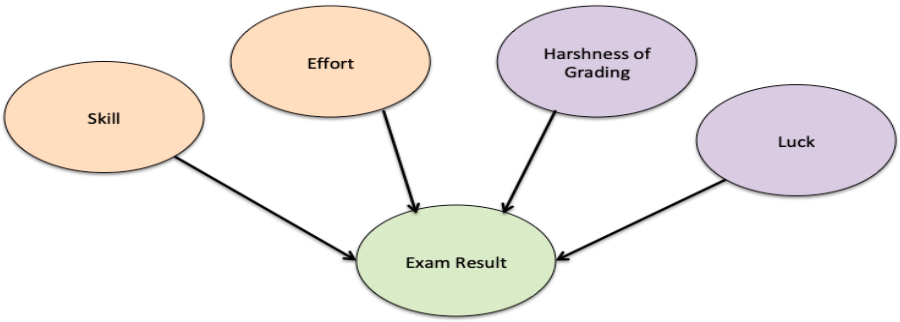

Figure 1. Graphical representation of an intuitive theory a student may have about how exam results come into existence. Each node of the graph represents a concept. The directed edges show which concepts influence other concepts.

generative process. The main advantage of modelling intuitive theories in such a way is that it allows us to perform inference on unobserved values and perform simulations about the outcome for different settings of the variables. Below it is argued that this inference and mental simulations correspond to cognitive processes performed by humans. Probabilistic programming offers the possibility to flexibly denote any probabilistic graphical model and perform inference on it using a universal programming language. The necessary tools related to probabilistic programming are also introduced in section 4.

At the end of the day our goal is to model higher cognitive processes. An intuitive theory itself is only an idea how humans structure their knowledge. However, humans can use their knowledge to make predictions, give explanations for why events happened or make judgements about values of certain variables. The next step is now to describe higher cognitive processes as inference and simulations using intuitive theories. Rather than arguing that any cognitive process can be described using this framework a number of typical higher cognitive processes are explained in this framework. Take for example making a prediction about the future. Humans constantly form an expectation over what is about to happen next. There are many well-studied effects in different disciplines of psychology about the influence of expectation on how an event is perceived. In marketing it is well known that the expectation of the taste of a product has a strong influence on the actual taste. In one study participants taste two different brands of beer. One group is informed prior to the tasting event that one of the beers contains a few drops of vinegar. For both groups the ingredients of the beer brands are exactly the same. Only the group with the prior information about the vinegar marks one of the beers as tasting bitter, indicating that expectation plays a large role in how a person perceive taste (Lee, Frederick, & Ariely, 2006). The strength of the self-serving bias which will formally be introduced in section 3 also depends on the expectation of a

person about the outcome of the related performance event. Given a generative process about the event, an expectation can be formed by sampling the generative process for a few times and then form the average over the received outcomes. Returning to the example of a student in an exam situation, the student may have an expectation about the outcome of the exam depending on his judgement about his skill, effort and the harshness of the teacher's grading. Another cognitive process that people regularly embark on is planning. The goal of planning is to find an action plan that realizes a certain goal. This is modelled by sampling the generative process with different setting for the unobserved variables and determine for which setting the goal is reached. In case the student wants to perform well in the exam, then a high skill level and effort make a good performance more likely and the plan could be to increase his studying efforts. The last higher cognitive process that can be modelled in this framework are explanations. Explanations of events try to give a causal answer to the question 'Why did this event happen?'. This can be done by simple inference or counterfactual inference. If the generative process is conditioned on the observed outcome one can infer likely values for the unobserved variables. If the inferred values are differ strongly from one's initial assumptions, then one can argue that these variables caused the event.

The next section introduces the SSB and outlines its relation with attribution theory. Experimental results, moderators and process theories of the SSB are discussed. Finally, a process model of the SSB is presented. In section 5 this model is implemented and evaluated on how well it can simulate experimental data.

3 Self-Serving Bias

The self-serving bias refers to the phenomenon that people tend to attribute success internally and failure externally (Campbell & Sedikides, 1999). A typical situation in which the self-serving bias can be observed is a student getting back exam results (McAllister, 1996). If a student performs well the success is usually attributed to his own efforts and skill while failure is blamed on external factors such as a bad night sleep in the night before the exam or harsh marking. From a more abstract perspective people try to explain an event, where the event is the success or failure in the exam. How people explain events is studied in attribution theory. Therefore, a small overview of basic attribution theories is given before the self-serving bias is looked at in more detail. For the process model of the self-serving bias only a small part of attribution theory is relevant. However, as will become more clear below attribution theory deals with humans trying to explain events. Attribution theory is therefore strongly related to the idea of intuitive theories and generative processes that model how an event comes into being. The next section should illustrate how attribution theory can be used as a theoretical basis to construct intuitive theories for different domains.

3.1 Attribution Theory

Following (Malle, 1999) attribution describes two distinct processes. Firstly explanation of an event, secondly inference about unobserved values. The process of explanation assigns a cause to an event, while the process of inference assigns a value to an unobserved variable. A person saying 'She shouted because the cars driving by were so noisy' explains the event 'shouting' and assigns the cause 'noisy environment' to the event. A person saying 'I believe he is a very kind person' assigns the value 'very kind' to the unobserved personality characteristic kindness. Both forms of attribution are interrelated. To illustrate that, lets look once again at a performance situation. The skill level of a student is for the student himself not directly observable but can only be inferred from diagnostic situations. Such a situation is the performance in an exam. Now the event, failure of exam could be explained by inferring a low skill value. Therefore the process of explanation happens via the process of inference. In the following Heiders theory of attribution (Heider, 1958), Kellys covariation theory (Kelley, 1973) and Malle's folk-conceptual theory of behaviour explanation (Malle, 1999) are explained. Especially Malle's theory of attribution offers a clear connection to intuitive theories.

Heider is often seen as the father of attribution theory (Malle, 1999). He studied intensively how humans perceive other people (Heider, 1958) and is often credited with introducing a person-situation dichotomy into attribution theory. This dichotomy describes how the cause of an event is either attributed to a situational factor or to personality trait of the person involved. The distinction between person and situation is not suitable for all types of attribution. For example an intentional action can never solely be explained by a situational factor since a person had the intention to perform the action. As explained before the self-serving bias also follows this dichotomy by distinguishing between external factors (factors of the situation) and internal factors (factors within the person). Kelly's covariation model (Kelley, 1973) identifies causes to an event by looking with which factors the event covaries. Potential causes for an event are the person, the stimulus and the situation. For example the principle of consistency says that if a behaviour is shown repeatedly across time then the person is a likely cause of that behaviour. However, if people can freely choose which information they need to determine the cause of an event, most people choose information of the underlying mechanism rather than covariation information (Gelman, 1995). Covariation information is also often difficult to obtain for events that rarely occur. Malle's folk conceptual theory of behavior explanation distinguishes between different types of events and different modes of explanation (Malle, 1999). Events are separated into unintentional action, intentional action and outcomes. An unintentional action, such as 'dropped the yoghurt', often has an immediate cause ('his hands were slippery from the sweat') that is taken as an explanation. To explain an intentional action people use two different modes of explanation which Malle referred to as 'reasoning explanation' and 'causal history of reasoning'. Reasoning explanations are based on the desire and belief that motivated a person to perform the intentional

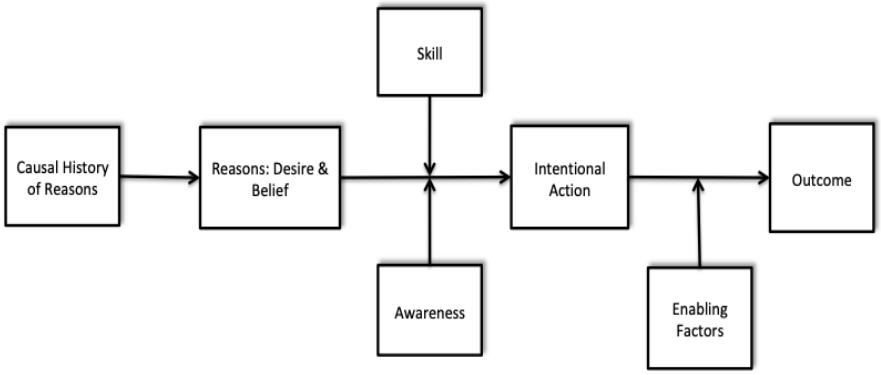

Figure 2. Overview of the different component of Malle's theory on how intentional actions and outcomes are explained.

action. As an example take the sentence 'She ran to the bus station to catch the last bus as she thought walking would not get her there in time.'. The action 'running' is explained in term of the woman's desire to catch the last bus and the belief that walking will not get her there in time. The desire and belief that lead to an intentional action must not always be stated explicitly but can also be implied. In the last example simply saying that 'she run to the bus station to catch the last bust' would be enough to infer that she thought only running would get her there in time. Explanations using the 'causal history of reasoning' are focused on how some beliefs and desires are formed. Possible explanations work via personality traits or culture. Finally, an outcome is explained by the factors that enabled the event to happen, which can be separated into situational factors and personal factors. As an example take the explanation "He scored the freekick, since he practiced shooting all week". The outcome 'scoring the freekick' is explained by the personal action 'practicing all week'.

Two remarks are noteworthy. The event failure or success in a performance situation is an outcome. Therefore, in theory an enabling-factor explanation is used by people to determine the causes of the outcome of a performance event. Further, figure 2 shows an illustration for the mode of explanation that is employed by people to explain an intentional action. Is can be seen as a generative process of the action as discussed in section 2. Therefore Malle's theory offers an intuitive starting point for developing an intuitive theory for explaining intentional action. This could then be used to model human attribution processes for intentional action with the same framework used here to model the self-serving bias.

3.2 Theories of the Self-Serving Bias

Various theories that aim at explaining the self-serving bias exist. Overall one can distinguish between motivational and cognitive explanations. However, the

distinction should not be seen as a way to frame the self-serving bias as a purely motivational or cognitive phenomenon but rather that the self-serving bias is produced by an interaction of motivational and cognitive processes. Shepperd, Malone and Sweeny (2008) give an overview of different mechanisms that possibly explain the self-serving bias. Self-enhancement and self-representation are two motivational processes that can cause people to show self-serving attributions. The idea is that humans aim at maintaining a positive self-worth. Attributing an undesirable event to oneself harms the self-worth, such that the event is attributed to external factors. For an undesirable event to damage the self-worth of someone it must be of importance to the person. The importance of the event has been found to moderate the self-serving bias (Miller, 1976). If the event is important to the person success is attributed more strongly to internal factors and failure is attributed more strongly to external factors. Cross-cultural studies have further emphasized the effect of the relevance of the event to the person as a moderator. In western cultures individual success is highly linked to the self-worth as has been demonstrated in academic context (H. W. Marsh & Yeung, 1997). However, in collectivist cultures the self-worth is more closely linked to the standing of one's own social group. This is reflected in the fact that people from collectivist cultures make more group-serving attributions (Al-Zahrani & Kaplowitz, 1993). Any activity that is aimed at influencing how other people view oneself is also known as impression management. People are motivated to be seen positively by their peers and actively construct a favorable self-image (Leary & Kowalski, 1990). For attribution processes in performance situation there are two conflicting motives that result from the goal of self-representation. If a person takes credit for success this may lead his peers to view him as skilled and therefore more favourably or they could it see as bragging and therefore evaluate him more negatively. Which of these processes outweighs the other, seems to be moderated by social anxiety. People high on social anxiety try to prevent any disapproval from the group. They do not take the risk of disapproval that comes with taking credit for success and therefore show less self-serving bias (Arkin, Appelman, & Burger, 1980). People low on social anxiety try to gain approval and take the risk that comes with taking credit for success.

Next to motivational mechanisms, Shepperd et al. (Shepperd, Malone, & Sweeny, 2008) discuss cognitive processes that have the potential to explain the self-serving bias. In the case of a performance situation people have an expectation of how well they will do. People's expectation are often biased towards positive outcomes (Shepperd et al., 2008). One explanation for that positivity bias is the asymmetrical retrieval of positive and negative past experience. People are more likely to remember positive memories rather than negative ones (L. Marsh, Edginton, Conway, & Loveday, 2019). If an outcome agrees with the expectation then no further search for potential causes is made and the event is simply attributed to the factors that also gave rise to the positive expectation. In case of performance situations, a positive view of one's own skill leads to positive expectations such that a person is likely to attribute a success to their own abilities. However, if the outcome is different from the expectations, one must

look for other causes that were not integrated into the expectation formulation before. These are mostly external causes, leading to the process of attributing failure to external factors. One conclusion from this model is that people that have negative expectation also show less self-serving bias. Evidence for that claim comes from studies on the self-serving bias conducted with participants that suffered from depression. People with depression show less self-serving bias (Greenberg, Pyszczynski, Burling, & Tibbs, 1992).

Another theory focuses on the self-concept of a person (Shepperd et al., 2008). Shavelson, Hubner & Stanton (Shavelson, Hubner, & Stanton, 1976) define the self-concept as 'a person's perception of himself'. Similar to the positivity bias in expectation, there is a positivity bias in the self-concept. People evaluate themselves more positively than the average person, especially on personality traits that are seen as desirable (Alicke, 1985). Outcomes in performance situations either confirm or contradict the self-concept. In case of success an attribution to the self aligns well with a positive self-concept. If the outcome is seen as a failure it contradicts the self-concept, attributing the cause of such failure to external factors enables one to explain the event as well as to continue to maintain a positive self-concept. This view on the self-serving bias also offers another explanation of why depressed people do not exhibit the self-serving bias. Under the assumption that people with symptoms of depression maintain a negative self-concept, failure should be attributed to the self and success to external factors.

A third cognitive process is focused more on biases in the way humans determine causes of events. Biased hypothesis testing (Shepperd et al., 2008) postulates that humans determine causes of phenomena as naive scientists. That is they form a hypothesis and then test this hypothesis using data. However, the way people test an hypothesis is biased. That is they look for information that confirms their hypothesis and are more critical of information that does not fit with their hypothesis (Pyszczynski & Greenberg, 1987). The self-serving bias can now be explained by assuming that people formulate hypothesis that favor themselves. After, failing an exam a student may ask himself 'Am I smart?' instead of asking 'Am I dumb'. Attributing the outcome to external causes confirms the hypothesis and leads to the self-serving bias.

As discussed before all of these processes may interact to produce self-serving attributions in different situations. For example, the role of expectation, self-concept and self-worth are closely connected. A positive expectation for a future event such as an exam may be based on a positive self-concept with respect to one's own ability. This positive self-concept may also be the basis for one's self-worth. An unexpected failure, leads people to question their self-concept which in turn has effects on a persons self-worth.

Campbell and Sedikides (1999) tried to unite past research on the moderators of the self-serving bias under the theoretical framework of self-threat. The idea is based on the fact that self-threat emerges whenever a dimension of the self-concept is challenged that a person deems important to himself. In total fourteen moderators are identified and classified on whether they increase or decrease self-threat. The moderators, their theoretical effect on self-threat and

Table 1

Theoretical and Empirical Effect of Moderators on the self-serving bias

Moderator	Theor. effect on the SSB	Empirical Effect
Task Importance	+	supported
Self-esteem	+	supported
Achievement motivation	+	not supported
Self-focused attention	+	supported
Outcome expectancy's	+	supported
Perceived task difficulty	+	supported
Interpersonal orientation	+	unsupported
Status	equal	supported
Gender	male	supported
Affect	+	supported
Locus of control	external	supported
Task-type (skill/interpersonal)	skill	supported
Role	actor	supported
Task Choice	free choice	not supported

Note. The second column contains the setting of the moderator for which the theory predicts that it enhances the the self-serving bias (SSB). The + stands for higher values.

the empirical evidence is summarized in table 1.

It is worth noting that many of the moderators are not independent but are interrelated. For example, different theories exist on why males exhibit a stronger self-serving bias than women. These theories are based on the idea that males and females differ on other moderators that affect self-serving attributions. One potential explanations is that most tasks in the self-serving bias literature are male oriented. That means males tend to consider these tasks as more important to themselves and via the moderator of task importance the SSB is enhanced. Another explanation is based on the fact that men have higher global self-esteem compared to women (Hayes, Crocker, & Kowalski, 1999). Therefore the stronger self-serving bias can be explained via the moderator self-esteem.

Duval & Silvia (2002) noticed that success attribution and failure attribution differ. Success is consistently attributed internally while failure is sometimes attributed internally and sometimes attributed externally. To explain this empirical fact they proposed a dual process model for the self-serving bias. They argue that humans aim at self-enhancement in case their self-worth is endangered by discrepancies between the self-concept and reality. However, humans also aim at producing accurate attributions, that is given an event human's goal is to find the most plausible cause of that event. Both of these processes can lead to similar or contradicting conclusions. Under the assumption that a person has a positive self-concept any success event agrees with the self-concept. If oneself is also a plausible cause of that event than success will be attributed internally.

Therefore in the case of success, both processes agree on their conclusion. This is reflected by the empirical evidence which shows that success is consistently attributed internally. Take the same situation (positive self-concept and the self as a plausible cause for the event), but let the event be categorized as a failure. The self-enhancement process would attribute the failure to an external cause, but the need for accurate self-assessment would suggest otherwise. Now the question is which process is applied or dominates the other. Duval & Silvia (2002) suggest that the probability of improvement acts as a moderator. That is if the probability of improvement is high then the process of accurate self-assessment takes place since an accurate judgement of the cause could prevent that the failure happens again. If the probability of self-improvement is low then an external attribution is made. Another moderator variable is self-awareness. The self-enhancement process only takes place if the discrepancy between the self-concept and the evidence from the environment is noticed by the person. If self-awareness is low this should not be the case and failure should be attributed to oneself.

3.3 A process model of the Self-Serving Bias

This section focuses on combining the presented theories regarding the self-serving bias and empirical evidence about the self-serving bias to form a process model of the SSB. The structure of the process model is based on the ideas from section 2, that is on expressing cognitive processes as inference mechanisms over intuitive theories.

The first step is to construct an intuitive theory that reflects how a person perceives a performance situation. The intuitive theory must be abstract such that any performance situation can be represented by it. Based on Malle's theory of behaviour explanations an outcome of an action is explained looking at the factors that enabled the event. The enabling factors in performance situations are the skill and effort of a person, luck and external reasons. The concept skill refers to the ability of a person to perform the task at hand. The concept effort describes the investment into preparation and the involvement during performance of the task. Skill is an unobserved latent variable, whereas effort could be more accurately estimated by comparison with other participants in the task, for example other students, taking the exam. Both of these concepts are related to the person and abstract enough such they can be related to any situation. Any task demands a skill to be performed and this skill can often be practiced. The concept external summarizes all relevant factors of a task that are independent of the person but potentially affect the performance. For example, factors that have an influence on the performance of a student in an exam could be the harsh grading of the corrector, the format of questions (Multiple Choice vs Free-form answers) and others. The concept luck summarizes relevant factors for performance that are hard to attribute to any other person. Factors such as the type of questions in an exam can be attributed to the person who designed that exam. The graphical model in figure 3 summarizes the structure of the intuitive theory.

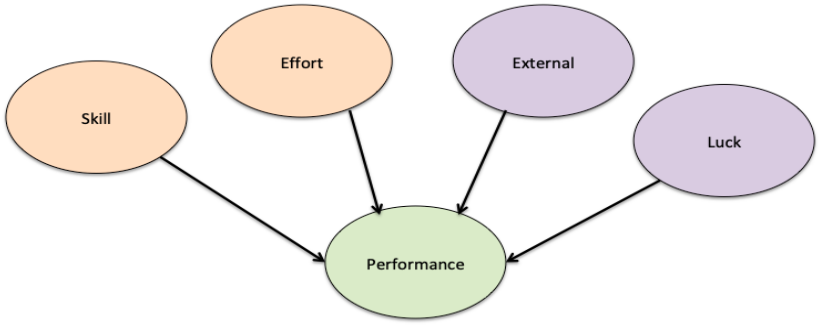

Figure 3. Graphical representation of an intuitive theory of performance situations.

Now the idea is to model the self-serving bias as a process that uses the intuitive theory to attribute the outcome to either any of the external factors (external and luck) or any of the internal factors (skill and effort). The model assumes that a person has an immediate attribution to the internal factors. That is one observes the outcome of the event and draws inference on the values of the internal factors such as skill and effort, while holding all the external factors constant. If self-awareness (or self-focused attention) is low, the results of this immediate attribution are used to identify the cause of the performance outcome. In case self-awareness is high the further inference process depends on how desirable an accurate identification of the true underlying causes of the performance outcomes are. If achievement motivation is high and probability of improvement as well an accurate inference could benefit future results. Therefore, the immediate inference process is used to identify causes for the outcome. In case there is no achievement motivation or probability of improvement the difference between the self-concept which consists of the concepts skill and effort before the task and after the task is compared. If the difference is favorable and no danger for the self-worth is present the inference procedure focuses more on the internal factors as possible causes for the outcome. However, if the self-worth is endangered a different inference procedure is taken. Inference on the contributions of the different causes for the performance outcome are determined, while conditioning on a constant self-worth and the observed outcome. A graphical representation of this inference process can be seen in figure 4.

Not all of the discussed moderators and process mechanisms are included in the presented model. For example, the effect of impression management is ignored. The idea of this thesis is not to have an exhaustive model that considers every possible variable and mechanism but rather to give a proof of concept that empirical results can be simulated via this type of computational model. By restricting the evaluation to experiments that only consider variables and mechanisms that are present in the model, the proof of concept should

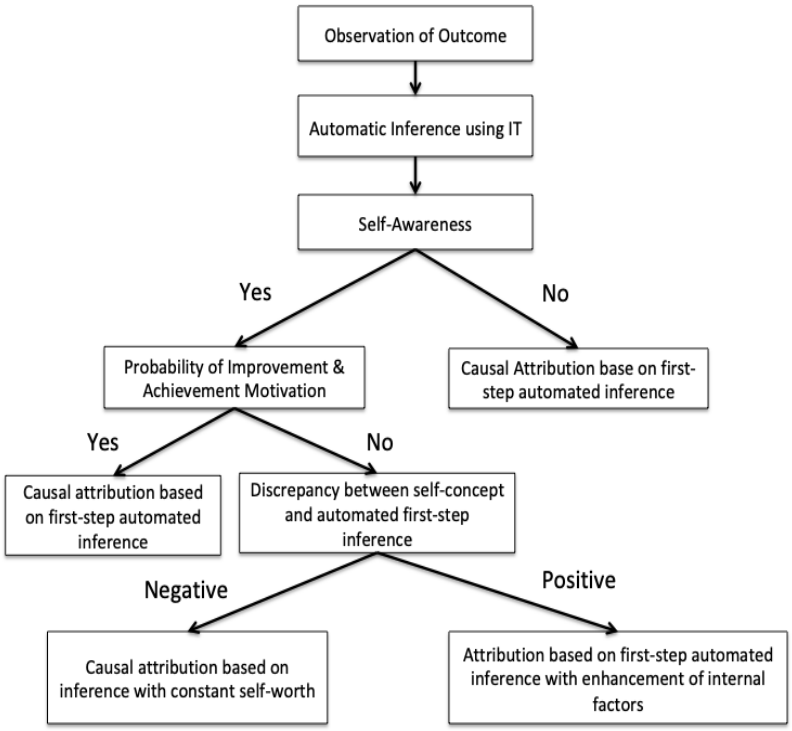

Figure 4. Process model of the self-serving bias represented as a flow chart.

still be possible. Furthermore, the idea is to design a framework that can be extended easily to different process models or extend an existing process model with more moderators. As will become apparent in section 5 one advantage of this approach to modelling cognitive processes could be to include a number of moderator variables in the model that have been shown to have an effect on the self-serving bias, but for which there is no experimental study looking at the interaction of these moderators. After introducing probabilistic graphical models and probabilistic programming this process model of the self-serving bias will be implemented computationally and evaluated using results from past experiments.

4 Probabilistic Programming

To model and possibly understand human inference with intuitive theories we need to be able to denote an intuitive theory and run inference on the theory itself. What are some desiderata for a denotational language for intuitive theories. One must be able to denote arbitrary abstract concepts and their relations. Secondly, the tool to represent an intuitive theory needs to be probabilistic and support probabilistic inference. It needs to be probabilistic since the information humans make use of is often noisy and incomplete. Probabilistic inference is crucial to get estimates of values for unobserved variables. Thirdly, it should support productivity and compositionality. This means that an intuitive theory that describes the generative process of a certain outcome can be used in another intuitive theory that uses this outcome variable. In that sense intuitive theories can be reused and combined.

4.1 Probabilistic Graphical Models

Let $X = (X_1, X_2, ..., X_n)$ be a random vector and P its probability distribution. The idea of probabilistic graphical models is to denote properties of the joint probability density p of the random vector X using graphs. The properties that are encoded using the graph are conditional independence properties. The goal is to estimate the joint probability density p from data. For large n this quickly becomes an intractable problem. Take the case of estimating a probability density p over n binary variables. To fully describe such a density distribution $2^n - 1$ parameters are needed and must also be estimated. Since the number of parameters grows exponentially in the number of variables, to estimate these parameters from data becomes impossible even for small n. Conditional independence assumption can simplify the problem a lot. In the estimation problem above, assume that all random variables are independent. Then the density would factorize to $p = \prod_{i=1}^{n} p_i$ which is fully specified only with $n - 1$ parameters. Depending on the type of graphs (directed, undirected, factor graphs) used to encode the probability distribution those models have different names. Here, the focus is on directed graphs. The corresponding probabilistic graphical models are called Bayesian networks. Before the definition of a Bayesian

network is given, the notion of a directed graph is introduced.

Definition 1 (Directed Graph) *A finite directed graph $G = (V, E)$ is a tuple consisting of a set of vertices V and a set of edges $E \subseteq V \times V$. The graph is denoted by drawing a node for each vertice $v \in V$ and a directed edge pointing from node i to node j for each element $e \in E$. A directed path in a directed graph is a sequence $(v_1, e_1, v_2, ..., v_n)$ such that e_i is an edge between v_i and v_{i-1}. A directed graph is called acylic if it does not contain any cycles, i.e. there is no directed path such that the starting and end node are the same.*

Definition 2 (Bayesian network) *A Bayesian network is a tuple (X, D) where X is a random vector with joint probability density p and D is an acyclic graph. Each vertice of D represents one random variable from the random vector. To be a Bayesian network the tuple must satisfy the Markov factorization property:*

$$p(x) = \prod_{i=1}^{n} p(x_i | x_{p(i)}),$$

where $x_{p(i)}$ denotes the random variables that are the parents of node i in D. This means that the random variable i is independent of all other random variables given the parents of node i.

Our next aim is to denote the intuitive theory of performance outcomes presented in section 3.3 as a Bayesian network. For a concise representation the following abbreviations are used: E=Effort, S=Skill, Ext=External, L=Luck, P=Performance. The random vector that captures all variables of interest is denoted with $X = (X_S, X_E, X_{Ext}, X_L, X_P)$ and following definition 2 the joint density of X can be written as

$$p(x_S, x_E, x_{Ext}, x_L, x_P) = p(X_S)p(X_E)p(X_{Ext})p(X_L)p(X_P | x_E, x_{Ext}, x_S, x_L).$$

One question that emerges is how each of the individual distributions of the different concepts and the conditional distribution look like. Different choices for possible distributions are discussed in section 5. The intuitive theory here is a very simple example of a Bayesian network. There is a large literature on properties of Bayesian networks and inference algorithms that are especially designed for Bayesian networks. Most of them as well as other information on probabilistic graphical models can be found in Koller & Friedmann (Koller & Friedman, 2009).

The next section is about Bayesian inference in general and in particular Likelihood-Weighting as a general Bayesian inference procedure. Using a general inference procedure instead of an algorithm specifically designed to be applied to Bayesian networks such as variable elimination or belief propagation offers the advantage that the framework can be extended to more complex intuitive theories. For example, an intuitive theory in which the concepts of skill and effort can themselves be expressed as outcomes of intuitive theories or scenarios

in which the intuitive theory contains conditional-statements. These extensions would render traditional inference methods useless. Additionally, Likelihood Weighting is easy to implement using the probabilistic programming language Pyro (Bingham et al., 2019).

4.2 Bayesian inference and Likelihood Weighting

The basic modelling idea of Bayesian statistics is to update beliefs about latent variables given observations. Let $X = (X_{obs}, X_{lat})$ be a random vector that consist of random variables whose values are observed and latent random variables. Bayes rule simply gives us an expression for the probability density over the latent variables conditioned on the observed variables

$$p(x_{lat}|x_{obs}) = \frac{p(x_{obs}|x_{lat}) * p(x_{lat})}{p(x_{obs})}.$$

Here $p(x_{lat}|x_{obs})$ is usually referred to as the posterior distribution, $p(x_{obs}|x_{lat})$ as the likelihood and $p(x_{lat})$ as the prior. The quantity $p(x_{obs})$ is a normalization constant that has no effect on the distribution over the latent variables and is usually referred to as the evidence. The goal of Bayesian inference is to describe the posterior distribution under some model for the likelihood and the prior distribution. This problem can only be solved analytically in the case of conjugate priors. These are cases in which the posterior distribution has the same distribution as the prior distribution, e.g. Beta-Binomial model. There are are a range of methods to do Bayesian inference such as Markov-Chain Monte-Carlo methods, Laplace Approximation, Sampling methods (Rejection Sampling, Importance Sampling) and stochastic variational inference (SVI). These methods differ in their applicability, simplicity and efficiency. Here, the focus lies on likelihood weighting which is a form of importance sampling. The general idea of importance sampling is to exchange the posterior distribution with a distribution from which one can sample. Likelihood weighting can be used to determine the expectation of the latent variables under the posterior distribution.

Given a distribution q over the real random vector X of size n and a function $f : \mathbb{R}^n \to \mathbb{R}$ a Monte Carlo estimate with sample size K of the expectation of $f(X)$ under q is given by

$$\mathbb{E}_q[f(X)] \approx \frac{1}{K} \sum_{k=1}^{K} f(x_k), \text{where} \quad x_k \sim q.$$

The idea of importance sampling is to compute a Monte Carlo estimate but instead of sampling from the unknown posterior distribution p we sample from a proposal distribution q:

$$
\begin{aligned}
\mathbb{E}_{p(X|Y)}[f(X)] &= \mathbb{E}_q[\frac{p(X|Y)}{q(X)} f(X)] \\
&\approx \frac{1}{K} \sum_{k=1}^{K} \frac{p(x_k|Y)}{q(x_k)} f(x_k) \\
&= \frac{1}{K} \sum_{k=1}^{K} \frac{p(Y|x_k) * p(x_k)}{p(Y) q(x_k)} f(x_k), \quad x_k \sim q,
\end{aligned} \tag{1}
$$

where in the last step Bayes formula is used. The fractions $\frac{p(x_k|Y)}{q(x_k)}$ are called likelihood weights. An immediate question arising from equation 1 is what a good proposal distribution q might be. First, for the term in equation 1 to be defined the distribution p needs to be dominated by q, i.e. p is 0 at every point x for which $q(x) = 0$. Also q should be large where the posterior p is large to prevent enormous likelihood weights. One idea is to simply take the prior distribution over the latent variables. This simplifies the last term in equation 1 to $\frac{1}{K} \sum_{k=1}^{K} \frac{p(Y|x_k)}{p(Y)} f(x_k)$. The likelihood term $p(Y|x_k)$ can be calculated, whereas the evidence term $p(Y)$ needs to be estimated as well. This estimation can be done again via importance sampling

$$
\begin{aligned}
p(Y) &= \mathbb{E}_{p(X)}[p(Y|X)] \\
&= \mathbb{E}_{q(X)}[\frac{p(Y|X)p(X)}{q(X)}] \\
&\approx \frac{1}{K} \sum_{k=1}^{K} \frac{p(Y|x_k)p(x_k)}{q(x_k)} \\
&= \frac{1}{K} \sum_{k=1}^{K} p(Y|x_k), x_k \ \sim q,
\end{aligned} \tag{2}
$$

where in the last line the proposal equation q was replaced by the prior p. Combining equation 1 and 2 gives us the following likelihood weighting estimate of the expectation under the posterior

$$
\mathbb{E}_{p(X|Y)}[f(X)] = \frac{1}{K} \sum_{l=1}^{L} \frac{p(Y|x_l)}{\sum_{k=1}^{K} p(Y|x_k)} f(x_l).
$$

Taking a closer look the estimate is simply a weighted mean of the samples where the weight of a sample is determined by its likelihood normalized by the sum of all likelihoods. Likelihood weighting works well if the number of random variables whose joint probability density is modelled is small. If there are too many random variables the number of samples needed until one samples a configuration of random variables that has a probability not close to zero is too large to be efficient. The only hyperparameter for the inference method is the number of samples L that are used to compute the Monte Carlo estimate.

```python
# Model that cannot be compiled into a graphical model
import pyro
import pyro.distributions as pyrd

def poisson_model(lam):
    '''
    lam: Parameter of the Poisson distribution (must be positive)
    '''
    assert lam>0, 'Poisson parameter must be positive'
    n = pyro.sample('number',pyrd.Poisson(lam))
    xs = []
    for i in range(int(n.item())):
        xs.append(pyro.sample(f'sample_{i}',pyrd.Normal(0,1)))
    return sum(xs)
```

Figure 5. A simple model that cannot be compiled into a graphical model before execution. The number of random variables contained in the graphical model is equal to the sampled valued of the poisson random variable, which is only known at runtime of the function.

A larger sample size leads to more precise estimation. Next, probabilistic programming and its realization in Pyro are introduced. Based on this introduction a realization of the likelihood weighting procedure in Pyro is presented.

4.3 Probabilistic Programming with Pyro

Probabilistic programming uses the expressive power of programming languages to denote probabilistic models and perform inference on these models. It has raised interest for its ability to denote any probabilistic model and automate Bayesian inference on these types of models (van de Meent, Paige, Yang, & Wood, 2018). Most of the high-level programming language offer packages to sample from different distributions. The key difference between a high-level programming language such as python and its extension to a probabilistic programming language in form of Pyro (Bingham et al., 2019) is the ability to combine different sampling statements into a model and offer methods to perform inference on a conditioned form of the model. There are different possibilities to realize a probabilistic programming language. This has lead to the emergence a variety of probabilistic programming languages. One class of languages, of which STAN (Carpenter et al., 2017) is the most prominent, compiles any model code into a probabilistic graphical model and applies known inference algorithms for probabilistic graphical models to the compiled model. This restricts the class of models that can be denoted but guarantees fast inference. One simple model that cannot be expressed by such a language is denoted in Pyro in figure 5.

First a Poisson variable is sampled and the value of this sample determines the number of times a normally distributed random variable is sampled. Then

the sum of these normally distributed random variables is returned. Since the number of random variables (or nodes in the corresponding graphical model) is not known at compilation time it cannot be compiled into a graphical model. Other approaches try to make use of the expressiveness of general purpose programming languages such as python. This makes efficient inference more difficult. There are multiple ways of implementing inference for a model that is denoted in a general-purpose programming language. The architectural choices surround two questions: (1) How is the model represented (2) How is the model representation used to do inference. Here, the focus will be on the design of 'Pyro' (Bingham et al., 2019), which is an extension of the general purpose language python based on the deep learning framework 'PyTorch' (Paszke et al., 2019).

In Pyro any model is denoted as a stochastic function. That is a function that contains sample statements. To be more precise any callable, that is any object that implements the '__call__'-method, can be seen as a model. To sample from a probability distribution the pyro.sample function must be used. Each sample must have a unique name. The sampling happens from a distribution object from Pyro. These distributions are just wrappers around the distribution objects from PyTorch. Figure 5 displays one example of such a model. To later test the implementation of the likelihood weighting inference algorithm the beta-binomial model is introduced and implemented as a stochastic function in Python. For the beta-binomial model the number of successes from a binomial experiment with n trials is observed. The goal is to perform inference on the probability of success θ for the binomial experiment. A popular Bayesian modelling approach is to model the success probability θ with a beta distribution. Then the posterior distribution is again a beta-distribution and takes the following form:

$$\begin{gathered} \text{Model:} \\ \theta \sim Beta(a, b) \text{ (prior)} \\ X|\theta \sim Ber(\theta) \text{ (likelihood)} \\ \theta|X \sim Beta(a + x, b + n - x) \text{ (posterior)}, \end{gathered} \tag{3}$$

where x represents the number of successes. The code in figure 6 realizes the model in Pyro. By running the model several times one can obtain a prior distribution over the return variables (figure 7).

The key to condition the model on observations and to perform inference in Pyro is the effect handling library poutine. Pyro has a gloabl stack called PYRO_STACK which collects all effect handling classes that are created during the program. In python every object that implements an __enter__ and an __exit__ method is an environment object. The keyword with can be used to enter an environment. The syntax construct with Class() as v creates an instance of the class, passes it to the variable v and calls the enter method of the object. All effect handling classes implement these methods. The idea is to execute a model in an environment that changes the execution of the model and collects information necessary to perform inference algorithms. For the purpose of creating a likelihood weighting inference algorithm it is enough to know the effect handling

```python
# Beta-Binomial model
import torch
import pyro
def betabinomial(a,b,n):
    a = torch.tensor(a,dtype=torch.float)
    b = torch.tensor(b,dtype=torch.float)
    theta = pyro.sample('theta',pyro.distributions.Beta(a,b))
    x = pyro.sample('successes',pyro.distributions.Binomial(n,theta))
    # Any sample statement returns a torch tensor
    # For one-dimensional torch tensor
    # the item() function returns the value of the tensor
    return int(x.item())
```

Figure 6. A realization of the beta-binomial model in Pyro.

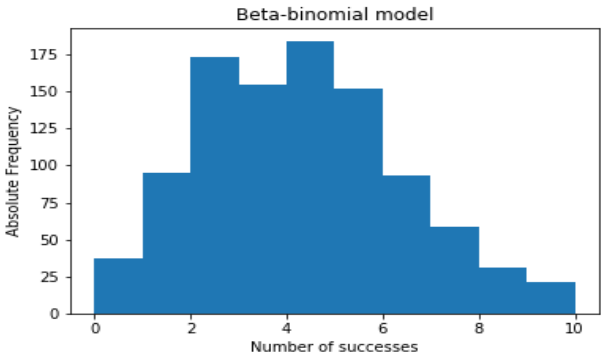

Figure 7. Histogram over the number of successes in a beta-binomial model. The prior distribution of the success probability is Beta(3,5) and the number of trials is 10. The model is run a 1000 times.

classes Condition and Trace. To understand the effect these classes have on the execution of a model, one must have closer look at the inner workings of the pyro.sample function. Instead of just returning a sample from the distribution object that is passed as a parameter it creates a message data structure. This message is a python-dictionary which contains information such as the name of the sample, the distribution object and a boolean variable that indicates whether the variable is observed or not. If the PYRO_STACK is not empty each effect handling class of the stack is applied to the message. The value returned by the sample statement is the value element of the message-dictionary.

The effect handling class Condition always gets passed a dictionary with observations. The dictionary contains the sample names of the observed sample statements as keys and the observations as values. If an instance of the effect handling class Condition is now applied to the message of a sample statement,

```
cond_model = pyro.condition(betabinomial, \
             data={'successes': torch.tensor(4,dtype=torch.float)})
cond_model(3,5,10) # returns 4
```

Figure 8. The beta-binomial model is conditioned on observing four successes. Running the conditioned model will always return four.

it is checked whether the name of the sample statement can be found in the dictionary containing the observations. If that is the case then the boolean indicating whether the sample statement is observed is set to true and the value of the message is changed to the value of the observation. If the name cannot be found the value is sampled normally from the distribution object. Conditioning the beta-binomial model from figure 6 on an observed number of successes and running the conditioned model always returns the observed value as an outcome, independent of any of the input parameters (figure 8).

The Trace class constructs a data structure to represent a directed graph that captures the corresponding execution trace for one run of the model. The data structure is named trace and collects necessary information to be able to apply different inference algorithms. For example, it can be used to calculate all log-probabilities of the sampled values in an execution trace. Nodes can be accessed by the names of the corresponding sample statements and groups of nodes such as all observational nodes can also be accessed. The data structure is built up newly with each run of the model.

4.4 Likelihood Weighting inference with Pyro

The class LW (appendix A) can be used to perform inference on any arbitrary stochastic model. It is designed to register a model and a dictionary of functions during instantiation. Then for arbitrary sample sizes and arbitrary observations likelihood weighting inference on the conditioned model can be performed using the inferLW method. To accomplish that, the model is first conditioned on the observations. Then for each of the L samples the conditioned model is executed and the trace data structure recorded. From this data structure the log-probabilities of the observed random variables are extracted and the sampled values for the unobserved variables are recorded. These information are then combined to produce an estimate of $E[f(X)]$ for each function f that is passed at time of instantiation of an instance of the LW class. The code for the implementation can be found in appendix A.

To test the implementation of the inference algorithm, it is evaluated on the beta-binomial model introduced above (equation 2). Analytically given values for the parameters a and b and an observed number of successes x in n trials, the posterior distribution is again a beta-distribution with parameters $a + x$ and $b + n - x$. The expectation of the posterior is therefore $\frac{1}{(1+\frac{b+n-x}{a+x})}$. The likelihood weighting algorithm is tested for $L = 100, 1000, 10000$. The results

Table 2

Evaluation of the likelihood weighting inference algorithm in the beta-binomial model for different sample sizes

L	analyt. post. prob. of success	estimated post. prob. of success
100	0.389	0.377
1000	0.389	0.386
10000	0.389	0.388

Note. L is the sample size for the inference algorithm.

can be seen in table 2 and the code for generating the experiments can be found in appendix A.

One can see that a reasonable precision can already be achieved with a sample size of a 100 and that precision is improving with sample size. In the next section a computational model of the process model of the SSB from section 3.3 is implemented using Pyro.

5 Computational model of the Self-Serving Bias

This section first introduces a computational realization of the process model of the SSB discussed in section 3.4. The goal is to realize the model as part of a framework that allows to flexibly explore other process models or different realizations of the same process model. An additional goal is to design the framework such that any cognitive process framed as an inference process over an intuitive theory can be implemented computationally.

5.1 Computational framework

The framework is structured into three classes and their implementation can be found in appendix B. The first class is called Variable and is designed to represent concepts that can be part of an intuitive theory. Instances of the Variable class have attributes that characterize the concept. The attributes are the name of the concept, whether it is an external (part of the situation) or internal (part of the person) concept, and a description of the distribution that the concept follows. The class Variable has a class attribute that holds a dictionary with different possible distributions. The parameters of the distribution must also be passed at the time of initialization. The method sample updates the current value by sampling from the distribution and is called for the first time at the time of initialization. The method return_dist returns the current distribution object which is necessary, so that it can be passed as an argument to the pyro.sample function.

The intuitive theory of a performance situation is implemented as a stochastic function called intuitive_theory (appendix B). It takes the concepts that are

not determined by any other concepts as parameters and returns a dictionary with all the sampled values and their corresponding names. The parameters of the function are instances of the Variable class and represent the concepts skill, effort, external and luck. The stochastic function implements the generative story that is described by the intuitive theory described in section 3.3. First, values for the concepts skill, effort, external and luck are sampled. These then influence whether the performance is a success or not. Two key modelling decision must be made to implement the intuitive theory. What distributions do the concepts follow ? How is is the conditional distribution implemented, that is how do the values of skill, effort, external and luck influence how likely success is. Each variable but success is implemented as a standard normal distribution. This has the advantage that all concepts have the same scale, meaning that high values for skill and effort correspond to similar numerical values. This simplifies the weighting of these variables in further calculations. Other choices are possible. For example, the variance of the distribution of a concept could describe as how amenable a certain concept is perceived by a person. It could well be that people perceive skill to be more fixed than effort. Success is implemented as a Bernoulli random variable. The idea is that higher values of the variables skill, effort, external, and luck should make success more likely. However, the probability of success must lie between 0 and 1. Applying the sigmoid function to the sum of skill, effort, external and luck gives the correct scaling and the effect that higher values lead to higher probabilities. As already mentioned earlier many of the design choice are based on intuition rather than theory. This is actually not uncommon in machine learning. For example most innovation for training deeper neural network architectures such as batch normalization layers (Ioffe & Szegedy, 2015) have been empirically proven to be useful, without having any sound theoretical guarantees. If the design solely follows intuition, experience and a good evaluation metric can guide such design decisions. Since this framework has not been used or developed in similar ways before, prior experience is of no help. As will be seen below it is also not easy to find and make use of a good evaluation metric.

In the next step a class is introduced that represents one participant in an experiment. That class is called Human. The class is build around the idea that all relevant variables that describe the participant and the experimental situation are contained in a dictionary. At initilization time an instance of the Human class receives an intuitive theory and attributes that contain lists of variables names. For example the attribute self-concept contains all variable names that are relevant for the self-concept of the participant. A dictionary containing all relevant variables can be passed to an instance of the class Human via the set_variables method. The actual process model of the self-serving bias is implemented in the inference method of the class and follows the steps displayed in figure 4. Once again certain modelling decisions must be made based on intuition and not guided by theory. One question concerns the method, by which the inferred values of the latent concepts are used to determine to which causes the participant attributes the performance. Here, the change in the mean of the distributions for the internal and external variables is taken. The main

reason for this choice is that the change in mean gives a rough intuition how strongly the previous assumption of the value of the variable disagrees with the observation.

To test how well the computational model captures the SSB, the aim is to replicate experimental results. To simplify the setting up of an experiment the class Experiment is implemented. At time of initialisation the Experiment class is passed an instance of the Human class, a name and a dictionary containing the base settings for all the relevant variables. An experimental condition can be described via a dictionary. It contains the number of participants in the condition, the name of the condition and the setting of the manipulated variables. A condition can be added to an experiment via the register_condition method. The method run simply runs each condition and saves the results in a dictionary. The average z-score for each condition can be displayed using the method plot_results.

5.2 Simulation of Experiments

For the process model and the whole framework to make sense it must be able to replicate the self-serving bias. In an experiment with two conditions, where one group observes a success and the other one a failure, participants in the success-group should show more internal attributions compared to people in the failure-group. To test this an experiment is set up with 50 participants in each condition. From the implementation perspective an instance of the Experiment class is created. The experiment is named and gets an instance of the Human class and a dictionary containing the base settings of the experiment variables passed as parameters. If the variables probability of improvement and self-awareness are not directly manipulated it is assumed that people are self-aware and do not see any probability of improvement. Therefore, the normal distribution of the concept self-awareness has a 'positive-bias', that is the mean is 0.5 and the normal distribution of the concept probability of improvement has a negative bias, that is the mean is -0.5. If the task importance is not actively manipulated it is set to one. This has the effect that any difference between the self-concept and what inference suggests does not get amplified or attenuated. Two conditions are registered with the experiment; one for each possible outcome. The results can be seen in figure 9 and the code for the implementation is show in appendix C. In case of failure participants make more external attributions than participants that observe a success.

As a second test for the process model, it is checked whether well-established moderator effects can be qualitatively replicated. The meta-analysis by Campbell and Sedikides (1999) illustrates that high task importance leads to a stronger self-serving bias. To test this a variable 'TI' capturing the task-importance is manipulated in two conditions. The simulations assumes there are 50 participants for each condition. The experiment has four conditions since the design is 2×2, the first factor being the outcome (success/failure) and the second factor being the task importance (low/high). The implementation of the experiment is displayed in appendix C. Figure 10 shows the mean z-scores for each condition.

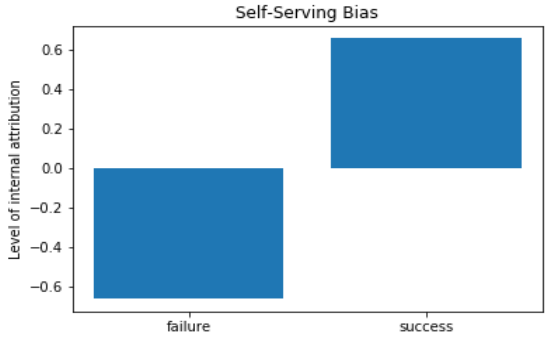

Figure 9. The bar plot displays the level of internal attribution in case participants observed a success or a failure.

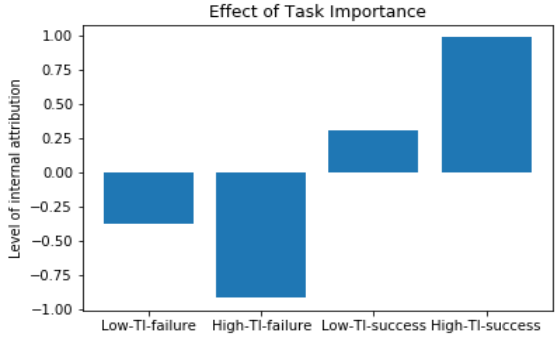

Figure 10. The boxplot displays the effect of task importance (TI) on the level of internal attributions

One can see that the self-serving bias is enhanced if the task-importance is high. In case of high task importance failure is attributed more strongly to external factors and success more strongly to internal factors. In the same manner it is tested whether having a high self-esteem enhances the self-serving bias. A high self-esteem is implemented by letting the skill and effort variable have a positive bias, that is a mean of 0.5. In contrast, a low self-esteem is implemented by letting the skill and effort variable have a negative bias. As before there are four conditions. The results can be seen in figure 11.

The simulation does not replicate a stronger self-serving bias for people with higher self-esteem perfectly. Failure is attributed more strongly to external causes for people with high self-esteem but success is attributed less to the self compared to people with low self-esteem. A possible reason could be that a high self-esteem is implemented as participants having a high trust in their skills, so

Figure 11. The boxplot displays the effect of self-esteem (SE) on the level of internal attributions

people with a high self-esteem that observe a success do not see it as unexpected compared to people with low-self-esteem. This domain specific trust in ones skill could have a qualitative different effect on attribution compared to global self-esteem. An experiment trying to distinguish the effects of local self-esteem and global self-esteem on the self-serving bias could give further insight into whether a distinction must be made between those concepts.

In the following two experiments and their translation into the computational framework are described. The simulation is evaluated on whether its results follow the same qualitative behaviour as of the experimental results. On a program level each experiment is implemented in the same manner as the experiments before. For each condition the number of participants is the same as in the experiments.

To test their dual process model Duval & Silvia (2002) let participants believe that they were taking part in a test that would assess their quantitative judgement skills. The experiment has a 2×2 design. The factors varied are the self-awareness of participants and the probability of improvement. The outcome of the performance is always failure. The probability of improvement and the test result are returned at the end of the test and had nothing to do with the actual performance of the participants. In the high self-awareness condition participants were told that their were filmed during the test for the purpose of evaluation of the testing conditions. Participants attribution was assessed by asking how much they thought they were responsible for the event (internal attribution) and how much they thought external factors were responsible. A final attribution score was determined by subtracting the external attribution score from the internal attribution score. The z-scores of the attribution score and the corresponding results of the simulation can be seen in table 3. A graphical comparison is displayed in figure 12

The simulation can replicate the general tendencies. That is participants that perceive a high probability of improvement and have high self-awareness

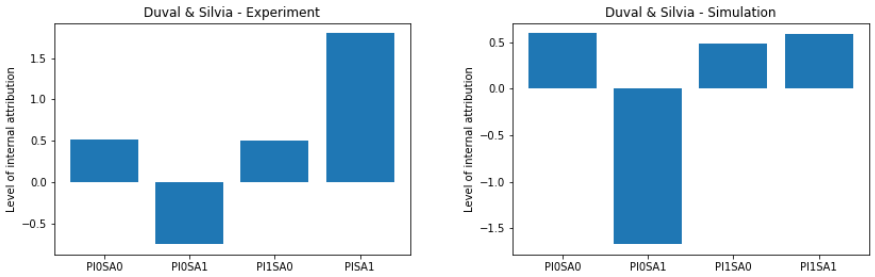

Figure 12. The left boxplot graph shows the z-scores for the internal attributions in the experiment conducted by Duval & Silvia (2002). The right boxplot shows the corresponding z-scores produced by the simulation.

Table 3

Experimental results and simulation results on the effects of probability of improvement and self-awareness on the self-serving bias

Model	Conditions	Attribution to self	Simulation
Duval and Silva (2002)	High-SA, High-PI	1.8	0.54
	Low-SA, High-PI	0.5	0.57
	High-SA, Low-PI	-0.75	-1.66
	Low-SA, Low-PI	0.49	0.55

Note. The presented attribution scores are z-scores. Higher values represent stronger internal attribution.

attribute the failure internally. However, the simulation predicts a much weaker internal attribution than observed in the experiment. If the probability of improvement is low participants attribute the failure to external causes. The external attribution in the simulation is stronger than observed in the experiment. In case there is no self-awareness participants attribute the outcome internally to a similar degree, independent of the probability of improvement. One problem with evaluating the process model using past experimental results is that it is not ensured that the effects that are found in the study can be replicated. That is, if another experiment tries to replicate the results of the experiment from Duval & Silvia (2002), it could be the case that especially the quantitative relations between the experimental groups change. Therefore it is difficult to judge whether the computational model is wrong. Only if more experimental data is gathered which supports the claim that internal attribution in the case of high self-awareness and high probability of improvement are far higher then under the conditions of no self-awareness, one can argue that the process model

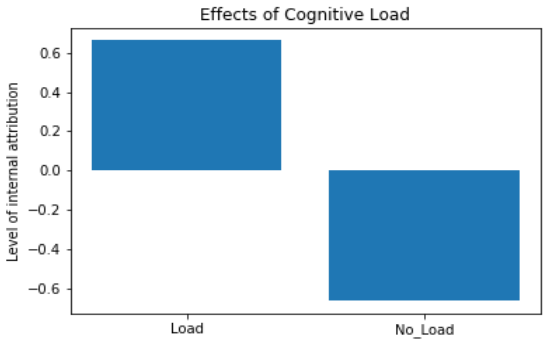

Figure 13. The boxplot displays the effect of cognitive load on the level of internal attributions in the case of failure.

does not capture the self-serving bias correctly.

Based on the results of Duval & Silvia (2002) self-awareness is a crucial component for noting differences between performance feedback and the self-concept. Since introspection is an active process it takes cognitive resources. It can be expected that a high cognitive load prevents a comparison process from happening and that it has the same effect as missing self-awareness. A study that investigates the effect of cognitive load on the self-serving bias is conducted by Sakaki Murayama (2013). After working on a matrix completion task people answered a questionnaire regarding the attribution of failure under two conditions. In the 'no-load' condition participants just answered the questionnaire, whereas participants in the 'load' condition had to complete a sound task at the same time. The studies reported attribution scores with respect to the ability and the task. Unfortunately not enough information is reported to determine the z-score for a one-dimensional attribution score. The hypothesis that self-serving attributions are significantly stronger under cognitive load is confirmed. The simulation replicates the effect as can be seen in figure 13.

To conclude, one can say that the computational model of the process model shows the self-serving bias, moderator effects and can simulate experimental results. The section has also illustrated how difficult it is to evaluate the computational framework. First of all it is not clear, when the simulation has accurately reproduced experimental results. Must the z-scores exactly match or must only the relations between the z-scores be the same as in the experiment? Another problem is that it is not always clear why the computational model fails to reproduce results. Is it due to the exact implementation or due to the underlying process model. Does changing the parameters of the distribution or even the type of the distribution help? One way to make the model more flexible would be to parameterize it and try to optimize parameters to fit to the experimental data. For example the weighting of internal and external factors in the final attribution score could be parameterized. This also bears the risk of making

the model too flexible such that any experimental results can be fitted. The goal of the approach should be to offer insights in the true underlying cognitive process that causes the self-serving bias. The lack of a good evaluation metric regarding the presented framework makes this difficult.

6 Discussion

The thesis explores a new framework for building computational models of cognitive processes. The framework is based on the idea that cognitive processes can be framed as inference processes over intuitive theories. It is tested by introducing a new process model of the self-serving bias and implementing it using the probabilistic programming language Pyro. The model is able to simulate the main self-serving bias, moderator effects and experimental results.

Before discussing what modelling decision could have been made differently or what conclusions one can draw about the introduced process model of the self-serving bias, it is worth looking at what the approach has to offer for the subjects cognitive psychology and artificial intelligence.

As illustrated in the introduction there has been tremendous progress in many subdisciplines of AI such as natural language processing and computer vision by scaling deep learning to massive datasets using enormous computational resources. However, there has been made little progress towards creating models that learn and reason as flexibly as humans do. One key idea that has been put forward to tackle this problem is to have a model that learns disentangled representations of concepts that can be arbitrarily combined. This has the advantage that a model can generalize to unseen situations combining the learned concepts in new ways. Several recent approaches have worked with this idea to create models that can reason over visual input (Hudson & Manning, 2019; Mao, Gan, Kohli, Tenenbaum, & Wu, 2019). However, the concepts extracted from language and visual input are often not connected in any meaningful way. One of the key ideas of modelling with intuitive theories is that the concepts are related by strong principles. This enables people to learn from only a few examples and perform inference even in unseen situations. The framework of denoting an intuitive theory and performing inference over it using techniques from probabilisitic programming seems like a fruitful approach to tackle the problem of human-level learning, generalization and inference. Still, there remain many challenges to advance the approach to a level such that it can be useful. At the moment the intuitive theory must be designed by hand, it would be preferable to learn the intuitive theory from data. This is challenging on multiple levels. The model would need to identify latent concepts and learn the relations between those concepts from perceptual data that may not even contain these concepts. By simply observing different people in a performance situation the model would need to learn some form of representation of the latent concepts skill, effort and luck and their influence on the performance of a person. With current methods this seems rather ambitious.

What does the framework has to offer for cognitive psychology or psychology

in general? On the first sight not much. The framework is designed with the idea that different models of cognitive processes can be evaluated against each other and new process models can be tested before implementing a time and cost intensive experiment. The main problem is that there is no obvious way of determining which process model performs well or when one model performs better than another. A good evaluation metric is needed and a way to distinguish between effects that are caused by a certain implementation choice and effects that are caused by a choice in the design of the process model.

Despite that the framework offers a way to unite past research. In psychology in general but especially in cognitive psychology research often involves studying a certain effect and its moderators. The effect often describes a noticeable or surprising phenomenon or a bias. For example the self-serving bias has been studied for more than 50 years since its introduction by Heider (1958). Many moderator variables have been studied and a couple of process theories have been proposed. However, connections between moderators and also theories are rarely studied. For example, in the case of the self-serving bias the variables achievement motivation and probability of improvement have been suggested to moderate the self-serving bias, but both work via the same process, that is they increase the desire for accurate inference. By implementing the computational model one may realize that two theories actually do not differ much in their underlying mechanisms and only give different names to the same processes. Furthermore, the computational model forces the scientist to be explicit in stating how the key variables interact and by that may open up new questions that could be interesting to answer.

Another question that has not been answered in this thesis is how the framework compares to other approaches that try to uncover relations between variables from data such as structural equation models or other approaches that try to summarize research such as meta-analyses. To summarize, the thesis presents a first attempt at introducing a new framework for modelling cognitive processes. Whether it is a fruitful research direction to pursue, has yet to be shown.

Appendices

A Likelihood Weighting Inference

This section contains the code for the class that implements the Likelihood-Weighting inference algorithm.

```
### Inference with Likelihood Weighting

from collections import defaultdict
from numpy import np
from pyro import poutine


class LW():
    '''
    A class to register a stochastic function and
    given some observations perform inference
    using Likelihood-Weighting
    '''

    def __init__(self,model,observation,f = {}):
        '''
        model: a stochastic function,
               with pyro.sample() statements

        observation: a dictionary with the name
                     of the variables observed as key and the
                     values of the observation as values.

        f: a dictionary of functions for which E[f(X)]
           is determined using Likelihood-Weigthing, here
           X is an unobservable
        '''

        self.model = model
        self.observation = observation
        self.f = f


    def inferLW(self,L,*args,**kwargs):
        '''
        L: the number of samples to draw
           to perform Likelihood-Weighting-Inference.
        *args/**kwargs: arguments and key-word arguments
```

```
                necessary to execute the generative process.
'''

# Condition the model on the given observations
cond_model = pyro.condition(self.model,data = self.observation)
# the sum of the log-probabilities
# of a draw from the conditioned model
logWs = []
# E[f(X)] is saved in estimates for each unobserved variable
estimates = defaultdict(dict)
# The sampled values of the unobserved variables
# are stored in samples
samples = defaultdict(list)
# To obtain the likelihood weights of one sample
# run the conditioned model.
# Determine the probability of the observations
# under the sampled values
# Save the sampled values of the unobserved variables
for _ in range(L):
    # Run the conditioned model
    # and construct the directed graphical model
    trace_DS = poutine.trace(cond_model) \
                .get_trace(*args,**kwargs)
    # Dertermine the log-probability of each sampled value
    trace_DS.log_prob_sum()
    # Get the observational nodes from the data structure
    obs_nodes = trace_DS.observation_nodes
    # For each observational node extract
    # the log-probabilities of the sampled values
    # Sum those values up to obtain
    # the log probability of the observed value
    logWs.append(sum([trace_DS.nodes[elem]['log_prob_sum'] \
                      for elem in obs_nodes]))
    # Add the sampled values
    # for each unobserved variable to a list
    __ = [samples[elem].append(trace_DS.nodes[elem]['value']) \
          for elem in trace_DS.stochastic_nodes]
# obtain the likelihood ratios
# by summing all observed likelihoods
# and then normalize the observed likelihoods by this sum
# need likelihoods not log-likelihoods
logWs = [np.exp(elem) for elem in logWs]
logW_sum= sum(logWs)
weights = np.array(logWs)/logW_sum.item()
# Determine the estimates of
# the expectations for each function f of the class
```

```
        for key in samples.keys():
            for name,elem in self.f.items():
                estimates[key][name]= \
                sum(elem(np.array(samples[key])) * weights)
        return estimates, logW_sum
```

The following code is used to test the implemented Likelihood-Weighting inference algorithm.

```
L = 100 # 1000, 10000
a = 3 # parameter of the prior
b = 5 # parameter of the posterior
n = 10 # number of trials
x = 4 # number of observed successes

# Analytical results:

a_post = a + x
b_post = b + n-x
exp_post = 1/(1+(b_post/a_post))
print(exp_post)

# Results by Likelihood-Weighting inference
inference_method = LW(betabinomial,\
                      {'successes':torch.tensor(x,dtype=torch.float)},\
                    f = {'mean':lambda x:x})
estimates,logW = inference_method.inferLW(L,a,b,n)
print(estimates)
```

B Framework for computational models of cognitive processes

The implementation of the class Variable.

```
### Class Variable
class Variable():

    distributions = {'Normal':pyro.distributions.Normal, \
                     'Binomial':pyro.distributions.Binomial,\
                     'Exponential':pyro.distributions.Exponential}

    def __init__(self,internal,name,dist,param):
        # binary variable indicating what type of variable it is
        self.internal = internal
        self.name = name # name of the random variable
        self.dist = dist # name of the distribution
```

```
        # ordered dictionary containing the parameters
        # to describe the distribution
        # parameters must be entered into the dictionary
        # in the same sequence they are entered into the
        # distributional object
        self.param = param
        self.sample()

    def return_dist(self):
        '''
        Return a distributional object that can be
        passed to the pyro.sample statement
        '''
        return distributions[self.dist](*self.param.values())

    def sample(self):
        '''
        Update the current value
        by sampling from the distribution
        If the distributional typ is 'fixed'
        then the variable is a constant
        '''
        if 'fixed' in self.dist:
            self.current_value = self.param['fixed']
        else:
            self.current_value = \
            distributions[self.dist](*self.param.values()).sample()

    def get_current_value(self):
        '''
        Returns the current value
        '''
        return self.current_value

    def __gt__(self,i):
        '''
        i (float): value the current value of variable
                   is compared to.

        Implements the magic method __gt__
        for automatic '>'-comparison
        '''
        c = self.get_current_value()
        if c>i:
            return 1
        return 0
```

```
    def __eq__(self,i):
        '''
        i (float): value the current value of variable
                   is compared to.

        Implements the magic method __eq__
        for automatic '=='-comparison
        '''
        c = self.get_current_value()
        if c==i:
            return 1
        return 0
```

The implementation of the intuitive theory.

```
def sigmoid(x):
    '''
    Implements the sigmoid function for a one-dimensional input
    '''
    return 1/(1+np.exp(-x))

def intuitive_theory(Skill,Effort,External,Luck):
    '''
    Skill,Effort,External,Luck: Instances of the Variable class

    return: dictionary of the sampled values
    '''
    # Sample skill
    skill = pyro.sample(Skill.name,Skill.return_dist())
    # Sample Effort
    effort = pyro.sample(Effort.name,Effort.return_dist())
    # Sample External
    external = pyro.sample(External.name,External.return_dist())
    # Sample luck
    luck = pyro.sample(Luck.name,Luck.return_dist())
    # Determine success probability
    success_prob = sigmoid(skill+effort+external+luck)
    # Sample success
    success = pyro.sample('success',\
            pyro.distributions.Bernoulli(success_prob))
    # Create a dictionary with a the sampled variables,
    # return this dictionary
    dic = {Skill.name:skill,Effort.name:effort,External.name:external\
           ,Luck.name:luck,'success':success}
    return dic
```

The implementation of the class Human.

```
### Class Human
class Human():
    '''
    Class to represent a participant of an experiment
    The class is equipped with a dictionary variables.
    This dictionary keeps track of all the variables
    that describe the person or the experiment
    The class additionally contains attributes that
    keep track of lists that specify the names of
    classes of variables. All variables that are relevant
    for the intuitive theory for example are saved in
    the attribute self.intutive_theory_params

    Which classes of variables matter depends
    on the intuitive theory
    Additionally the Human class has attributes
    specifying the intutive theory the participant
    has of the performance situation
    and specifying what parameters are needed to perform
    inference on the intutive theory.
    '''
    def __init__(self,self_concept,relevance,\
                intutive_theory_params,intuitive_theory\
                ,inference_params):
        # variable names defining the self concept
        self.concept = self_concept
        # variable names inlfuencing the relevance of the task
        self.relevance = relevance
        # register the intuitive theory with the participant
        self.intuitive_theory = intuitive_theory
        # variable names defining the intuitive theory
        self.intuitive_theory_params = intuitive_theory_params
        # variable names of inference parameters
        self.inference_params = inference_params

    def set_variables(self,variables,sample):
        '''
        Set current variables for a person
        Specify whether the variables should be sampled
        '''

        self.variables = variables
        if sample:
            for elem in self.variables:
                if isinstance(elem,Variable):
                    elem.sample()
```

```
def get_inference_params(self):
    '''
    Returns a list of variables
    selected from the variables dictionary
    that are specified as inference parameters
    '''

    return [self.variables[elem] for elem in self.inference_params]

def get_intuitive_theory_params(self):
    '''
    Returns a list of variables
    selected from the variables dictionary
    that are specified as intuitive-theory parameters
    '''
    return [self.variables[elem] for elem in self.intuitive_theory_params]

def get_self_worth(self,current_situation):
    '''
    Determines the level of self-worth of
    a person, before any observation has been made

    current_situation is a dictionary containing
    all variable names and values
    '''
    self_worth = 0
    for elem in self.concept:
        self_worth += current_situation[elem]['mean']
    return self_worth

def get_relevance(self):
    '''
    Determines the relevance of the experimental task
    to the participant
    '''

    relevance = 0
    for elem in self.relevance:
        relevance += self.variables[elem].get_current_value()
    return relevance

def decorate_self_worth(self):
    '''
```

```
        Returns an extended intuitive theory which
        also samples self-worth, such that prior self-worth
        can be conditioned on.
        '''

        def new_it(*args):
            # Run the initial intuitive theory
            sampled_var = self.intuitive_theory(*args)
            mean = 0
            # determine the sampled self-worth
            for elem in self.concept:
                mean += sampled_var[elem]
            self.worth = pyro.sample('self-worth',\
                                     pyro.distributions.Normal(mean,1))
            return self.worth
        # return the extended intutive theory
        return new_it

    def discrepancy(self,prior,post):
        '''
        Determine the difference between the variables
        that describe the self-concept, before and after
        inference is made.
        '''

        diff = 0
        for elem in self.concept:
            diff += post[elem]['mean']-prior[elem]['mean']
        return diff

    def do_attribution(self,prior,post,alpha,beta):
        '''
        Given prior and posterior means of the unobserved
        variables, determine the difference and weight it
        to get a measure of how much the participant attributed
        an event internally vs. externally.
        '''

        internal = sum([abs(post[elem.name]['mean']-prior[elem.name]['mean'])\
                        for elem in self.get_intuitive_theory_params()\
                        if elem.internal==1])
        external = sum([abs(post[elem.name]['mean']-prior[elem.name]['mean'])\
                        for elem in self.get_intuitive_theory_params()\
                        if elem.internal==0])
        attr = alpha*internal-beta*external
        return attr.item()
```

```
def do_attribution_internal(self,prior,post,alpha,beta):
    '''
    Does the same as do_attribution but ignores the external variables
    '''

    internal = sum([abs(post[elem.name]['mean']-prior[elem.name]['mean'])\
                    for elem in self.get_intuitive_theory_params()\
                    if elem.internal==1])
    attr = alpha*internal
    return attr.item()

def inference(self):
    '''
    Implements the process model of the SSB
    and returns a measure of the level of
    internal attribution by the participant
    '''

    # Perfrom inference only letting the internal
    # variables vary.
    prior = {elem.name: elem.param for elem \
             in self.get_intuitive_theory_params()}

    # Add the outcome to the observations
    obs = {'success':self.variables['success']}

    # Add the external variables to the
    # observations
    for elem in self.get_intuitive_theory_params():
        if elem.internal==0:
            obs[elem.name]=prior[elem.name]['mean']

    # Create the LW-inference class
    inference_class = LW(self.intuitive_theory,self.variables['f'])
    estimates, logW_sum = inference_class.inferLW\
    (*self.get_inference_params(),obs,*self.get_intuitive_theory_params())

    # Extract results
    post = {elem.name: estimates[elem.name] \
            for elem in self.get_intuitive_theory_params()}

    # Check whether self-awareness is high
    self.variables['SA'].sample()
    if not (self.variables['SA']>0):
        # if self-awareness is low use the automatic
```

```
        # inference at the beginning to do causal
        # attributions
        print('no-self-awareness')
        return self.do_attribution_internal(prior,post,1,1)
    else:
        print('self-awareness')
        # if self-awareness is low
        # The idea is that we calculate
        # how relevant is the task to the person
        relevance = self.get_relevance()
        # Is there any discrepancy between
        # the self-concept and what is implied by
        # the inference
        diff = self.discrepancy(prior,post)
        # Discrepancy is worse/better for relevant tasks
        diff = relevance*diff
        # Is the inference better than our self-concept or worse
        if diff>=0:
            print('positive diff')
            # Positive inference
            # Internal attribution is stronger
            return self.do_attribution_internal(prior,post,1+diff,1)
        else:
            print('negative diff')
            # Negative Difference
            # is there a probability of improvement
            self.variables['PI'].sample()
            if self.variables['PI']>0:
                print('probability of improvement exists')
                # Accurate inference
                return self.do_attribution_internal(prior,post,1,1)
            else:
                print('no probability of improve')
                # Determine self-worth before
                # the event happended
                self_worth = self.get_self_worth(prior)
                # Extend the intutive theory to contain self-worth
                self_worth_model = self.decorate_self_worth()
                # Condition the model on self_worth and do inference
                obs = {'self-worth':self_worth,\
                       'success':self.variables['success']}

                inference_class = LW(self_worth_model,self.variables['f'])

                estimates, logW_sum = inference_class.inferLW\
                (*self.get_inference_params(),obs,\
```

```
                    *self.get_intuitive_theory_params())

                # Get values after conditioning
                post = {elem.name: estimates[elem.name] \
                        for elem in self.get_intuitive_theory_params()}
                # Do attribution
                return self.do_attribution(prior,post,1,1+abs(diff))
```

The implementation of class Experiment

```
# Class Experiment

class Experiment():
    '''
    To run a simulation for an experiment is simplified
    by this class

    A list of experimental conditions is kept.
    The experimental conditions are described
    by a setting of the variables in the variables
    dictionary, a number of participants in each
    experimental group and a name for the condition

    The conditions can be run and results can be plotted

    '''
    def __init__(self,name,human,variables):
        self.name = name
        self.conditions = []
        self.n_conditions = 0
        self.human = human
        self.variables = variables

    def register_condition(self,condition):
        '''
        condition: a dictionary that contains
        a setting of the variables, number of participants
        and a name of the condition
        '''

        self.n_conditions += 1
        self.conditions.append(condition)

    def run(self):
        '''
        Run each condition and save the results
        in a dictioanry with condition names as keys
```

```
        and a list of attribution values as values
        '''
        # Dictionary that stores results
        self.results = {}
        # Repeat for each condition
        for elem in self.conditions:
            attributions = []
            # set up a dictionary that contains the values
            # of the variables in that condition
            vs = self.variables
            for var_name,var_value in elem.items():
                if var_name!='N' and var_name!='name':
                    vs[var_name]=var_value

            # Repeat for the number of participants in a condition
            for _ in range(elem['N']):
                # Create a particpant
                self.human.set_variables(vs,True)
                # Do inference and save it
                attributions.append(self.human.inference())
            # save results for that condition
            self.results[elem['name']] = attributions

    def z_transform(self,attr,mean,std):
        '''
        Given an array of values and a mean and std
        Apply the z-score transformation
        '''

        return (np.array(attr)-mean)/std


    def plot_result(self):
        '''
        Plot the mean z-score for each condition
        '''
        # Change all the attribution values into z-values
        # First create an array that contains all
        # attribution values of all conditions
        total = []
        for l in self.results.values():
            total +=l
        # Calculate the mean and std
        # of all values for z-transformation
        mean = np.mean(total)
        std = np.std(total)
```

```
        z_values = {name: np.mean(self.z_transform(value,mean,std))\
                    for name,value in self.results.items()}

        # Plot the different values for each experimental group
        names = z_values.keys()
        means = z_values.values()
        x = np.arange(self.n_conditions)
        plt.bar(x,means)
        plt.xticks(x,names)
        plt.title(self.name)
        plt.ylabel('Level of internal attribution')
        plt.savefig('Experiment_results.png')
        plt.plot()
        return z_values
```

C Experiments

Experimental setup to test whether the model can replicate the SSB

```
# Testing whether the SSB bias can be replicated
from collections import OrderedDict

variables = {}

default = OrderedDict()
default['mean'] = 0
default['std'] = 1
positive_bias = OrderedDict()
positive_bias['mean'] = 0.5
positive_bias['std'] = 1
negative_bias = OrderedDict()
negative_bias['mean'] = -0.5
negative_bias['std'] = 1

# Experimental parameters

# Task importance
variables['TI'] = Variable(1,'TI','fixed',{'fixed':1})

# Controlled parameters
variables['SA'] = Variable(0,'SA','Normal',positive_bias)
variables['PI'] = Variable(0,'PI','Normal',negative_bias)
variables['success'] = 0

# IT Process parameters
variables['skill'] = Variable(1,'skill','Normal',default)
```

```
variables['effort'] = Variable(1,'effort','Normal',default)
variables['external'] = Variable(0,'external','Normal',default)
variables['luck'] = Variable(0,'luck','Normal',default)

# Inferencee parameters
variables['L']=100
variables['f']={'mean':lambda x:x,'2ndMoment':lambda x:x**2}


intuitive_theory_params = ['skill','effort','external','luck']
self_concept = ['skill','effort']
relevance = ['TI']
inference_params = ['L']

human = Human(self_concept,relevance,intuitive_theory_params\
              ,intuitive_theory,inference_params)
ExperimentTI = Experiment('SSB',human,variables)

condition1 = {'N':50,'name':'failure','success':0}
condition2 = {'N':50,'name':'success','success':1}


ExperimentTI.register_condition(condition1)
ExperimentTI.register_condition(condition2)

ExperimentTI.run()
ExperimentTI.plot_result()
```

Experimental Setup to test the main effect of Task Importance.

```
# Testing the moderator effect of Task importance!

from collections import OrderedDict

variables = {}

default = OrderedDict()
default['mean'] = 0
default['std'] = 1
positive_bias = OrderedDict()
positive_bias['mean'] = 0.5
positive_bias['std'] = 1
negative_bias = OrderedDict()
negative_bias['mean'] = -0.5
negative_bias['std'] = 1

# Experimental parameters
```

```
# Task importance
variables['TI'] = Variable(1,'TI','fixed',{'fixed':0})

# Controlled parameters
variables['SA'] = Variable(0,'SA','Normal',positive_bias)
variables['PI'] = Variable(0,'PI','Normal',negative_bias)
variables['success'] = 0

# IT Process parameters
variables['skill'] = Variable(1,'skill','Normal',default)
variables['effort'] = Variable(1,'effort','Normal',default)
variables['external'] = Variable(0,'external','Normal',default)
variables['luck'] = Variable(0,'luck','Normal',default)

# Inferencee parameters
variables['L']=100
variables['f']={'mean':lambda x:x,'2ndMoment':lambda x:x**2}


intuitive_theory_params = ['skill','effort','external','luck']
self_concept = ['skill','effort']
relevance = ['TI']
inference_params = ['L']

human = Human(self_concept,relevance,intuitive_theory_params\
             ,intuitive_theory,inference_params)
ExperimentTI = Experiment('Effect of Task Importance',human,variables)

condition1 = {'N':50,'name':'Low-TI-failure',\
              'TI':Variable(1,'TI','fixed',{'fixed':0.5}),'success':0}
condition2 = {'N':50,'name':'High-TI-failure',\
              'TI':Variable(1,'TI','fixed',{'fixed':2}),'success':0}
condition3 = {'N':50,'name':'Low-TI-success',\
              'TI':Variable(1,'TI','fixed',{'fixed':0.5}),'success':1}
condition4 = {'N':50,'name':'High-TI-success',\
              'TI':Variable(1,'TI','fixed',{'fixed':2}),'success':1}
ExperimentTI.register_condition(condition1)
ExperimentTI.register_condition(condition2)
ExperimentTI.register_condition(condition3)
ExperimentTI.register_condition(condition4)

ExperimentTI.run()
ExperimentTI.plot_result()
```

References

Alicke, M. D. (1985). Global self-evaluation as determined by the desirability and controllability of trait adjectives. *Journal of personality and social psychology*, *49*(6), 1621.

Al-Zahrani, S. S. A., & Kaplowitz, S. A. (1993). Attributional biases in individualistic and collectivistic cultures: A comparison of americans with saudis. *Social Psychology Quarterly*, 223–233.

Arkin, R. M., Appelman, A. J., & Burger, J. M. (1980). Social anxiety, self-presentation, and the self-serving bias in causal attribution. *Journal of personality and social psychology*, *38*(1), 23.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., ... Goodman, N. D. (2019). Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, *20*(1), 973–978.

Campbell, W. K., & Sedikides, C. (1999). Self-threat magnifies the self-serving bias: A meta-analytic integration. *Review of general Psychology*, *3*(1), 23.

Carey, S., & Bartlett, E. (1978). Acquiring a single new word.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., ... Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of statistical software*, *76*(1).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Duval, T. S., & Silvia, P. J. (2002). Self-awareness, probability of improvement, and the self-serving bias. *Journal of personality and social psychology*, *82*(1), 49.

Forgas, J. P. (1998). On being happy and mistaken: mood effects on the fundamental attribution error. *Journal of personality and social psychology*, *75*(2), 318.

Gelman, S. A. (1995). The role of covariation versus mechanism information in causal attribution. *Cognition*, *54*, 299–352.

Gerstenberg, T., & Tenenbaum, J. B. (2017). Intuitive theories. *Oxford handbook of causal reasoning*, 515–548.

Greenberg, J., Pyszczynski, T., Burling, J., & Tibbs, K. (1992). Depression, self-focused attention, and the self-serving attributional bias. *Personality and Individual Differences*, *13*(9), 959–965.

Hayes, S. D., Crocker, P. R., & Kowalski, K. C. (1999). Gender differences in physical self-perceptions, global self-esteem and physical activity: Evaluation of the physical self-perception profile model. *Journal of Sport Behavior*, *22*(1), 1.

Heider, F. (1958). *The psychology of interpersonal relations*. New York: Wiley.

Heider, F., & Simmel, M. (1944). An experimental study of apparent behavior. *The American journal of psychology*, *57*(2), 243–259.

Hoffmann, J., & Engelkamp, J. (2016). *Lern-und gedächtnispsychologie*. Springer-Verlag.

Hudson, D., & Manning, C. D. (2019). Learning by abstraction: The neural state machine. In *Advances in neural information processing systems* (pp. 5901–5914).

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kelley, H. H. (1973). The processes of causal attribution. *American psychologist*, *28*(2), 107.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and brain sciences*, *40*.

Leary, M. R., & Kowalski, R. M. (1990). Impression management: A literature review and two-component model. *Psychological bulletin*, *107*(1), 34.

Lee, L., Frederick, S., & Ariely, D. (2006). Try it, you'll like it: The influence of expectation, consumption, and revelation on preferences for beer. *Psychological science*, *17*(12), 1054–1058.

Lord, C. G., Ross, L., & Lepper, M. R. (1979). Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of personality and social psychology*, *37*(11), 2098.

Malle, B. F. (1999). How people explain behavior: A new theoretical framework. *Personality and social psychology review*, *3*(1), 23–48.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., & Wu, J. (2019). The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Marsh, H. W., & Yeung, A. S. (1997). Causal effects of academic self-concept on academic achievement: Structural equation models of longitudinal data. *Journal of educational psychology*, *89*(1), 41.

Marsh, L., Edginton, T., Conway, M., & Loveday, C. (2019). Positivity bias in past and future episodic thinking: Relationship with anxiety, depression, and retrieval-induced forgetting. *Quarterly Journal of Experimental Psychology*, *72*(3), 508–522.

Martin, W. A., & Fateman, R. J. (1971). The macsyma system. In *Proceedings of the second acm symposium on symbolic and algebraic manipulation* (pp. 59–75).

McAllister, H. A. (1996). Self-serving bias in the classroom: Who shows it? who knows it? *Journal of Educational Psychology*, *88*(1), 123.

Miller, D. T. (1976). Ego involvement and attributions for success and failure. *Journal of personality and social psychology*, *34*(5), 901.

Nickerson, R. S. (1998). Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology*, *2*(2), 175–220.

Nilsson, N. J. (2009). *The quest for artificial intelligence*. Cambridge University Press.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (pp. 8024–8035).

Pyszczynski, T., & Greenberg, J. (1987). Toward an integration of cognitive and motivational perspectives on social inference: A biased hypothesis-testing model. In *Advances in experimental social psychology* (Vol. 20, pp. 297–340). Elsevier.

Sejnowski, T. J. (2018). *The deep learning revolution*. MIT Press.

Shavelson, R. J., Hubner, J. J., & Stanton, G. C. (1976). Self-concept: Validation of construct interpretations. *Review of educational research*, *46*(3), 407–441.

Shepperd, J., Malone, W., & Sweeny, K. (2008). Exploring causes of the self-serving bias. *Social and Personality Psychology Compass*, *2*(2), 895–908.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, *38*(3), 58–68.

Turing, A. M. (1950). Computing machinery and intelligence. *MIND: A Quarterly Review of Psychology and Philosophy*, *59*(236), 433–460.

van de Meent, J.-W., Paige, B., Yang, H., & Wood, F. (2018). An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*.

Weizenbaum, J., et al. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Wright, J. C., & Mischel, W. (1988). Conditional hedges and the intuitive psychology of traits. *Journal of personality and social psychology*, *55*(3), 454.