24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Learning Normative Behaviors through Abstraction

Stevan Tomic and **Federico Pecora** and **Alessandro Saffiotti**[1]

Abstract. Future robots should follow human social norms to be useful and accepted in human society. In this paper, we show how prior knowledge about social norms, represented using an existing normative framework, can be used to (1) guide reinforcement learning agents towards normative policies, and (2) re-use (transfer) learned policies in novel domains. The proposed method is not dependent on a particular reinforcement learning algorithm and can be seen as a means to learn abstract procedural knowledge based on declarative domain-independent semantic specifications.

1 Introduction

In order to be accepted in human society, robots need to comply with human social rules. The main goal of our research is to make robots capable of behaving in a socially-acceptable way, i.e., to adhere to social norms. One possible approach to do this would be to create models from first principles which can be used for synthesizing socially-acceptable behavior, e.g., appropriate planning domains. However, describing such models requires a significant amount of effort: the difference between normative and norm-breaking behavior may be subtle, and its specification may require a high level of detail. This also limits the applicability of such models across domains. Another possibility is to endow agents with data-driven approaches and the ability to learn to act in model-free environments, generalizing over similar states. This could be achieved, e.g., using Reinforcement Learning (RL) methods. In recent years, RL has achieved notable successes in simulated environments [21] and with physical robot control [2]. Still, despite impressive results, RL is relegated to relatively tightly controlled settings. Using RL in real-world settings poses significant challenges, most notably the classical problems of *credit assignment* [12], which concerns how an action that was taken earlier influences the final reward, and *the curse of dimensionality* [4], where the states to be explored increase exponentially when new factors are considered. These challenges are further complicated if robots have to act in human social spaces. Policies obtained via RL lead an agent to act in a way that maximizes the reward function, meaning that an agent may learn to achieve its goal in a way which is not socially acceptable. This somewhat Machiavellian characteristic of RL agents indicates that policies that are to be used in social environments should be subject to, or at least biased by, social norms. Also open is the issue of applying learned policies to novel domains. This is related to the transfer of learning phenomenon studied in psychology [25], and to the well known open challenge of *transfer learning* in Machine Learning [17].

Several approaches for overcoming the difficulties of RL are reported in the literature. 'Reward shaping' [13] is a technique where agents receive intermediate rewards to direct learning toward achieving the final goal; usually, this requires significant engineering effort. A way to address the 'curse of dimensionality' problem is to reduce the size of the state space [11]. Intermediate rewards and state-space/action reduction are both crucial ideas that are utilized in our work. Social behavior in the context of RL is usually achieved through inverse reinforcement learning (IRL) algorithms [14], which focus on learning a reward distribution from exemplary behaviors and then use the obtained rewards in ordinary RL settings. RL and normative monitoring are combined when learning policies involves balancing between individual goal rewards and penalties for non-adherent behavior [10]. Another interesting approach in the context of RL safety uses the concept of 'restricting bolts', where learning can be performed to conform as much as possible to specifications in extended temporal logic [7].

This paper proposes an approach that combines modeling of behavioral rules from first principles (atomic statements) and RL. Humans learn social behavior guided by prior knowledge from parents and society, and robots should do the same. Specifically, our aim is to capture prior social knowledge in norms and to learn norm-adhering policies in an RL setting. We also aim to re-use these policies in novel domains. We rely on the concept of *institutions* [20], often used in normative multi-agent systems (MAS), to model social structures, and to enable the transfer of learned knowledge across domains. Institutions encapsulate *"the rules of the game in a society"* [15], providing the mechanisms that regulate interactions, and the 'count-as' principle [20] as a way to abstract, interpret and assign a social meaning to physical execution. The concept of (electronic) institutions is well-known in MAS [8]. These usually include norms defined with deontic operators associated with different types of logic to provide operational semantics for monitoring or execution [1]. The definition of an institution, the level of norms abstraction, expressiveness and monitoring capabilities vary [16].

The main contribution of this paper is the use of norms, represented in an institution framework [23], to shape the reward function of learning agents and to provide a formal way to abstract learning of norm-compliant policies. The proposed method is contrasted to ad-hoc policies that would be learned for a particular domain setting. This is an important characteristic of our approach that we examine in our evaluation. The institution framework that we use provides: (A) a means to represent social knowledge via abstracted, thus *domain-independent* norms; and (B) a mechanism based on declarative *norm semantics* to verify whether a social interpretation of the execution is adherent to the given norms. As we will see, feature (A) provides a state-space/feature selection mechanism which reduces the dimensionality of the problem; it also leads to abstracted normative policies, thereby addressing the transfer learning problem, so that we can apply the same norms in novel domains without re-learning. Feature (B) is used to automatically create a reward system to guide

[1] All authors are affiliated with Örebro University, Sweden; Contact email: stevan.tomic@aass.oru.se

learning, addressing the credit assignment problem.

2 The Formal Model of Institution

The institutional framework used in this paper is based on our previous work [23]. We briefly recall the parts of the framework which are relevant to this paper.

2.1 Institutions, Domains and Grounding

2.1.1 Institutions

Institutions capture the structure of a social situation by defining sets (categories) of roles ($Roles$), institutional actions ($Acts$), and artifacts ($Arts$) and a collection of *norms* that relates members (elements) of these sets. Examples of $Roles$ are a 'customer' in a store or a 'goal-keeper' in a football game. Similarly, artifacts ($Arts$) represent objects with certain institutional meaning, as 'goods' or a 'cash register', while *Acts* are actions meaningful in institutional (social) context, e.g., 'pay' for a carton of milk in a store. Norms are defined as predications over statements:

Definition 1. A *norm* is a statement of the form $q(trp^*)$, where q is a qualifier and trp is a triple of the form:

$$trp \in \mathit{Roles} \times \mathit{Acts} \times (\mathit{Arts} \cup \mathit{Roles})$$

Qualifiers can be unary relations like must or must-not, or n-ary ones like inside or before. For example, must can be used to represent the obligation 'A buyer *must* pay with cash':

$$\mathsf{must}\,((\mathsf{Buyer}, \mathsf{Pays}, \mathsf{Cash}))\,.$$

A qualifier at can indicate a spatial relation on the location of an action, as in 'Paying should be performed *at* a cash register': $\mathsf{at}\,((\mathsf{Buyer}, \mathsf{Pays}, \mathsf{CashRegister}))$. Binary qualifiers can express relations between statements, e.g., temporal relations such as before or during, for instance $\mathsf{before}\,((\mathsf{Buyer}, \mathsf{GetGoods}, \mathsf{Goods})\,, (\mathsf{Buyer}, \mathsf{Pays}, \mathsf{Cash}))$ indicating that 'A buyer should get the goods *before* paying'. Institutions put all the above elements together.

Definition 2. An *institution* is a tuple

$$\mathcal{I} = \langle \mathit{Arts}, \mathit{Roles}, \mathit{Acts}, \mathit{Norms} \rangle.$$

An institution is a (social) abstraction, which can be used to interpret and regulate execution in concrete physical systems that may be different. For instance, the same 'store' institution can be used to interpret and regulate behaviors of agents in different markets, irrespective of these agents being humans, robots, or a combination of both. Such a concrete system is called a *domain*.

2.1.2 Domain

A domain is a tuple $\mathcal{D} = \langle A, B, O, R \rangle$, where: A is a set of *agents*; this may include humans (e.g., john), robots (e.g., robby), or both; B is the set of all *behaviors* that agents can perform, like pick or speak; O is the set of *objects* in the domain, like door, cash; and R is a set of *state variables*, each defining an attribute (or property) pertaining to each entity (agent, behavior, or object) in the domain. For instance, pos(robby) and pos(forky) are state variables that indicate positions of the agent robby and the agent forky, while active(pick, robby) indicates the activation of robby's behavior pick. The set of possible values of a state variable $\rho \in R$ is denoted $\mathrm{vals}(\rho)$, e.g., $\mathrm{vals}(\mathsf{active}(\mathsf{pick}, \mathsf{robby})) = \{\top, \bot\}$.

Definition 3. The *state space* of $\mathcal{D}$ is $\mathcal{S} = \prod_{\rho \in R} \mathrm{vals}(\rho)$. We call any element $s \in \mathcal{S}$ a *state*. The value of ρ in state s is denoted $\rho(s)$.

In a dynamic environment, the values of most properties change over time. Time points are represented by natural numbers in $\mathbb{N}$, and time intervals by pairs $I = [t_1, t_2]$ such that $t_1, t_2 \in \mathbb{N}$ and $t_1 \leq t_2$. The set of all such time intervals are denoted by $\mathbb{I}$.

Definition 4. A *trajectory* is a pair (I, τ), where $I \in \mathbb{I}$ is a time interval and $\tau : I \to \mathcal{S}$ maps time to states.

In simple words, a trajectory represents how the domain evolves over a given time interval.

2.1.3 Grounding

Grounding establishes the relation between an abstract institution and a specific domain:

Definition 5. Given an institution $\mathcal{I}$ and a domain $\mathcal{D}$, a *grounding* of $\mathcal{I}$ into $\mathcal{D}$ is a tuple $\mathcal{G} = \langle \mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_O \rangle$, where:

- $\mathcal{G}_A \subseteq \mathit{Roles} \times A$ is a *role grounding*,
- $\mathcal{G}_B \subseteq \mathit{Acts} \times B$ is an *action grounding*,
- $\mathcal{G}_O \subseteq \mathit{Arts} \times O$ is an *artifact grounding*.

Grounding establishes the relation between abstract and concrete and enables the interpretation of an underlying domain in terms of institutional entities. As such, it provides the means to reuse the same abstract institution to regulate different physical systems that can be interpreted in the same way. For example, using different $\mathcal{G}$'s, the 'store' institution can be grounded in different physical or virtual marketplaces. Another institution, encapsulating the rules of a football game, may be grounded in a meadow serving as a playing field, with stones as goalposts, and children being grounded to roles like goalkeeper, defender, etc. In general, one of the main properties of grounding is *admissibility*. It defines what is regarded as proper grounding. Intuitively, if an institution includes an obligation norm for some role, then the agents to which that role is grounded should be capable of executing the actions required by the norm. For instance, the agent grounding the goalkeeper role would have to be capable of playing in that role for the grounding to be admissible. For formal definition of admissibility, please refer to [23]. In this paper we assume that all groundings are admissible and that $\mathcal{G}_a$, $\mathcal{G}_b$ and $\mathcal{G}_o$ each map institution elements to exactly one domain element, that is, groundings are functions.

2.2 Norms Semantics and Norms States

By grounding an institution $\mathcal{I}$ into a domain $\mathcal{D}$, we impose that the norms stated for abstract entities in $\mathcal{I}$ must hold for the corresponding concrete entities in $\mathcal{D}$. But what does "hold" mean here? In the institutional framework, norms are given semantics in terms of execution in a physical domain. The intuitive semantic of $\mathsf{must}\,((\mathsf{ro}, \mathsf{act}, \mathsf{art}))$ is that the agent taking role ro must perform at least once the behavior implementing action act using the object bound to art. Formally, this semantics is the set of all temporal trajectories in state space where the above condition is fulfilled.

Let T be the set of all possible trajectories (I, τ) over the state variables of domain $\mathcal{D}$. Given a norm $q(trp^*)$ and $\mathcal{G}$ we denote semantics by:

$$[\![q(trp^*)]\!] \subseteq T$$

We distinguish between the following types of norm semantics: *fulfillment* semantics, $[\![q(trp^*)]\!]_F$, defining the set of trajectories fulfilling the norm; and *violation* semantics, $[\![q(trp^*)]\!]_V$, defining the set of trajectories violating the norm. Naturally, a trajectory cannot be an element of both fulfillment and violation semantics for a given norm, that is, $[\![q(trp^*)]\!]_F \cap [\![q(trp^*)]\!]_V = \emptyset$. Accordingly, we define the function ns indicating the *state of a norm* given a grounding and a trajectory:

$$\mathrm{ns}(q(trp^*), (I,\tau)) = \begin{cases} f, \text{iff } (I,\tau) \in [\![q(trp^*)]\!]_F \\ v, \text{iff } (I,\tau) \in [\![q(trp^*)]\!]_V \\ n, \text{otherwise} \end{cases}$$

where f stands for *fulfilled*, v for *violated*, and n for *neither* fulfilled nor violated.

Example of Semantics. A variety of norm semantics can be expressed as constraints on trajectories, i.e., as constraints on the possible values of state variables over time. For example, the fulfillment semantics of the must norm can be expressed by requiring that the state variable indicating activation of the relevant behavior is true at least once. Formally:

$$[\![\,\mathsf{must}\,((role, act, art))]\!]_F \equiv$$
$$\{(I,\tau) \mid \forall a \in A_{\mathrm{role}}.\exists (b,t) \in B_{\mathrm{act}} \times I : \mathsf{active}(b,a)(\tau(t)) = \top\},$$

where A_{role} is the set of all agents role is grounded to, and B_{act} is the set of all behaviors act is grounded to. Intuitively, these semantics select all trajectories where any agent taking role activates at least once a behavior that performs act (in this example, we do not require that the artifact art is used). Variants of this semantics are possible, for example, stating that the behavior should be enacted at all times (not just once in the trajectory). The semantics specifying spatial 'at' relation can be defined as follows:

$$[\![\,\mathsf{at}\,((role, act, art))]\!]_F \equiv$$
$$\{(I,\tau) \mid \forall (b,a,t) \in B_{act} \times A_{role} \times I.\exists o \in O_{art} :$$
$$\mathsf{active}(b,a)(\tau(t)) = \top \implies \mathsf{pos}(b,a)(\tau(t)) = \mathsf{pos}(o)(\tau(t))\}.$$

Examples of temporal semantics are available in [23], while an example of violation semantics is available in [24]. In general, the expressiveness of norms depends on two factors, namely, on the complexity of relations in norm semantics, and on the availability of appropriate state variables R.

Semantics are defined over the states of the same types (attributes), and as such, they are applicable to institutional elements. For instance concrete values of $\mathsf{active}(\mathsf{role}_1, \mathsf{act}_1)$ can be obtained given the grounding $\mathcal{G}_a(\mathsf{role}_1, \mathsf{robby})$ and $\mathcal{G}_b(\mathsf{act}_1, \mathsf{pick})$ which will result in the concrete state-variable $\mathsf{active}(\mathsf{pick}, \mathsf{robby})$, while $\mathcal{G}_a(\mathsf{role}_1, \mathsf{forky})$ and $\mathcal{G}_b(\mathsf{act}_1, \mathsf{pick})$ will result in $\mathsf{active}(\mathsf{pick}, \mathsf{forky})$. Hence, norm semantics retain a degree of abstraction, which is fully grounded upon selection of a particular grounding. As we show in Section 4, this feature is key to enabling transfer learning.

2.3 Adherence

We are now in a position to define what it means for a given physical system to behave in compliance with an institution. Consider an institution $\mathcal{I}$, and suppose that $\mathcal{I}$ has been grounded in a given domain $\mathcal{D}$ through some grounding $\mathcal{G}$. Further, suppose that all the norms in $\mathcal{I}$ are given fulfillment semantics in $\mathcal{D}$ through the function $[\![\cdot]\!]_F$.

The following definition tells us whether or not a specific, concrete execution in $\mathcal{D}$ *adheres to* the abstract institution $\mathcal{I}$ given the above grounding and semantics.

Definition 6. A trajectory (I,τ) *adheres* to an institution $\mathcal{I}$, with admissible grounding $\mathcal{G}$ and semantics function $[\![\cdot]\!]_F$, if

$$(I,\tau) \in [\![\mathsf{norm}]\!]_F, \forall\, \mathsf{norm} \in \mathit{Norms}.$$

Adherence allows us to distinguish between trajectories (executions) which are norm-adherent and others which are not. Computationally, we rely on the adherence verification mechanism described in [23], which is used in this paper to evaluate the state of each norm during agent execution, providing us with an implementation of the ns function.

3 Applying Institution Models to RL agents

Markov Decision Processes (MDPs) [18] are an example of how reinforcement learning can be used to direct the actions of agents. A MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is a model, given as transitions probability between states depending on actions, and $\mathcal{R}$ is a set of rewards – a scalar feedback signal which agents get as they change their states. $\gamma \in [0,1]$ is a discount factor indicating the importance of future rewards. The goal of a RL agent (the RL problem) is to find a mapping between states and actions which maximizes the amount of reward the agent receives, known as cumulative reward or return. Such a mapping is represented by a policy $\pi(a_t \mid s_t)$, which gives the probability of taking action $a_t \in \mathcal{A}$ given the state $s_t \in \mathcal{S}$. RL algorithms use the notion of value functions: a value function $V^\pi(s)$ provides the expected return of rewards starting from state s acting according to policy π; and/or a action-value function (Q-function), $Q^\pi(s,a)$, which estimates the value of a state given the action. The learning process consists of value updates which propagate the rewards until they converge to (near) optimal values. For example, a standard idea in RL to learn Q-values is known as temporal difference learning:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)),$$

where α is a learning rate, and s' and a' are the state and action subsequent to s and a. For a variety of real-world problems the model $\mathcal{P}$ is unknown and the state space may be too large to store. Value function approximation methods address these problems. In such approaches state-space variables are represented as a *feature vector*, where each feature is a number, describing some property in the state-space. In this paper we are interested in learning social behavior in a model-free environment ($\mathcal{P}$ is unknown), driven by *prior knowledge* of social norms, and in applying the learned policies in novel domains.

3.1 From Norms to Rewards

The ns function (see Section 2.2) evaluates whether norms are fulfilled, neutral, or violated. This can be used to feed back signals regarding the agents' adherence to norms. The schematic view of this approach is shown in Figure 1. The 'institutional model' takes only those states which are used in the norm semantics and evaluates the norms. Naturally, to evaluate norms which have a temporal dimension, a history of states has to be stored. Note, however, that the semantics of temporal relations are expressed using state variables that describe state activation; similarly, the semantics of spatial relations are expressed via state variables that describe the position. Hence, an increase in the number of norms usually does not entail a proportional increase in the number of stored states. Note that the 'institutional model' is separated from the concrete RL algorithm and is used only to provide feedback signal to RL agents.

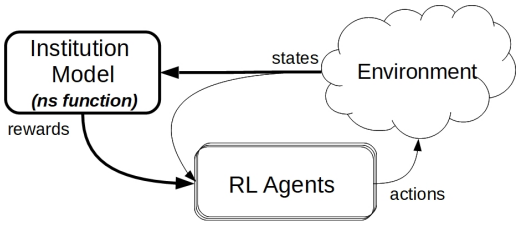


Figure 1: RL agents receive feedback signals from the institution.

Given a trajectory and a norm, the ns function evaluates the state of the norm as fulfilled, violated or neither. The norm reward function maps two successive evaluations to a feedback signal:

$$\mathcal{F}_{\text{norm}} : \{f, v, n\} \times \{f, v, n\} \mapsto \mathbb{R}$$

There are 9 possible transitions of norm evaluations which can be used to signal positive or negative feedback, namely: (f, f), (f, n), (f, v), (n, f), (n, n), (n, v), (v, f), (v, n), (v, v). The $\mathcal{F}_{\text{norm}}$ function can be realized in different ways; while its most effective form may depend on the underlying RL algorithm, we explore the following general ideas.

Feedback from Full Adherence. The adherent reward is the main reward or the *final goal* for learning normative behavior. Agents receive this reward only when their trajectory is adherent to the institution, i.e., to all institutional norms (see Definition 6). This can be realized with a procedure $\mathcal{F}_{\text{norm}}$ which is called at each training step or predefined intervals. For each norm, ns checks its fulfillment. If all norms are fulfilled, a 'full adherence' reward with a value 1 is returned. However, this means that an agent has to fulfill all the norms before it receives 'full adherence' reward and does not provide any feedback to norms states separately. Thus this reward is sparse and does not provide helpful intermediate feedback for the credit assignment problem.

Dynamic Feedback from Norms Evaluations. Our verification mechanism can evaluate each norm separately. This approach provides feedback to the agents as norms are evaluated. This is used for reward shaping that is based on the *partial achievement* of the final goal. The norm reward function assigns numerical values to norm evaluation transitions. For example, transitions to fulfilled state, e.g., (n, f), may be rewarded while transitions to violated state, e.g., (n, v), may be penalized. In such a case we assign the value $1.0/$*(number of norms)* for transitions to fulfillment, while in the penalizing case, we assign the same negative value. If the order of execution, as specified by temporal norms, is violated, we stop all future rewards until the end of the training episode, since this enables agents to learn only from temporally consistent (ordered) normative behaviors. The feedback from norms can be used to either learn each norm in a separate policy and then combine them to achieve full adherence required by the institution, or to learn all norms in one policy.

3.2 Learning Through Abstraction

An interesting distinction that is often made in multi-agent systems is whether or not agents know about the organization in which they take part [5]. When learning, agents create ad-hoc policies that depend on a particular domain while not being 'aware' of the underlying institutional structure. In such a case, changing the grounding for the same norms (same pattern of behaviors) would require re-learning. The question is then how RL can be based on abstract re-usable institutional structures. An abstraction of domain elements implies that their meaning is defined through abstract categories such as roles, artifacts, and institutional actions. For example, a carton of milk and a bag of potatoes mean the same (belongs to the same category) as 'goods' in a Store institution. Our institutional model provides a method where an agent's observations and action space can be interpreted as such reusable structures. This is achieved by representing relevant parts of our framework in a matrix form as follows.

Concrete and Abstract State-Space Representation. State variables in R can be classified into types. For instance, all state variables describing 'position' of agents (or objects) belong to the same class. Each class represents a particular attribute of agents, objects, behaviors or their relation (e.g., active(ag,,b)). If the total number of all attributes is n, then each domain element and their 2-combinations are represented as a $1 \times n$ dimensional (row) vector. Vectors of domain elements are arranged in a matrix $\mathbf{D}^{m \times n}$, where m is the sum of the number of domain elements and their 2-combinations (e.g, $agent_1behavor_1, agent_1behavor_2$, etc.). Each entry in $\mathbf{D}$ denoted $d_{i,j}$ assigns the state value of the domain element (or 2-combination) in row i described by attribute in column j at time t. Hence $\mathbf{D}$ is a function of time, i.e., $\mathbf{D}(t)$. The trajectory (I, τ) maps time to state, thus it can be also represented in matrix form $\mathbf{D_I}^{m \times n \times |I|}$, where the third dimension represents $\mathbf{D}(t)$ at each $t \in I$.

An institution also has its own structure, which captures the states of institutional elements in roles, actions, and artifacts. Such a structure is defined by *transfer* matrix $\mathbf{T}^{k \times n}$, where dimension k is the sum of the number of institutional elements and their 2-combinations. Each entry in $\mathbf{T}$, denoted $st_{i,j}$, represent the value of the attribute in column j describing the institutional element (or their combination) in row i at the time t. Learning normative behavior from the states of institutional elements would produce (abstract) procedural knowledge based on declarative domain-independent semantic specifications, applicable to abstract categories (e.g., roles), and would not depend on a particular domain. Such a policy maps the values of domain-independent attributes (state-space) and the actions in Act. We call such a policy an *abstract* (or transfer) policy, and we denote it with π_λ.

At this stage, we have defined the structure of the state spaces of a domain as $\mathbf{D}$ and an institution as $\mathbf{T}$. Still, it is not clear how the values in $\mathbf{T}$ are obtained. Here the following question arises: How to map elements in $\mathbf{D}$ to a particular place in the reusable structure $\mathbf{T}$?

Structural Mapping. An institution is established by $\langle \mathbf{T}, [\![\cdot]\!] \rangle$, that is, its *structure* and its *function*. Its function (norm semantics) is already given in abstract terms, through relations of attribute values describing institutional elements. To interpret (map) domain states as institutional states, and to isolate attributes used only by semantics, we introduce two additional matrices, related to grounding and semantics. Let $\mathbf{G}^{k \times m}$, be an *interpretation* matrix, which represents an (admissible) grounding, where its entry $g_{i,j}$ is 1 if the domain element(s) represented by column j are grounded by the institution element(s) in row i, 0 otherwise. Similarly, let matrix $\mathbf{S}^{n \times h}$ be a *semantics indicator* matrix, where h is the number of attributes used by the semantics functions (e.g. 'must' norm uses only one attribute active). Its entries $s_{i,j}$ are assigned value 1 if the attribute in row i matches (is the same as) the attribute in column j, otherwise 0 is assigned. While matrix $\mathbf{G}$ is a function of grounding, $\mathbf{S}$ depends on the

attributes used in the semantics. The relation

$$\mathbf{GD} = \mathbf{T} \tag{1}$$

transforms $\mathbf{D}$ into structure $\mathbf{T}$. It transforms the rows of $\mathbf{D}$ to those of $\mathbf{T}$ via grounding $\mathbf{G}$, determining how domain elements 'count-as' abstract institution elements. The columns of the result $\mathbf{T}$ are the (all) attributes describing the relevant institutional elements. This is illustrated in Figure 2. Similarly to $\mathbf{D}(t)$, $\mathbf{T}(t)$ also maps time to institutional states, and ${\mathbf{T_I}}^{m\times n\times|I|}$ represents part of the full trajectory consisting of only state values of institutional elements. Still, evaluating norms semantics in this partial trajectory can be expensive since all attributes have to be stored for each time step. Fortunately, to evaluate the state of norms we need only attributes used by the semantics functions. This is achieved with the following relation: $\mathbf{TS} = \mathbf{T_S}$. Thus, the trajectory defined as ${\mathbf{T_{SI}}}^{m\times h\times|I|}$ can be used for evaluating norm states $\mathrm{ns}(q(trp^*), {\mathbf{T_{SI}}}^{m\times h\times|I|})$. For example, the result of ns evaluating the trajectory consisting of two subsequent states $\mathbf{T_S}(t_1)$ and $\mathbf{T_S}(t_2)$ may result in 'n', where evaluating it again with $\mathbf{T_S}(t_3)$ may result in 'f'. Thus $\mathcal{F}_{\mathrm{norm}}$ can produce the feedback signal (e.g., from $\{n, f\}$) needed for learning.

In general, $\mathbf{T}$ reduces the size of the state-space and provides an invariant functional structure, thus π_λ can be re-used in domains where the appropriate transform of $\mathbf{D}$ can be realized. We call *re-grounding* of an institution, changing its grounding and therefore associating the same abstract categories with a novel domain. This allows both learning abstract policy π_λ in different domains but also executing it in novel domains. Equation 1 can be applied for each time step $\mathbf{G}(t)\mathbf{D}(t) = \mathbf{T}(t)$. For the purpose of this paper, in the evaluation presented in Section 4 (Experiment 2), we will manually re-ground the institution at arbitrary time steps to demonstrate how already learned norms can be applied with different groundings. The ability to re-ground norms to other domains provides a workflow where one can train abstract policies in simulation (adapted to robotics) and then *re-ground* them to real robotic systems. The trained policy from simulation then can be updated by learning subtle details starting from an already trained (hence 'safe' from the normative point of view) policy. Some cognitive aspects of abstract learning are discussed in [24].

4 Experiments

The goal of this section is to test and evaluate the methods described so far and to demonstrate the benefits of using prior normative knowledge and institution abstraction in model free environments. Experiment 1 compares standard policy learning with learning abstracted policies. Here we hypothesize that learning will be significantly faster if the state space is reduced via abstraction. In experiment 2, we re-ground the abstract policy learned in experiment 1 on the new 'factory' domain, demonstrating the transfer of learning. Experiment 3 focuses on reward shaping and we demonstrate its applications on both standard learning and abstract learning. In the last experiment 4, we use an additional agent to demonstrate applicability of the method to multi-agent systems.

The experiments are based on the 'shopping' scenarios from real robotic competitions [2], where the typical tasks consist of picking an item from a store, deliver it to a specific place, pay for it, etc. We show how the learned abstract policies can be applied to novel domains without relearning, which can be used to address other scenarios from same competitions, e.g. waiter robots.

[2] sciroc.eu/e07-shopping-pick-pack and www.robocupathome.org

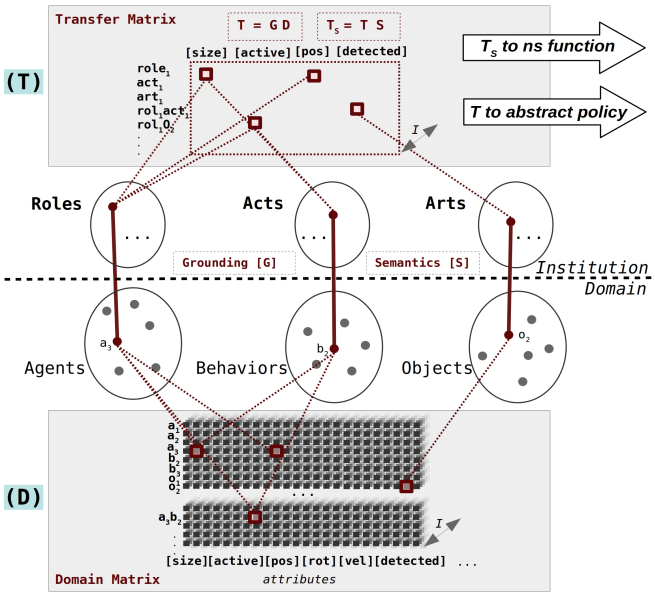


Figure 2: Illustration of equation $\mathbf{GD} = \mathbf{T}$. $\mathbf{D}$ is Domain state space, $\mathbf{T}$ is Institution state space, and $\mathbf{G}$ is interpretation matrix; States from $\mathbf{T}$ are encoded in feature vector while states from $\mathbf{T_S}$ are stored and used for 'norms state' evaluation

Videos accompanying this paper and further details to allow reproducibility, including norms semantics and all learning hyper-parameters used in the experiments, can be found at https://youtu.be/IeOihOKs25A.

4.1 Scenario

Robby is a robot that is aware of the 'Store' institution providing social knowledge on how to behave in a buying/selling scenario. Namely, Robby is aware of the following norms: An agent that wants to buy something should pick the desired object and not other objects in the store, and it should pay for the object at a designated place before leaving. The 'Store' institution consists of role Buyer, actions Pay and GetGoods, and artifacts Goods and PayPlace. Norms are specified as follows (we increase the complexity of this institution as we progress through the experiments):

$$\begin{aligned}&\mathsf{MustUse}\,((\mathsf{Buyer}, \mathsf{GetGoods}, \mathsf{Goods}))\\ &\mathsf{MustAt}\,((\mathsf{Buyer}, \mathsf{Pay}, \mathsf{Payplace}))\\ &\mathsf{Before}\,((\mathsf{Buyer}, \mathsf{GetGoods}, \mathsf{Goods}),\\ &\qquad(\mathsf{Buyer}, \mathsf{Pay}, \mathsf{PayPlace}))\end{aligned}$$

The example domain includes Robby and the set of behaviors. Robby knows how to execute 3 complex behaviors: take items, execute wireless money transfer, and open doors. Robby is also capable of primitive actions: moving one step forward and backward, and rotating one degree left or right. The domain includes 13 items in the store, e.g., a battery, a drill, an ax, screwdrivers, as well as a cash register. The experiments focus only on learning normative policies, and not on particular behaviors (such as grasping an item), thus all robot behaviors are atomic (completed in one step).

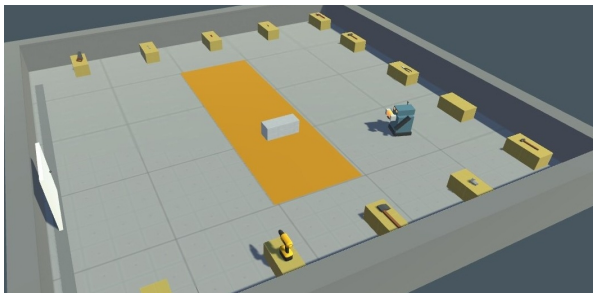


Figure 3: Simulated Store Environment

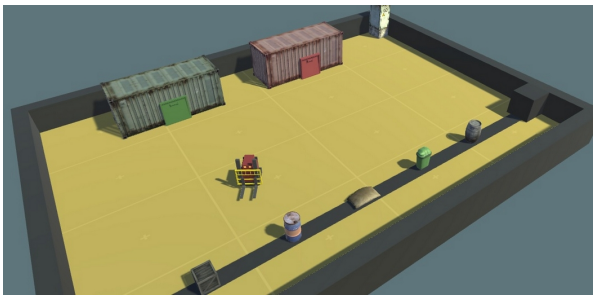


Figure 4: Factory-yard environment

4.2 Experiments Setup

Software. We use a game engine [22] to simulate a simplified store domain (see Figure 3). Learning is done with PPO [19] with the help of the Machine Learning toolkit [9] (v0.6). For all reported experiments, we used a maximum of 2000 steps per episode, and results are averaged over 10 trials (mean), where for each trial data is collected over 16 parallel simulations. In addition to the rewards regarding norm adherence, the agent receives small negative feedback ($1.0e-4$) for each simulation step to ensure that agents finish episodes as soon as possible. In all experiments, we used 3-layer neural network with 256 hidden units and the same hyper-parameters (see Table 1).

beta	gamma	lambd	learning_rate	hidden_units	time_horizon	batch_size
6e-3	0.99	0.95	3e-4	256	1024	1024

Table 1: Hyper-parameters used for training in the experiments. The full list of parameters is available at https://youtu.be/IeOihOKs25A

Encoding. In the standard learning approach, the policy is associated with a particular agent and its observable states (states available through agent sensors) are encoded from $\mathbf{D}$. The agent observes the state through 7 rays pointing in different directions, each ray encoding information about which object is detected (or none) and its distance. Given the 16 objects in the environment, this is encoded as a feature vector of length $(16 + 2) \times 7$. Additional states indicate if an object is 'near' or if the agent is 'holding' an object, activation of behaviors, and the velocity vector, making the total length of feature vector 158 elements. Action space consists of all 7 introduced behaviors. In the abstract case, a policy is learned for a particular *role*, where observable states are encoded based on the states of institutional elements from $\mathbf{T}$. For instance, the lenght of feature vector from 7 ray is reduced to 24 elements, since now rays detects *artifacts*. The action spaces consist of 2 behaviors grounded to $Acts$. Still, in addition to this, we include 4 motion primitive actions. This limits the policy abstraction to only normative aspects of behaviors and consequently it limits re-grounding of agents with substantially different motion primitives. The total size of the feature vector is significantly reduced to 39 elements.

Hardware. The simulations are executed on a desktop computer with the following configuration: Intel Core i7 4790K CPU @ 4GHz (x64), 16GB DDR3 RAM, Nvidia GTX970 (4GB GDDR5).

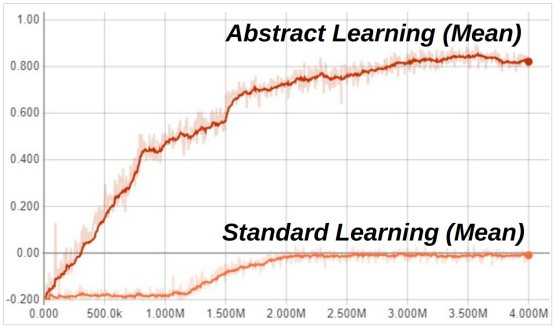
4.3 Experiment 1: Proof of Concept

The goal of this experiment is to learn the normative behavior specified by the Store institution. For training, we use feedback from full adherence in two different settings: (A) Standard learning, and (B) Abstract Learning. Groundings are given as follows: $\mathcal{G}_a = \{Buyer, Robby\}$, $\mathcal{G}_b = \{GetGoods, pick\}$, $\mathcal{G}_b = \{Pay, transfer\}$ and $\mathcal{G}_o = \{Goods, battery\}$. The hypothesis is that the learning will be significantly faster in setting (B) since the state-space is significantly reduced. Results confirm our hypothesis (see Figure 5a). In setting (A) the agent failed to learn the normative policy in almost all of the trials, whereas in setting (B) almost all training trials managed to converge to fully adherent behavior. Note that the only reason we can compare standard and abstract policy is that we have used the same grounding for both of them. In the following experiment, we investigate the case when abstract policy is used with a grounding other than that used for learning.

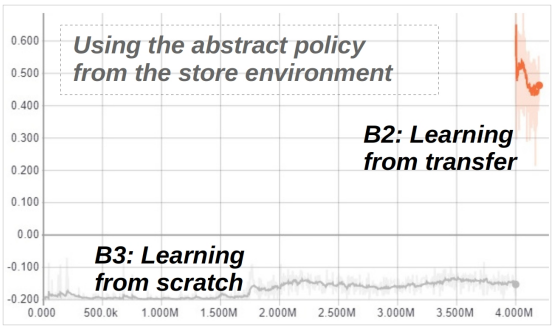
4.4 Experiment 2: Transfer of Learning

The goal of this experiment is to measure the ability of the approach to enable transfer of learning. In scenario (A) we test a new grounding in the same domain: $\mathcal{G}_o = \{Goods, drill\}$. We expect that the agent will know to buy the drill instead of the battery since now the drill is abstracted to 'Goods'. It is important to stress that while humans may assign different names to roles, actions or artifacts over different domains, the pattern of agent behaviors may still have the same semantics. Scenario (B) demonstrates this concept, where we use a factory-yard domain. A robot named Forky is required to sort out different items by locating them on a conveyor belt, lifting them and bringing them to a container's hatch for disposal. We ground the Store institution to the new domain as follows: $\mathcal{G}_a = \{Buyer, Forky\}$, $\mathcal{G}_b = \{GetGoods, lift\}$, $\mathcal{G}_b = \{Pay, leave\}$ and $\mathcal{G}_o = \{Goods, box1\}$. Forky is twice as slow than Robby, the size of the environment is increased, and the spatial layout is somewhat different (see Figure 4). Additionally, items that Forky is required to lift are moving, and sometimes they are not in the environment at all (they appear on the conveyor belt), which can additionally confuse the agent. In the trials, we distinguish between (B1) directly applying the abstract policy learned in the Store scenario, (B2) where we continue training of the transferred policy in the new domain for an additional 200k steps; and (B3) where we train the agent from scratch.

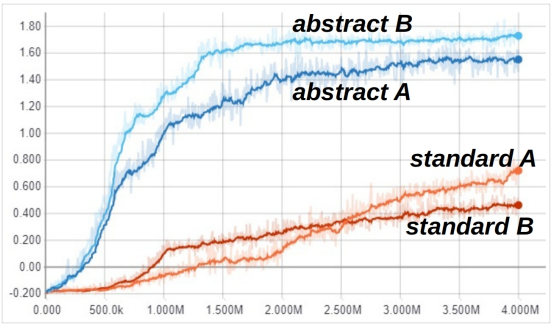
Results show that in the trial (A), Robby successfully locates the drill (instead of the battery), picks it up and goes to the cash register to pay. In scenario (B), with the original store policy (case B1), Forky manages to search for and navigate to the grounded factory item (box), lift it, and in most of the episodes manages to reach the hatch to dispose of the item. Figure 5b shows that continuing to train the policy (B2) did not improve agent behavior, while learning from scratch (B3) did not manage to achieve any significant results within 4M steps.



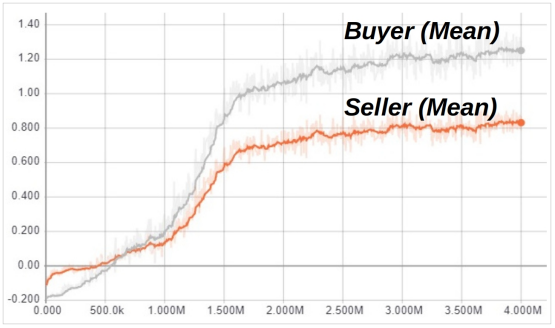
(a) Experiment 1: Proof of Concept



(b) Experiment 2: Transfer of Learning



(c) Experiment 3: Normative Feedback



(d) Experiment 4: Multiple Agents

Figure 5: Summary of Results; 'Cumulative Reward' is shown on the vertical axis and the 'Number of Steps' on the horizontal axis and shaded area is 95% confidence interval. Results are averaged over 10 trials (mean), where for each trial data is collected over 16 parallel simulations

4.5 Experiment 3: Normative Feedback

Figure 5a shows that standard learning was unable to synthesize satisfactory policies in most cases. The goal of this experiment is to test learning based on feedback from evaluation of individual norms and to assess whether it can more effectively guide an agent towards a normative policy. The temporal norms are hardest to learn since they directly introduce the (temporal) credit assignment problem. We make the problem more challenging (compared to experiment 1) by adding an additional temporal norm stating that the buyer, after paying, is expected to exit the store through the door. The hypothesis is that Robby should learn the policy in a more complex environment even without institution abstraction, while the abstraction will lead to even faster learning. We test two approaches: (A) if any norm is violated, all future rewards are stopped, and (B) if any norm is violated the episode is restarted, and an amount of negative feedback is assigned to the learning agent.

The approaches (A) and (B) are applied to standard (full-state space) learning and abstracted learning (see Figure 5c). Even with additional temporal requirements, the results are significantly improved compared to experiment 1. Sill, in standard learning, 4M steps were not sufficient for all trials to reach fully adherent policies.

4.6 Experiment 4: Multiple Agents

While it is possible to learn one abstract policy to guide all the agents in an institution, in this experiment each agent learns its own abstract policy. A known problem in MAS is how to determine the right 'reward structure' [6] so that agents are rewarded appropriately to their contribution to achieving a shared task. Our approach allows us to address this problem by breaking social space into norms. Then the rewards are distributed in the following way. The acting of all agents together creates a single trajectory which can be either adherent to an institution or not, thus the final reward is given to each agent that is grounded by the institutional roles. Norms of a particular agent depend on its role, thus each agent gets feedback depending on the norm it is related to. Some norms include more than one role: in such cases, agents have to cooperate, so that if a norm is satisfied, they both get the same reward, and no reward otherwise.

In this experiment, we use an additional robotic agent named 'Kobby', which is capable only of navigating, receiving, and accepting/declining payments from nearby agents. The institution now includes the additional role of Seller, and action ReceivePayment, with additional norms indicating that the seller has to receive payment at the place of payment and a temporal norm 'equals' ensuring that agents synchronize their behaviors. This means that the agents grounded to the roles of Seller and Buyer have to cooperate to achieve adherence, i.e., 'Paying' by one agent should be done at the same time as 'ReceivingPayment' by another agent. The 'Buyer' has to fulfill more norms than the 'Seller', which results in the difference in cumulative reward. Learning trials for both agents converge to fully adherent policies (see Figure 5d).

4.7 Summary of Results

Experiment 1 reveals that learning normative behavior using only adherent feedback is possible, but does not scale to complex settings. Much better results are achieved with learning abstract policies. Experiment 2 demonstrates that abstraction enables us to apply policies to novel domains, even when the policy in a novel domain is very hard to learn from scratch. Experiment 3 demonstrates automatic normative reward feedback, where the best results are achieved when combining normative feedback and abstraction. Finally, in Experiment 4, we show that it is possible to learn policies for coordinating

multiple agents.

5 Conclusions and Future Work

In this paper we showed how (1) behavior rules can be learned from automatic *shaping* of rewards from prior normative knowledge, addressing the credit assignment problem and how (2) given an admissible grounding, we can automatically and significantly *reduce* the size of the state-space and action space, addressing the curse of dimensionality problem. (3) Learned *schematic* behavior in abstract policies can be applied (via re-grounding) to novel domains, thus achieving a transfer of learning.

As an additional contribution (4), we provided Equation 1 which *operationalizes* (defines) relations between concrete domains and abstract reusable functional structures. The iterative nature of this equation: $\mathbf{G_2}(t)[\mathbf{G_1}(t)\mathbf{D}(t)]$ suggests that behaviors can be learned hierarchically, from primitive functionalities to social behaviors or used for explaining complex behaviors through primitive ones. The abstract policies are the state-transition function, hence they can be formalized as operators (or *simulators* [3]) in the context of task planning and then re-used for deliberate reasoning hierarchically, by learning or reason about *grounding*. This is left for future work. In addition, abstract learning should be further explored through the lens of cognitive science: "considerable empirical evidence suggests that structural representation pervade human knowledge" [3].

Acknowledgments. This work has been partly supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 825619 "AI4EU".

REFERENCES

[1] T. Ågotnes, W. van der Hoek, J. A. Rodríguez-Aguilar, C. Sierra, and M. J. Wooldridge, 'On the logic of normative systems', in *Procs of the Int Joint Conf on Artificial Intelligence (IJCAI)*, pp. 1175–1180, (2007).

[2] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al., 'Learning dexterous in-hand manipulation', *The Int Journal of Robotics Research*, **39**(1), 1177–1187, (2020).

[3] L. W. Barsalou, 'Abstraction in perceptual symbol systems', *Philos. Trans. of the Royal Society of London. Series B, Biological Sciences*, **358**(1435), 1177–1187, (2003).

[4] R. E. Bellman, *Adaptive control processes: a guided tour*, Princeton university press, 2015.

[5] O. Boissier, J. F. Hübner, and J. S. Sichman, 'Organization oriented programming: From closed to open organizations', in *Engineering Societies in the Agents World VII*, eds., G. O'Hare, A. Ricci, M. O'Grady, and O. Dikenelli, pp. 86–105, (2007).

[6] G. Chalkiadakis and C. Boutilier, 'Coordination in multiagent reinforcement learning: A bayesian approach', in *Procs of the Int Joint Conf on Autonomos Agents and Multi-Agent Systems (AAMAS)*, pp. 709–716, (2003).

[7] G. De Giacomo, L. Iocchi, M. Favorito, and F. Patrizi, 'Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications', in *Procs of the Int Conf on Automated Planning and Scheduling (ICAPS)*, pp. 128–136, (2019).

[8] A. García-Camino, J-A Rodríguez-Aguilar, C. Sierra, and W. Vasconcelos, 'Norm-oriented programming of electronic institutions', in *Procs of the Int Joint Conf on Autonomos Agents and Multi-Agent Systems (AAMAS)*, pp. 670–672, (2006).

[9] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, 'Unity: A general platform for intelligent agents', *arXiv preprint arXiv:1809.02627*, (2018). https://github.com/Unity-Technologies/ml-agents.

[10] J. Li, F. Meneguzzi, M. Fagundes, and B. Logan, 'Reinforcement learning of normative monitoring intensities', in *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pp. 209–223, (2015).

[11] M. J. Matarić, 'Learning in behavior-based multi-robot systems: Policies, models, and other agents', *Cognitive Systems Research*, **2**(1), 81–93, (2001).

[12] M. L. Minsky, E. A. Feigenbaum, and J. Feldman, *Computers and Thought*, McGraw-Hill, 1963.

[13] A. Y Ng, D. Harada, and S. Russell, 'Policy invariance under reward transformations: Theory and application to reward shaping', in *Procs of the Int Conf on Machine Learning (ICML)*, pp. 278–287, (1999).

[14] A. Y. Ng and S. Russell, 'Algorithms for inverse reinforcement learning', in *Procs of the Int Conf on Machine Learning (ICML)*, p. 2, (2000).

[15] D. C. North, *Institutions, institutional change and economic performance*, Cambridge university press, 1990.

[16] E. Ostrom, *Understanding institutional diversity*, Princeton university press, 2009.

[17] L. Y. Pratt, 'Discriminability-based transfer between neural networks', in *Procs of Advances on Neural Information Processing Systems (NIPS)*, pp. 204–211, (1993).

[18] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, 2014.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, 'Proximal policy optimization algorithms', *arXiv preprint arXiv:1707.06347*, (2017).

[20] J. R. Searle, 'What is an institution', *Journal of institutional economics*, **1**(1), 1–22, (2005).

[21] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., 'Mastering the game of 'go' without human knowledge', *Nature*, **550**(7676), 354, (2017).

[22] Unity Technologies. https://unity.com/, Accessed June 2019.

[23] S. Tomic, F. Pecora, and A. Saffiotti, 'Norms, institutions, and robots', *arXiv preprint arXiv:1807.11456*, (2018).

[24] S. Tomic, F. Pecora, and A. Saffiotti, 'Robby is not a robber (anymore): On the use of institutions for learning normative behavior', *arXiv preprint arXiv:1908.02138*, (2019).

[25] R. S. Woodworth and E. L. Thorndike, 'The influence of improvement in one mental function upon the efficiency of other functions', *Psychological review*, **8**(3), 247, (1901).