Formalising Learning from Demonstration

Erik A. Billing billing@cs.umu.se
Thomas Hellström thomash@cs.umu.se

UMINF 08.10
ISSN-0348-0542

Department of Computing Science
Umeå University
Umeå, Sweden

June 2008

***Abstract*—The paper describes and formalizes the concepts and assumptions involved in *Learning from Demonstration* (LFD), a common learning technique used in robotics. Inspired by the work on planning and actuation by LaValle [31], common LFD-related concepts like *goal, generalization*, and *repetition* are here defined, analyzed, and put into context. Robot behaviors are described in terms of trajectories through information spaces and learning is formulated as the mappings between some of these spaces. Finally, behavior primitives are introduced as one example of useful *bias* in the learning process, dividing the learning process into the three stages of *behavior segmentation*, *behavior recognition,* and *behavior coordination*.**

***Index Terms*—Action selection, Behavior, Bias, Generalization, Goal, Learning from Demonstration, Robot Learning, Segmentation**

I. Introduction

Learning From Demonstration (LFD) is a well established technique for teaching robots how to perform useful tasks. The basic idea is that the robot learns to repeat a behavior after being teleoperated through one or several demonstrations performed by a human teacher. The research area is attractive, both in its intuitive approach to human robot interaction and as a framework for a theoretical analysis of knowledge representation and transfer of knowledge between intelligent agents.

Research on LFD is influenced by a variety of fields, such as control theory, artificial intelligence, psychology, ethology, and neuro physiology. While primarily being a big asset, the multidisciplinary nature of LFD also contributes to the lack of a unified formalism for the different components constituting the research field. Furthermore, it should not come as a surprise that the terminology used differs for works conducted by researchers from varying areas. In this paper, we are trying to identify, define, and formalize the common basic ideas and principles involved in LFD. The presented work is both a survey of how these concepts are used in research, and an attempt to describe them in the light of related concepts in machine learning, planning, and psychology. To our knowledge this has not been previously done in a unified way and the result can be used both as a somewhat theoretical introduction to the field and as framework for further development and research.

The approach is inspired by the work on planning and actuation by LaValle [31] and therefore does not always follow the terminology and notation found in common literature on LFD. Where this is the case, we clearly point it out and also refer to the commonly used terms.

First, a few basic concepts that form the basis for the formal description of the learning process are introduced.

II. Basic concepts

A. State space

One fundamental component in classical AI is the concept of a *state space* X, described by a *world ontology* [54, pp. 222]. The state space can be defined as a set of all possible situations that could arise in the world [31]. More specifically, the state space only includes the *relevant* aspects of the world, given a certain task or limited set of tasks. However, if the task is unknown it is very difficult to identify which aspects of the world are relevant. One could of course try to include all aspects that might be of interest, but even if possible, that would result in a huge and complex space, implying tremendous sensing requirements when applied to a field such as LFD. Furthermore, defining a state space introduces many unnecessary assumptions about the world, and requirements for information which make the problem much more complex than necessary. This observation is nicely illustrated by Simons' ant [57] and is also related to the classical frame problem [40], [29].

For these reasons, it is desirable to create new spaces, less task-specific and sensor-demanding, in which behaviors can be represented. Such a redefined representation is referred to as an *information space* [31]. Interestingly, the concept of information spaces is also common within LFD, but appears under different names. In order to facilitate learning, approaches to LFD often utilize so called *primitives* or *skills*. These primitives can be seen as building blocks from which more complex behaviors can be composed, which results in

moving the learning process away from the state space into a new representational space composed of the available skills, e.g. [22], [48], [43], [6], [30], [46]. Many of these approaches relate strongly to *Behavior Based Control* (BBC) [36], [37], [3]. BBC has its roots in the reactive paradigm, but emphasizes parallel, loosely connected *behaviors* for control of the robot as an emergent property, rather than a single stimuli-response loop.

We further investigate the possibilities of applying the concept of information spaces within LFD, but first a few other basic concepts have to be introduced.

B. Sensing and acting

Imagine an agent interacting with the environment. It perceives the world through its sensors and acts upon the world with its actuators. The sensors are defined as a function $h : X \rightarrow Y$ transforming a certain state $x \in X$ into a sensor state $y \in Y$ [31]. Y denotes the *observation space*, i.e., the set of all possible readings returned by the agent's sensors. Note that each $y \in Y$ is a vector (y(1), y(2), ...) comprising simultaneous values from all sensors. Typical examples are a thermometer that maps physical temperatures x to numbers $y(1) \in \mathbb{R}$ or a GPS receiver that maps physical positions to latitude and longitude, i.e. $y(1) \in \mathbb{R}^2$. Y corresponds to the *stimulus domain* in behavior-based robotics [3].

On the actuator side, actions can be said to transform a certain state into another state. Hence, actuators implement the function $f : X \times U \rightarrow X$ where U denotes the *action space*, i.e., the set of all possible actions the agent can execute. A typical example is the requested velocity for each motor of the robot. Note that this does not specify the actual motor velocity, and only the outgoing information is represented in U. The actual velocity is normally represented in state space X.

Now a description of how the agent behaves, i.e. generates actions, can be introduced. In general, such a description is referred to as a *controller*, but is also known as a *plan* [31], *behavior mapping* [3], [18], [46], [47], or *motor primitive* [2]. Several important differences between these terms do exist, for example in terms of abstraction level and temporal extension, but for now they can all be said to implement the function π:

$$\pi : X \rightarrow U. \tag{1}$$

Hence, π maps states $x \in X$ to actions $u \in U$. As mentioned before, X is not explicitly represented in the agent. Still, the physical sensors and actuators can be said to implement the functions h and f, respectively. In contrast, π can not be implemented without an explicit definition of and access to X. To solve this issue, π is later redefined and controls the agent based on the information space instead of the state space.

C. Information space

The observation and action spaces are widely adopted by the robotics community. One control paradigm referred to as *sensory-motor coordination* (SMC), focuses on creating representations within the so-called *sensory-motor space* $I = U \times Y$ [49], [50]. In each stage k the robot experiences a *sensory-motor event* $e_k = (u_{k-1}, y_k) \in I$. The action in $k-1$ is used since u_k changes the current stage to $k+1$ [31].

From an SMC perspective, sensing and acting are not two separate processes. In contrast to classical reactive systems, SMC does not view the information flow purely as going from sensors to actuators. Actions give rise to a certain stimulus, just as much as a stimulus influences some action. If the agent can predict these relations, it can intentionally control its interactions with the world. Hence, control is seen as a problem of coordination. Similar views are common within psychology, anthropology, and cognitive science, [23], [58], [28].

The sensory-motor space I has several advantages compared to the state space. Most importantly, it is easily defined. If an agent is designed with a fixed number of sensors and actuators, the size of I remains constant independently of environment and task. Of course this limits the possibility to add new sensors or actuators to the agent without corrupting the robot's knowledge, but for many application this is a reasonable limitation. The sensory motor space also has a number of disadvantages. In contrast to state space which by definition, at each moment, contains all information necessary to make a control decision, I does not necessarily have this property. A decision, i.e., a selection of the next action, may have to be based not on the most recent sensor and motor readings, but on complex patterns of previously observed sensory-motor events. Let $\tilde{Y}_k$ denote the *history observation space*, i.e., the set of all possible observation histories $\tilde{y}_k$ until current stage k:

$$\tilde{y}_k = (y_1, y_2, \ldots, y_k) \in \tilde{Y}_k \tag{2}$$

where each vector $y_i \in Y$ is provided by the sensors at stage i. Similarly, let $\tilde{U}_k$ be the *history action space*, i.e., the set of all possible action histories until current stage k:

$$\tilde{u}_k = (u_1, u_2, \ldots, u_k) \in \tilde{U}_k \tag{3}$$

where each $u_i \in U$ is a particular action vector issued at stage i.

The histories $\tilde{y}_k$ and $\tilde{u}_k$ in combination with the preconditions η_0 form a *history information state* η_k, also referred to as an *event history*. η_k includes all accumulated information up to stage k [31]:

$$\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k) \in I_k \tag{4}$$

The history information state is a central concept in the formalism since it represents all the information the agent has received, and as a consequence η_k is always known in stage k. I_k is known as the *history information space* and should be understood as the set of all possible event histories up until stage k [31]:

$$I_k = I_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k \tag{5}$$

where I_0 represents the set of all possible preconditions.

The definition of I_k becomes impractical in cases where the number of stages is not fixed. Instead, we normally refer to the *information history space* I_{hist}, with unspecified length [31]:

$$I_{hist} = I_0 \cup I_1 \cup I_2 \cup \ldots \tag{6}$$

It is worth observing that I_{hist} is huge, it includes all possible combinations of everything the agent could possibly observe and do. Most $\eta \in I_{hist}$ will of course never be observed, due to limitations imposed by the environment and the physical shape of the robot. For example, imagine a simple robot, equipped with a proximity sensor on each of its four sides, placed in an empty large square box. In this environment, the robot never observes a y_k with high activation of all proximity sensors simultaneously. This is a simple result of physical properties of the environment and the robot itself. The same way of reasoning could easily be applied to a human agent. There is a huge amount of patterns the human senses theoretically could perceive, but that will never be observed.

Most of the formal definitions in this paper take place in history information space I_{hist}. You might ask why representations take place in such a huge and complex space when only a fraction of its representational power is actually used. I_{hist} should not be understood as *the* representational space, but *a* representational space, a very basic one. Any information the agent can acquire is representable as an event history $\eta \in I_{hist}$. Furthermore, I_{hist} is, in contrast to state space X, both well defined and completely task invariant and is as such very suitable for learning purposes. However, in many other respects I_{hist} is not the best representational space. As mentioned before, I_{hist} is huge and bears a lot of redundant information, making it difficult to extract features relevant to the specific task. For this reason, a new *derived information space* I_{der} may be created. I_{der} should be seen as a simplification of I_{hist}, where relevant features are represented, while irrelevant information is not contained, [31]. The use of derived information spaces as bias in learning is further discussed in Sections III-B and III-D.

D. Controller

The controller defined in Equation 1 can now be reformulated in a form that allows it to be used without full access to state space X :

$$u_k = \pi(\eta_k) \tag{7}$$

where $u_k \in U$ is the action vector issued at stage k and $\eta_k \in I_k$ is the agent's event history a stage k. π is defined here as a function from information history space to action space:

$$\pi : I_{hist} \rightarrow U. \tag{8}$$

In simple cases, a controller can be modeled as a function of only the most recent sensory-motor event. Systems based purely on such single-event controllers are called *reactive systems* [12]. Formally, these systems implement π as

$$u_k = \pi(y_k) \tag{9}$$

which can be seen as a special case of Equation 7. This definition of π is similar to Arkin's *behavior mapping* $\beta : S \rightarrow R$, where S and R are stimulus and response, respectively [3]. However, in the general case we use the wider definition of π given in Equation 7.

E. Behavior

The word behavior is commonly understood as an agent's actions in relation to the environment [59], but in the robotics community it has many different meanings. We would like to describe a behavior as a purposeful way of acting. This does not imply that behaviors include explicit representations of goals, but from an observer's point of view, the behavior can still be said to implement some kind of purpose, or goal. The concept of goals is further discussed in Section III-C.

Using the introduced terminology, a *behavior* B may be defined as a subset of I_{hist}:

$$B = \left\{\eta^{(1)}, \eta^{(2)}, \ldots\right\} \subset I_{hist} \tag{10}$$

where each $\eta^{(i)}$is an event history (of unspecified length). The mechanisms, programs or plans which may produce B were introduced as the controller π in Equation 7.

Often, no explicit distinction is made between the observable interactions with the world, and the mechanisms producing these interactions. However, in our terminology, Equation 10 describes nothing about how the behavior is produced, and therefore the notion of a behavior is different than the terminology commonly used within behavior-based robot architectures [3], [46], [18], [37]. As is clarified further on, this distinction serves several purposes.

III. Learning From Demonstration

Learning From Demonstration (LFD), is a well established technique for robot learning. An overview of early work is found in [5] while recent work can be found in [46]. Another excellent survey of the area can be found in [8]. The basic idea in LFD is that the robot learns to do things by observing other agents, be it human beings or other robots. Several flavors of this idea exist and the used terminology differs somewhat in published research. By *LFD* we denote in this paper learning where the other agent (often denoted *teacher*) directly controls the robot, e.g. by teleoperation or kinesthetic teaching (e.g. by manually guiding a robot arm) [14]. The recorded data from such a control session is denoted *demonstration* and the purpose of LFD is to create a controller π capable of "repeating" the demonstrated behavior. Formally, a demonstration b can be seen as an event history $\eta_k \in I_{hist}$ (refer to Equation 4) where $\tilde{u}_{k-1}$ is the sequence of actions issued by the teacher up to stage $k-1$ and $\tilde{y}_k$ is the sequence of observations up to stage k.

LFD assumes a direct correspondence between events in the demonstrations and the sensors and actuators forming the interface to controller π. I.e., the observations y_k in the demonstration are assumed to correspond to the observations that are generated in real-time by the sensors, and fed into the controller. Furthermore, the observed action variables u_k are assumed to directly correspond to the actuator signals generated by the controller π. The assumptions simplify learning

significantly, but are not valid if a teacher demonstrates a behavior by itself, and not by teleoperating the robot. In these cases, the *correspondence problem* has to be resolved as part of the learning process; which action or actions correspond to an observed sequence of events? *Imitation Learning* deals with this kind of learning scenarios. A formal description of the correspondence problem in robot and animal learning is given by Nehaniv and Dautenhan in [45]. Hereinafter we focus on LFD and thus ignore the special problems involved in solving the correspondence problem.

LFD is related to the more general terms *Programming By Demonstration* (PBD) or *Programming By Example* (PBE) but should not be confused with the aim of creating or modifying the behavior of computer programs by using demonstrations in general [17], [32]. This paper presents a formalism for robot learning through demonstration, which, while it can be seen as the creation of a specific kind of computer programs, does not apply to the wider interpretations of PBD or PBE.

The goal of LFD is to generate a controller π that enables a robot to *repeat* a demonstrated behavior B. If successful, the robot is said to have learned behavior B. Formally, the process of learning B from a set of N demonstrations b is understood as selecting a controller π from the *controller space* Π using a *learning function* λ:

$$\pi = \lambda(b) \in \Pi \tag{11}$$

where

$$b = \left\{\eta^{(1)}, ..., \eta^{(N)}\right\} \subset B. \tag{12}$$

The LFD process is illustrated in Figure 1. Normally all demonstrations $\eta^{(i)}$ are assumed to belong to the wanted behavior B, i.e. $b \backslash B = \emptyset$. Π contains all possible controllers for a specific chosen observation space and action space. This is of course a huge space that is never computed explicitly.

The selected controller π must have certain qualities for the learning to be regarded successful. These qualities are related to the event histories η that may be generated by a robot using π as a controller. The *realization space* $R \subset I_{hist}$ for a controller π is defined as the set of all such event histories, generated by the *realization function* $R = \Lambda(\pi) \in I_{hist}$.

Λ can be seen as an abstraction of the physical robot placed in a particular environment and controlled by a specific π, able to produce the set of all possible trajectories through I_{hist}. Of course, the robot can not control the produced event histories $\eta \in R$ entirely on its own, but relies on an external component, the environment. This creates a nice analogy to λ, which also relies on an external component, called bias. Thus the learning function λ can be seen as the inverse function of the robot represented by Λ. λ maps a set of event histories to a controller and Λ maps a controller to a set of event histories. This discussion is further developed in Section III-B.

The process of learning π has many similarities to system identification, where a model of the system is constructed from observed input and output data [33]. The system, consisting of the agent and its environment, is modeled such that the system output y_{k+1} can be predicted given a sequence of previous inputs and outputs η_k until stage k. However, the aim of system

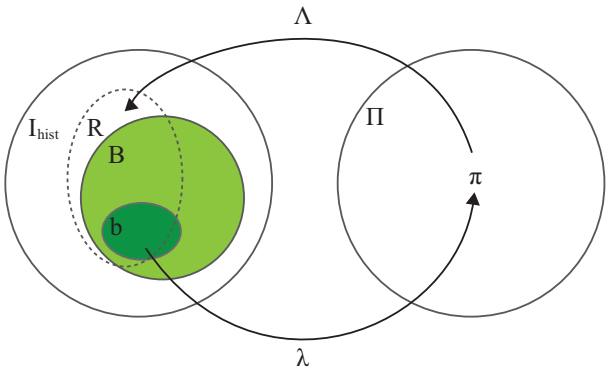

Figure 1. The LFD process. The light colored area represents the wanted behavior B which is demonstrated with N training demonstrations $b = \left\{\eta^{(1)}, ..., \eta^{(N)}\right\} \subset B$ represented by the dark colored area. The learning function λ creates a controller $\pi \in \Pi$. In interaction with the environment, π realizes (repeat) the learned behavior. The realization set $R \subset I_{hist}$ is marked by the dashed line.

identification is in one sense much more ambitious than LFD, since the system's response to any input y_k is to be predicted. In LFD, we are satisfied with a π producing an action that, if possible, leads to an event sequence $\eta_{k+1} \in B$ given that $\eta_k \in B$. In other words, LFD does not necessarily model the outcome of all possible actions u_k in each state, only the ones that can occur for the robot in a particular environment.

It is important to realize that B is normally not explicitly defined. Instead, it should be understood as the set of event histories the human demonstrator associates with a certain desired behavior. E.g. if the demonstrator wants to teach the robot to move to the door, B would contain all *acceptable* event histories where the agent ends up by this door.

The quality of the generated π is typically described as the ability to "repeat a behavior", which is the topic of the next section.

A. What does it mean to repeat a behavior?

As been mentioned a few times already, the goal of LFD is to generate a controller π that enables a robot to *repeat* a demonstrated behavior B given a set of demonstrations b. This may sound like a well defined mission, but is actually both vague and ambiguous. Consider the following example of a, seemingly trivial, demonstration.

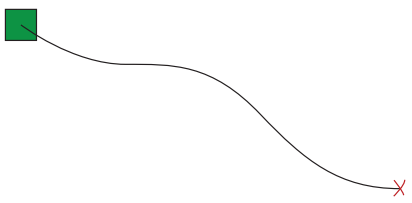

Figure 2. A simple demonstration where the tip of a robot arm starts at the red cross in the lower right corner and moves over the table until it is positioned over the green cube. The demonstration can be interpreted in a number of fundamentally different ways.

A robot arm is moving over a table, and stops when positioned above a green cube (See Figure 2).

What does it mean to repeat the sequence of events described above? One could imagine a vast number of interpretations. Here are a few examples.

1) Assuming that the path is the important aspect of the demonstration, a successful controller may be written as $u = \pi_{PATH}(y)$ where the function π_{PATH} computes an action u for each pose y, such that the arm follows the demonstrated path. This kind of learning scenario refers to traditional programming of industrial robot arms, as well as path-tracking autonomous vehicles [27].
2) Instead, if the demonstration is seen as an example of how to reach the final position, the path itself becomes irrelevant and the controller described above would not be suitable. In this case, a successful controller could be written as $u = \pi_{TARGET}(y)$ where the function π_{TARGET} uses inverse kinematics to compute actions such that the *tip* of the robot arm reaches the target.

The interpretations in Example 1 correspond to what is often called *action-level imitation* [13] where the robot carries out the "same" actions as the demonstrator. The interpretations in example 2 are often called "functional imitation" [20] in which the robot is supposed to achieve the same *effect on the environment* [44]. One could of course imagine a vast number of other interpretations. Should the observed sequence of positions be understood as fixed coordinates, or relative to the robot arm's starting position? Is the green cube really the relevant target, or is the target defined by an absolute position? Is the target a cube of any color, or maybe the target is any green object? Using many demonstrations of the same behavior clearly reduces some of the ambiguity, but in general it is impossible to tell which interpretation is "correct" without further information.

The discussion about what it means to repeat a behavior gets further complicated when the robot acts in a dynamic, non-deterministic and partially accessible [54, chapter 2] environment. Demonstrated event sequences may be both incomplete and contain mistakes that should not be learned or repeated [19].

If the robot manages to successfully repeat a demonstrated behavior under different conditions than during the demonstration we say that the robot is able to *generalize* the demonstrated behavior. More specifically, we refer to the robot's ability to produce an event history $\eta_k \in B$, under conditions (η_{k-1}) not identical to the ones appearing during the demonstrations in b. This can be formally described as how well the realization space R corresponds to the wanted behavior B, e.g. as a minimization of $R \cap B^c$ and $B \cap R^c$, or equivalently $R \backslash B$ and $B \backslash R$ (refer to Figure 1).

Generalization can also be viewed as an extension of b by interpolation or extrapolation of the demonstrated event histories $\eta^{(i)}$. For this to work one has to specify the aspects of the demonstrated data that are important. This may be done by introducing a *metric of imitation performance* [45], [1], [8]. Repeating a demonstration means minimizing the distance between the demonstrations and the repetitions using this metric. To find the metric, the variability in many demonstrations is exploited such that the essential components of the task can be extracted. One promising approach to construct such a metric is to use the demonstrations to impose constraints in a dynamical system [24], [15].

Either way we describe it, generalization is a tough challenge, and the problem is well acknowledged also outside the robotics community. In *Machine Learning*, the term *generalization performance* of a learning algorithm relates to "its prediction capability on independent test data" [25, pp.193] which is identical to the common usage of the term in robotics. The general problem with machine learning in high-dimensional spaces is often expressed as *the curse of dimensionality* [21, pp.170], and is highly relevant also for robots with high-dimensional observation and action spaces. Learning in such situations becomes inherently difficult since the demonstrated data fills history information space very sparsely and interpolation and extrapolation become highly risky operations. The situation is related to the *No Free Lunch Theorem* [60], which states that for a large class of machine learning algorithms, there is no universal best algorithm to solve all problems. Instead, an algorithm has to be specialized to the problem under consideration to guarantee its superiority over any random algorithm. This specialization consists of additional task-dependent information that has to be supplied to the learning algorithm as *bias*. In the case of LFD, possible sources of bias are the robot's prior knowledge, feedback from the environment when the robot tries to repeat the demonstrated behavior and human feedback before, during, and after learning. The bias concept is further investigated in the next section.

B. Bias

The bias of a machine learning algorithm is defined as "any basis for choosing one generalization over another, other than strict consistency with the observed training instances" [41]. I.e., if we want to do anything but record and replay a demonstration, bias has to be applied. The "basis" may be seen as form of pre-evidential judgment, or "prejudice" regarding the structure of the data or the data generating process. In the case of numerical regression, assuming a linear relation between input and output corresponds to a high bias, while a cubic model corresponds to a lower bias. In the case of LFD, bias can be applied to three different parts of the problem definition:

1) Sensor variables. This can involve selection of relevant sensors, or extraction of specific features that are judged relevant for the specific task. It may also involve creation of intelligent sensors to facilitate feature extraction.
2) Action variables. Most often this involves restricting the output of the policy function π to one or a few actuators. E.g. when learning a grip operation, the actions for moving the robot may be regarded irrelevant while the gripper motion is highly relevant. This reduces the size of action space
3) Controller function π. Bias can restrict the functional form of π, e.g. to an artificial neural network of a certain size and architecture. Bias can also be expressed as general requirements of π, such as smoothness criterion or lower/upper bounds. The use of predefined skills as described below is another example.

Bias can be introduced into the learning process in a number of ways. First of all, it may be hard-coded into the learning algorithm, e.g. by choosing a specific neural network [35] or rule based framework [26] to represent π. Another common and very powerful technique to introduce bias is to use predefined skills or behavior primitives. Besides being biologically motivated [42], [56], the technique is commonly used in robotics research, e.g. [39], [38], [22], [46]. Learning is in this case reduced to selection of the right primitives and parameter estimation to adjust the primitives to the demonstrated data. The introduction of primitives is a way to reduce the dimensionality of the learning problem (i.e. to deal with the *curse of dimensionality* mentioned above). The set of allowed primitives is obviously much smaller than Π which clearly simplifies learning. An analogy is numerical regression with a large feedforward neural network compared to a low-level polynomial. The polynomial introduces bias that makes learning much easier, at the price of limiting the solution to the specific functional form of the bias.

Regarding bias for sensors and actuators, it is common to hard-code a set of relevant sensors and action variables for the task at hand, or to pre-process the data before feeding it to the learning algorithm. For a multi-modal robot with lots of sensors, this is essential bias to make learning possible at all. This kind of bias may also be introduced by interaction with the human teacher who tells the robot to use certain sensor modalities (e.g. "Use the camera!"), or to look out for certain sensor features (e.g. "Look out for a red ball!"). Bias may also be subject to meta learning, e.g. such that suitable sensors are selected based on demonstrated data. This relates to *attention* and *saliency* which are important concepts in theories for human and animal learning. The term *shared attention* refers to a teacher's and a learner's simultaneous attention to the same objects. Scassellati used the Cog platform [55] to investigate shared attention between humans and robots. Saliency refers to the components of the environment that are important for a given task, and it clearly introduces a bias by reducing the size of observation space Y. Breazeal and Scassellati, [10] describe the relation between attention and saliency and how the concepts can be used to facilitate learning in robotics. In psychology, the term *scaffolding* is often used to denote interaction between caretakers and infants in order to reduce distractions, marking a task's important attributes and reducing the number of degrees of freedom in the learning task in general [61], [11]. All these operations aim at simplifying the learning task by introducing bias to the problem definition.

From a formal perspective, bias regarding sensor and action variables may be introduced by moving away from I_{hist} into a new, derived information space I_{der} [31]. As mentioned in Section II-C, I_{der} is a reformulated or pre-processed version of the information in I_{hist}. The mapping from I_{hist} to I_{der} is denoted κ, and may have an arbitrary shape. Therefore, I_{der} does not serve as a general purpose representational space as I_{hist} does, but rather as a task-specific representation where relevant features become salient, while irrelevant information is not retained. The observant reader notices that the purpose of I_{der} looks very similar to the purpose of the state space X. In fact, a state space is one possible definition of I_{der}, but there are numerous other possible derived information spaces that do not aim at representing states in the world.

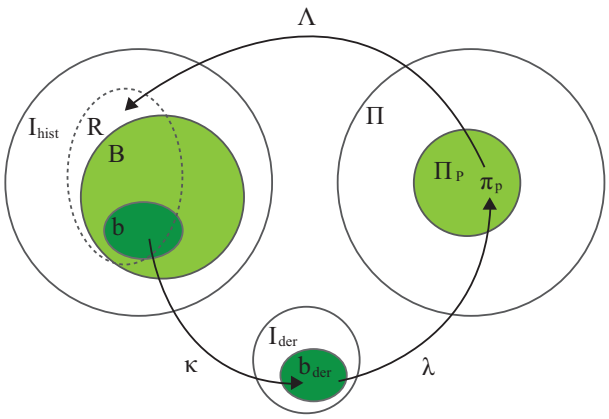

Figure 3. The LFD process with bias introduced. A derived information space I_{der} is introduced as a space where the behavior may be represented in a task-specific way. Training data b is mapped into I_{der} with an information mapping κ. The pre-processed information in I_{der} and various ways to introduce bias in λ result in a reduced set of possible controllers Π_P, illustrated by the light colored area in Π. Compare with Figure 1.

The LFD process with bias included is illustrated in Figure 3. Various ways to introduce bias regarding the control function π result in a reduced set $\Pi_p \subset \Pi$. The learning function λ maps from the derived information space I_{der} instead of straight from I_{hist}. This extended formulation of LFD is further discussed in Section III-D.

Most of the discussion here is focused on knowledge intentionally introduced into the system to facilitate learning. We like to refer to this kind of information as *ontological bias*. However, there are also a vast number of restrictions to the problem introduced for other reasons. As mentioned before, selecting a certain type of algorithm to represent π will introduce bias. A certain configuration of the robot's sensors and actuators restricts the ways in which it can solve a certain task. Often the choice of physical platform and software architecture is made for practical reasons rather than for an understanding of ontological implications. This kind of restrictions we like to phrase as *pragmatical bias*.

As mentioned above, using pre-defined skills or behavior primitives is a common way to define Π_p. The demonstrated data is in such cases used to identify a suitable primitive and then possibly tailoring it by adjusting parameters or set values. One way to define such primitives is to associate them with achievement of specific *goals*. This concept deserves special attention and is analyzed further in the next section.

C. Goal

The success or failure to repeat the demonstrated behavior is most often judged by the human demonstrator, and to describe the human intentions we use the word *goal*. The goal of a behavior is a human concept, but a lot can be gained if this information is transferred to the robot. This bias is essential for the learning process in general and for the generalization from demonstrated data in particular. The goal of a simple behavior, can be of two major types [46]:

1) Maintenance goals. A certain condition has to be maintained for a time interval, such as the path-tracking scenario described in Example 1 in Section III-A.
2) Achievement goals. A certain condition has to be reached, such as the motion to a green cube in Example 2 in Section III-A.

One reason for introducing the concept of behavior is that B gives a description of the intentions for a certain sequence of events $\eta \in B$. This can be understood as after performing B, some conditions in the world are satisfied. This is analogous with the common goal formulation from classical AI, where a goal G is a set of n states in state space [54]:

$$G = \{x_1, x_2, \ldots, x_n\} \subset X \tag{13}$$

All information the agent acquires about G is accumulated over time in $\tilde{y}$ and $\tilde{u}$. Therefore, any goal G which can be measured with the agent's sensors can also be formulated as a set of event histories in I_{hist}:

$$G_I = \left\{\eta^{(1)}, \eta^{(2)}, \ldots\right\} \subset I_{hist} \tag{14}$$

This should be understood as after observing an $\eta \in G_I$ we know that G is satisfied. A consequence of this formulation is that behaviors and goals are described in the same way, and since any $\eta \in B$ by definition satisfies the goal of B, G_I and B become identical:

$$G_I = B \tag{15}$$

Note that this goal formulation is more general than the original definition of G (Equation 13). This is both an advantage and a disadvantage. At the same time as it is convenient to formulate both goals and behaviors in the same way, some of the points with defining goals are lost. In state space, G works as a least common denominator, a neat formulation that describes the minimum requirements. Even though G is implicitly represented in B, B does not serve as the same stripped goal formulation and it is therefore very difficult to compare two event sequences $\eta^{(1)}$ and $\eta^{(2)}$ to see if they satisfy the same G. Still, if we know that both $\eta^{(1)}$ and $\eta^{(2)}$ are members of the same B, they satisfy the same G_I. What this G_I corresponds to in the world is of course still not known, and not necessarily explicitly described.

The reason we are still talking about goals is primarily that it is a natural concept for humans, and for that reason it is an important concept in LFD. In many learning situations such as in the examples in Section III-A, all information is simply not present in the demonstrated data. The missing information has to be transferred to the robot, one way or another, and the specification of a goal often contains the necessary information. The teacher's understanding of goals should be seen as a bias and may be represented in many different ways, as described in the previous section.

D. Learning

Based on the concepts of behaviors, bias, and goals introduced above, we now refine the definition of the learning task defined in Equation 11. In Section III-A it was concluded that λ requires some bias to be able to find a suitable controller, as illustrated in Figure 3. In the most basic form of LFD, λ is simply learned by fitting the demonstrated data to a more or less general functional form, such as a neural net [35] or a rule base framework [26] which in such case represents the reduced controller set Π_P in Figure 3. The use of primitives, which was introduced in Section II-A, is fully compatible with this description of learning bias such that learning consists of matching a demonstration with a pre-defined primitive. This process is normally denoted *behavior recognition* and can be approached in a number of ways as described below.

The description of LFD given above is valid for demonstrations of behaviors that can be repeated by choosing one single primitive. More complex behaviors demand sequences or combinations of primitives. For a given robot and class of learning scenarios, the set of primitives Π_P is normally chosen such that a demonstration may be divided into segments where each segment can be repeated by choosing the right primitive. The general LFD process illustrated in Figure 3 is here extended to include handling of such sequences.

Let us first look from a post learning perspective, at how sequence control can be described for a robot using a set Π_P of predefined primitives π_p. To include the assignment of parameters for parameterized primitives into the learning, Π_P is in the following regarded as containing all possible parametrization of primitives. Control can now be divided into two steps:

1) Action selection where a function π_{sel} selects a primitive $\pi_p \in \Pi_P$:

$$\pi_p = \pi_{sel}(\eta_{der}) \tag{16}$$

where π_{sel} performs the mapping

$$\pi_{sel} : I_{der} \rightarrow \Pi_P \tag{17}$$

$\eta_{der} \in I_{der}$ is a pre-processed or derived version of the original event history $\eta \in I_{hist}$, constructed by an information mapping function κ [31]:

$$\kappa : I_{hist} \rightarrow I_{der} \tag{18}$$

2) Low-level control using the chosen controller π_p to generate an action u_k.

Stepping back to the learning phase, the problem is now reduced to finding the action selection function π_{sel} using demonstrated data b pre-processed with the information mapping κ into the derived information space I_{der} (see Figure 4)[1].

While the approaches to sequence learning with primitives vary widely, the process of finding π_{sel} is often divided into three tasks:

[1]By comparing Equations 16 and 17 with Equations 7 and 8, the primitives π_p may be seen as a generalized actions, generated by a controller π_{sel}. Another interesting analogy can be made between action selection and the correspondence problem, i,e, the problem of finding the action(s) that corresponds to an observed event sequence. Viewing the primitives as actions leads to an equivalent problem formulation for action selection; find the primitive that corresponds to a an observed event sequence.

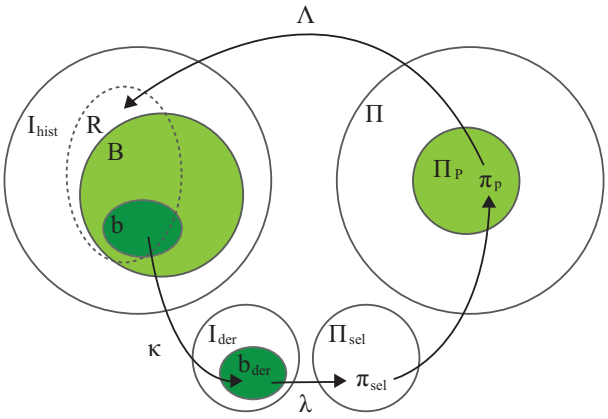

Figure 4. An extended version of the LFD process illustrated in Figure 3. Bias is here introduced into the learning process by restricting Π to a set of primitives Π_P, illustrated by the light colored area in Π. Primitives π_p are selected by selection function $\pi_{sel} : I_{der} \rightarrow \Pi_P$. Hence, the dimensionality of the learning problem is drastically reduced since λ is now selecting suitable $\pi_{sel} \in \Pi_{sel}$ based on the pre-processed trajectory information in I_{der} rather than working on the full I_{hist} and Π spaces. Compare with Figures 1 and 3.

1) *Behavior segmentation* where a demonstration $\eta^{(i)}$ is divided into smaller segments, referred to as *task segments*.
2) *Behavior recognition* where each segment is associated with a primitive $\pi_p \in \Pi_P$.
3) Identification of rules or switching conditions for how the primitives are to be combined into a sequence for repetition of the complex behavior. This task is referred to as *behavior coordination*.

Referring to Figure 4, these tasks are realized by the function λ. In practice, task 1 and 2 are often intertwined. For task 1, several approaches exist, for example variance thresholding [30], [48], sub-sequence frequency [53], [52], thresholding mean velocity of joints [43], [22], and entropy-based segmentation [16]. Task 2, is commonly seen as a classification problem. For example, Bentivegna [6] uses a nearest-neighbor classifier on state data to identify skills in a marble maze and air hockey task. In both tasks, each primitive is assigned a query point in state space, which is compared with the current system state. Pook and Ballard [51] present an approach where sliding windows of data are classified using Learning Vector Quantization in combination with a k-NN classifier. The complexity of the distance measure is highly dependent on the complexity of B. For simple behaviors, a Euclidean distance function has been shown to work well [7]. However, for more complex behaviors, other measures are necessary. One of the few approaches that address the complexity of higher level primitives can be found in work by Nicolescu [46], where two behaviors are regarded similar if their respective preconditions and goals match, regardless of their internal differences. We take another approach in [9] where three methods for behavior recognition are evaluated. An observed event sequence η is compared with a known behavior B, using an Auto-associative Neural Network (AANN), a prediction-based method inspired by S-Learning [52], [53] and an action comparison method. In a generalized sense these methods should be seen as an attempt to create a metric in I_{hist}, similar to the notion of a *metric of imitation performance*, e.g. [1], [8].

Approaches to solve task 3 vary widely depending on how the primitives are represented. Nicolescu [46] addresses the problem by assigning pre- and post-conditions to primitives. Several demonstrations represented as sequences of primitives are generalized into a behavior network. The process of computing preconditions for primitives is in this case equivalent to inferring a regular expression from a Finite State Machine. Bentivegna [6] has a drastically different approach. Instead of addressing behavior recognition during learning, suitable primitives are selected on-the-fly by computing sub-goals using Locally Weighted Learning [4], based on the current state. A distributed approach to behavior coordination has been presented by Maes and Brooks [34]. Global feedback is utilized, allowing the primitives themselves to learn suitable activation conditions by correlating certain stimuli with a positive or negative feedback. The feedback functions in combination with the primitives themselves can be seen as a type of bias.

IV. Summary

We have presented a formalism for robot behaviors and *Learning from Demonstration* (LFD). Building on terminology from LaValle [31], an agent´s sensory-motor history is conveniently described by an event history, and a controller maps event histories to actions in action space. As illustrated in Figure 1, a demonstration of a certain behavior can be seen as an event history $\in b$, and the behavior itself as the large set B of allowed event histories, i.e., all possible ways to realize the wanted behavior. The quality of the learned controller can be judged by the similarity between B and the realization space R.

The vague and ill-posed meaning of *repeating* a demonstrated behavior is exemplified and discussed from a machine learning perspective. The concept of *generalization* is defined in the framework of event histories and leads to a discussion of bias in learning. In LFD, bias is essential and can be introduced before, during, and after learning as feedback from the environment or the teacher. The huge information history space may be reduced to a derived space containing only information relevant to specific tasks. Behavior primitives are another common way to introduce bias, and are often associated with specific *goals,* which are explicitly or implicitly defined for each primitive. LFD can be described in this way at a higher level, as controller selection. The learning process then consists of finding and tuning a suitable primitive. More complex behaviors, such as sequences, are commonly learned by segmentation and repeated identification of suitable primitives and switching conditions.

The presented work is an attempt to structure and formalize general principles and assumptions in LFD. Our aim is not to present the single best way to speak about behavior, generalization, goals, and all other concepts related to LFD. Rather, we would like to encourage the effort of defining these concepts at all. Furthermore, it is our hope that the presented work will provide useful insights to the mechanisms involved

in LFD and thus contribute to further development of this powerful and promising area of robot learning.

ACKNOWLEDGMENTS

The authors would like to thank Lars Erik Janlert for many valuable comments on this paper, and Steven LaValle for inspiring discussions about information spaces.

REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with alice: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32:482–496, 2002.

[2] R. Amit and M. Matarić. Parametric primitives for motor representation and control. In *Int. Conf. on Robotics and Automation (ICRA)*, Washington DC, May 2002.

[3] R. C. Arkin. *Behaviour-Based Robotics*. MIT Press, 1998.

[4] C. Atkeson, A. Moore, and S. Schaal. Locally weighted learning. *AI Review*, 11:11–73, April 1997.

[5] P. Bakker and Y. Kuniyoshi. Robot see, robot do: an overview of robot imitation. In *Proceedings of the AISB Workshop on Learning in Robots and Animals*, pages 3–11, Brighton, 1996.

[6] D. C. Bentivegna. *Learning from Observation using Primitives*. PhD thesis, College of Computing, Georgia Institute of Technology, 2004. Director-Christopher G. Atkeson.

[7] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning similar tasks from observation and practice. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[8] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008. In press.

[9] E. A. Billing and T. Hellström. Behavior recognition for segmentation of demonstrated tasks. In *Proceedings of the IEEE International Conference on Distributed Human-Machine Systems*, Athens, Greece, March 2008.

[10] Breazeal and B. Scassellati. *Challanges in Building Robots That Imitate People*. MIT Press, 2002.

[11] C. Breazeal and B. Scassellati. Infant-like social interactions between a robot and a human caretaker. *Adaptive Behavior*, 8(1):49–74, 1998.

[12] R. A. Brooks. New approaches to robotics. *Science*, 253, 13:1227–1232, 1991.

[13] R. W. Byrne and A. E. Russon. Learning by imitation: a hierarchical approach. *The Journal of Behavioral and Brain Sciences*, 16,3, 1998.

[14] S. Calinon and A. Billard. What is the teacher's role in robot programming by demonstration? - Toward benchmarks for improved learning. *Interaction Studies. Special Issue on Psychological Benchmarks in Human-Robot Interaction*, 8(3):441–464, 2007.

[15] S. Calinon, F. Guenter, and A. Billard. On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation*, 37(2):286–298, 2007.

[16] P. Cohen, N. Adams, and Heeringa B. Voting experts: An unsupervised algorithm for segmenting. *To appear in Journal of Intelligent Data Analysis*.

[17] A. Cypher, editor. *Watch What I Do: Programming by Demonstration*. MIT Press, 1993.

[18] T. S. Dahl. *Behavior-Based Learning*. PhD thesis, Faculty of Engineering, University of Bristol, UK, 2002.

[19] N. Delson and H. West. Robot programming by human demonstration: the use of human inconsistency in improving 3d robot trajectories. volume 2, pages 1248–1255, Sep.

[20] J. Demiris and G. Hayes. Do robots ape? In *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents*, pages 28–31, 1997.

[21] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2001.

[22] A. Fod, M. Matarić, and O. Jenkins. Automated derivation of primitives for movement classification. In *Proc. of First IEEE-RAS International Conference on Humanoid Robots*, 2000.

[23] J. G. Greeno. Special issue on situated action. In *Cognitive Science*, volume 17, pages 1 – 147. Ablex Publishing Corporation, Norwood, New Jersey, 1993.

[24] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. *RSJ Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521–1544, 2007.

[25] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, August 2001.

[26] Thomas Hellström. Teaching a robot to behave like a cockroach. In *Proceedings of the Third International Symposium on Imitation in Animals and Artifacts in Hatfield UK*, pages 54–61, 2005.

[27] Thomas Hellström, Thomas Johansson, and Ola Ringdahl. *Development of an Autonomous Forest Machine for Path Tracking*, volume 25 of *Springer Tracts in Advanced Robotics*, pages 603 – 614. Springer, field and service robotics: results of the 5th international conference edition, 2006.

[28] E. Hutchins. *Cognition in the Wild*. MIT Press, Cambridge, Massachusetts, 1995.

[29] L. E. Janlert. *The Robot's Dilemma*, chapter Modeling Change - The Frame Problem, pages 1 – 41. Ablex Publishing, Norwood, New Jersey, 1987.

[30] N. Koenig and M. J. Matarić. Behavior-based segmentation of demonstrated tasks. In *International Conference on Development and Learning (ICDL)*, Bloomington, USA, May 2006.

[31] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[32] H. Lieberman, editor. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, San Francisco, 2001.

[33] L. Ljung. *System Identification*. Prentice-Hall, Simon & Schuster, Englewood Cliffs, New Jersey, 1987.

[34] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *National Conference on Artificial Intelligence (AAAI)*, pages 796–802, 1990.

[35] P. Martin and U. Nehmzow. Programming by teaching: Neural network control in the manchester mobile robot. In *Proc. Intelligent Autonomous Vehicles, Helsinki*. Springer Verlag, jan 1995.

[36] M. J. Matarić. Integration of representation into goal-driven behavior-based robots. In *IEEE Transactions on Robotics and Automation*, volume 8, pages 304–312, 1992.

[37] M. J. Matarić. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2–3):323–336, 1997.

[38] Maja Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, 1995.

[39] Maja J Mataric and Matthew J. Marjanovic. Synthesizing complex behaviors by composing simple primitives. In *Self Organization and Life, From Simple Rules to Global Complexity, European Conference on Artificial Life (ECAL-93)*, pages 698–707, 1993.

[40] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.

[41] T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, New Brunswick, New Jersey, 1980.

[42] Ferdinando A. Mussa-Ivaldi and Simon F. Giszter. Vector field approximation: a computational paradigm for motor control and learning. 67:479–489, 1992.

[43] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi. Recognition and generation of leg primitive motions for dance imitation by a humanoid robot. In *Proceedings of 2nd International Symposium on Adaptive Motion of Animals and Machines, Kyoto, Japan*, 2003.

[44] C. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, *Learning Robots: An Interdisciplinary Approach*, volume 24, pages 136–161. World Scientific Press, 1998.

[45] C. L. Nehaniv and K. Dautenhahn. *Imitation in Animals and Artifacts*, chapter The Correspondence Problem. MIT Press, 2002.

[46] M. Nicolescu. *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*. PhD thesis, University of Southern California, 2003.

[47] J. Peters and S. Schaal. Policy learning for motor skills. In *Proceedings of 14th International Conference on Neural Information Processing (iconip)*, 2007.

[48] R. A. Peters II and C. L. Campbell. Robonaut task learning through teleoperation. In *Proceedings of the 2003 IEEE, International Conference on Robotics and Automation*, pages 23 – 27, Taipei, Taiwan, September 2003.

[49] R. Pfeifer and C. Scheier. Sensory-motor coordination: the metaphor and beyond. pages 23 – 27, 1997.

[50] R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press. Cambrage, Massachusetts, 2001.
[51] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. pages 578–585, 1993.
[52] S. Rohrer, B. Hulet. Becca - a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA, 2006.
[53] S. Rohrer, B. Hulet. A learning and control approach based on the human neuromotor system. In *Procedings of Biomedical Robotics and Biomechatronics, BioRob*, 2006.
[54] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, NJ, 1995.
[55] B. Scassellati. Imitation and mechanisms of joint attention: A developmental structure for building social skills on a humanoid robot. *Lecture Notes in Computer Science*, 1562:176–195, 1999.
[56] F.A. Mussa-Ivaldi S.F. Giszter and E. Bizzi. Convergent force field organized in the frog's spinal cord. *Journal of Neuroscience*, 13(2):467–491, 1993.
[57] H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, 1969.
[58] L. A. Suchman. *Plans and Situated Actions*. PhD thesis, Intelligent Systems Laboratory, Xerox Palo Alto Research Center, USA, 1987.
[59] Wikipedia. Behavior. http://en.wikipedia.org/wiki/Behavior, December 2007.
[60] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. In *Evolutionary Computation, IEEE Transactions on*, volume 1, pages 67 – 82, Apr 1997.
[61] D. Wood, J. Bruner, and G. Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17:89–100, 1976.