Target Languages (vs. Inductive Biases) for Learning to Act and Plan

Hector Geffner
Universitat Pompeu Fabra, Barcelona, Spain
Institució Catalana de Recerca i Estudis Avançats (ICREA), Barcelona, Spain
Linköping University, Linköping, Sweden
hector.geffner@upf.edu

Abstract

Recent breakthroughs in AI have shown the remarkable power of deep learning and deep reinforcement learning. These developments, however, have been tied to specific tasks, and progress in out-of-distribution generalization has been limited. While it is assumed that these limitations can be overcome by incorporating suitable inductive biases, the notion of inductive biases itself is often left vague and does not provide meaningful guidance. In the paper, I articulate a different learning approach where representations do not emerge from biases in a neural architecture but are learned over a given target language with a known semantics. The basic ideas are implicit in mainstream AI where representations have been encoded in languages ranging from fragments of first-order logic to probabilistic structural causal models. The challenge is to learn from data, the representations that have traditionally been crafted by hand. Generalization is then a result of the semantics of the language. The goals of this paper are to make these ideas explicit, to place them in a broader context where the design of the target language is crucial, and to illustrate them in the context of learning to act and plan. For this, after a general discussion, I consider learning representations of actions, general policies, and subgoals ("intrinsic rewards"). In these cases, learning is formulated as a combinatorial problem but nothing prevents the use of deep learning techniques instead. Indeed, learning representations over languages with a known semantics provides an account of *what* is to be learned, while learning representations with neural nets provides a complementary account of *how* representations can be learned. The challenge and the opportunity is to bring the two together.

Introduction

A number of recent breakthroughs have shown the remarkable power of deep learning and deep reinforcement learning (LeCun, Bengio, and Hinton 2015; Bengio, Lecun, and Hinton 2021). These developments, however, have been tied to specific tasks like Chess, Go, or Atari games (Mnih et al. 2015; Silver et al. 2017a,b). Progress in out-of-distribution generalization or in the generation of modular components that can be assembled dynamically for different tasks, has been more limited (Lake et al. 2017; Darwiche 2018; Marcus 2018; Geffner 2018).

While it is assumed that these limitations can be overcome by adding suitable inductive biases in current neural network architectures (Garnelo and Shanahan 2019; Goyal and Bengio 2020), the notion of inductive biases itself is often left vague and does not always provide meaningful guidance. Traditionally, inductive biases refer to biases in the hypothesis space, and in the case of neural networks, to the structure of the parametric function captured by the architecture. More recently, the notion has been grounded on the invariant properties of such functions (Bronstein et al. 2021), but more often they are used to refer to intuitions that are not spelled out in formal detail and are not explicitly evaluated.

In this paper, I aim to articulate a more abstract approach to representation learning where the learned representations are not those that emerge after training a neural network, but those that result over a given *target representation language with a well understood semantics.* The approach is implicit in mainstream, symbolic AI, from McCarthy's observations about the representation of general abstractions (McCarthy 1960), to Pearl's emphasis on the language required for reasoning about causality (Pearl and Mackenzie 2018). The challenge is to learn from data the representations that have traditionally been crafted by hand without having to appeal to background knowledge.

The goals of the paper are to make the ideas behind the language-based approach to representation learning explicit, to place them in a broader context where the design of the target language is critical, and to illustrate them in the setting of learning to act and plan. For this, after a general discussion, I consider the problems of learning actions, general policies, and problem decompositions over suitable domain-independent languages.

Languages, Semantics, Generality

> *It is hard to find a needle in a haystack, but it helps to know what a needle looks like.* J. Pearl[1]

In *Generality in AI*, John McCarthy (1987), one of the founders of the field, quotes an early paper that says that "If one wants a machine to be able to discover an abstraction, it seems most likely that the machine must be able to represent this abstraction in some relatively simple way" (McCarthy

[1] Pearl's quotes are from his twitter feed unless otherwise noted; http://web.cs.ucla.edu/~kaoru/jp-tweets.

1960). From this and the need to use the learned abstractions in a *flexible way*, McCarthy concludes that the representations have to be expressed in a logical language.

Sixty years later, Yoshua Bengio, a leader in the field of deep learning interested in bridging the gap between deep learning and high-level reasoning, addresses similar issues but in slightly different terms:[2] "Systematic generalization is hypothesized to arise from an efficient factorization of knowledge into recomposable pieces corresponding to reusable factors ... This is related yet different in many ways from symbolic AI" (Goyal and Bengio 2020).

Bengio's point that the research agenda that he describes for capturing high-level reasoning is "related yet different" than McCarthy's (symbolic AI) is certainly correct. The claim in this paper, however, is that there is much to be gained by making the two research agendas *complementary*. In other words, symbolic AI has developed families of formal languages for "factorizing knowledge into reusable pieces" with the right semantics for supporting composition and generalization. The limitation of symbolic AI is not in the languages themselves but in their use by humans, which as Bengio says, does not scale. The challenge and the opportunity is to learn the representations over such languages (i.e., symbolic representations) directly from data.

Pearl's quote at the beginning of the section refers to representations for causal reasoning. I aim to illustrate that his point is more general and applies to all representations that must be learned and combined. The "needles" are the representations sought, and we know how they look like when they are representations over known languages. Interestingly, Bengio also finds language relevant for capturing the abstractions that are required for combinatorial generalization, but "language" for him, as for others, is natural language, not a formal language with a semantics.

Example

> *Toy problems is where you learn if you are on the right track. Non-toy problems is when you hide you don't know which track you are on.* J. Pearl

A simple toy problem will help us to make the discussion of structural generalization concrete. Figure 1 shows the Minigrid environment; a benchmark introduced for learning to interpret and achieve goal instructions (Chevalier-Boisvert et al. 2019). As shown in the figure, a goal may be: "pick up the grey box behind you, then go to grey key and open the door". The agent is the red triangle and the limited field of view is displayed in light-grey. The general problem is to learn a *controller* for the agent that accepts *goals* and *observations*, and outputs the *action* to do in each step for reaching the goals.

The Minigrid environment is similar to a classical planning problem (Geffner and Bonet 2013; Ghallab, Nau, and Traverso 2016) except that the action model and the goal language are not given. Both supervised and unsupervised approaches have been tried, and success has been partial (Chevalier-Boisvert et al. 2019; Chevalier-Boisvert, Willems, and Pal 2018): millions of trials are required to achieve the given goals, and even then success rates are not 100%. For improving performance, intuitions leading to alternative architectures and loss functions are introduced (e.g., presence of objects or sparse interactions) which are then evaluated experimentally in relation to baselines. From a methodological point of view, this is not entirely satisfying, and two key questions are *what* is it that we are trying to learn, and how this object can be characterized mathematically, independently of its computation. A step in this direction is to notice that we look for a mapping from *goals* G into general policies π_G for achieving them in a broad range of situations involving any number of objects at any locations, and hence, different state spaces.

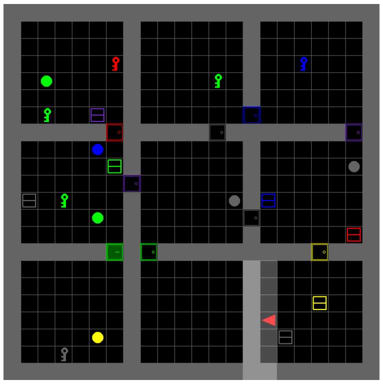

Figure 1: Minigrid environment. Problem is similar to a "classical planning problem" *except* that the domain predicates and action schemas are not known, and goals are to be achieved reactively by general policy, not by planning (Chevalier-Boisvert et al. 2019).

Even this toy example is a hard problem given how little is known a priori. The surprise is not that supervised and unsupervised DL approaches struggle in the problem but that they manage to generate a meaningful behavior at all. The immediate goal for us is not to do better in the task but to identify the building blocks that are needed for approaching this and other problems in a meaningful way. Key questions from the perspective of language-based representation learning are what is a good, domain-independent language for expressing the *general policies* π_G, and how can representation in that language be learned? Related questions are what is a good, domain-independent language for expressing the *dynamics* of Minigrid and how can representations over such a language be learned? The answers to these questions, to be sketched below, do not have to get in the way of developing neural architectures for solving these problems; the hope is that the answers can inform our understanding of the problems and their solutions, neural and otherwise.

Causal Models

> *My greatest challenge was to break away from probabilistic thinking and accept, first, that people are not probability thinkers but cause-effect thinkers and, second, that causal thinking cannot be captured in the*

[2] From the abstract at https://ijcai-21.org/invited-talks.

> *language of probability; it requires a formal language of its own.* J. Pearl[3]

One formal language that is making it into mainstream ML is the language of (structural) causal models (SCMs). This owes to the work of Judea Pearl and others that has revolutionized our understanding of causality by articulating a simple formal language and a semantics to talk about causes and effects (Pearl 2009; Pearl and Mackenzie 2018). The language accommodates observations, interventions, and counterfactuals. A *structural causal model (SCM)* can be understood as a deterministic Bayesian network (conditional probabilities are all one or zero) that defines not just one joint probability distribution over the variables but many. A SCM handles interventions (actions) of the form $do(X = x)$, by which a variable X is set to a specific value x, and the probability distribution that results from such actions is the distribution that is encoded by the "mutilated" Bayesian network where the parents of variable X are replaced by the single parent $do(X = x)$ for which $P(X = x|do(X = x)) = 1$. The answers to queries about combinations of observations, interventions, and counterfactuals are determined by a SCM once the priors on "exogenous" variables (those without parents) are given. The language of SCMs has been used, for example, to determine the conditions under which the answer to queries in one causal model "generalize" or "transport" to another causal model (Pearl and Bareinboim 2011).

In principle, structural causal models can be learned from data, and this is a very active line of research, both in the cases where the variables in the model are given, and when they are not (Schölkopf et al. 2021). This does not mean, however, that structural causal models can be bypassed when answering queries from data. In order to do that meaningfully, the design of the algorithms must take the semantics of SCMs into account (Pearl 2021). Doing this, model-free, while ignoring the language and semantics of SCMs runs the risk of reinventing the wheel with not much guidance, as experimental evaluations are no substitutes for a meaningful theory. This does not rule out the possibility of learning causal models using neural nets by stochastic gradient descent, but the architecture and loss functions must be aligned with the representations sought.

Languages, Models, and Solvers

> *Every science that has thriven, has thriven upon its own symbols.* De Morgan (1864), quoted by J. Pearl.[4]

Bayesian networks and structural causal models are models that make predictions from knowledge expressed in terms of variables, graphs, and probabilities. In AI, other languages and models have been developed as well some of which are relevant to our focus on actions and planning. For example, classical planning refers to planning with deterministic actions with known effects and preconditions, from a known initial state, given a compact encoding of the actions in terms of state variables. These encodings have a size that is polynomial in the number of variables but result in state models of exponential size. Compact languages for representing other state models like MDPs and POMDPs, have also been developed.

The use of languages for encoding state models has been motivated by two reasons. First, state models need to be specified in a concise manner, as they would not fit in memory otherwise. Second, it is assumed that a compact specification reveals structure that can be exploited computationally. For example, a common technique for solving classical planning problems is using heuristic functions, yet these heuristics can be extracted from compact representations but not from flat models (McDermott 1999; Bonet and Geffner 2001).

The benefits of languages supporting compact action representations, however, go well beyond the facilities that they provide for model specification and computation, as they also provide the ingredients needed for *generalization, transfer, and knowledge reuse.* Indeed, these languages have been designed for human to use with these goals in mind: when writing the description of a planning problem, we want the description to be reusable with minor modification in similar problems. The use of *first-order languages* for referring to objects and relations has been essential for this purpose.

Consider for example, a simplification of the Minigrid domain, that we refer to as Delivery, where there are N objects in a grid $n \times m$, and the goal is to pick up the objects, one by one, and deliver them to a target cell. The actions available are to pick up and drop an object, and to move one cell at a time. Different instances of the Delivery domain are encoded in planning languages such as PDDL in two parts as $P = \langle D, I \rangle$. One part, the domain D, encodes what is common about all the Delivery instances in terms of three *action schemas*: *pick*, *drop*, and *move*. The other part, I, details the objects in the instance, the ground atoms that are initially true, and those that must be made true in the goal (McDermott 2000; Haslum et al. 2019). For example, the action schema *pick* can be defined as:

$pick(o, x, y)$:
  Prec: $at(o, x, y), at_r(x, y), handempty$
  Eff: $hold(o), \neg at(o, x, y), \neg handempty$

where o, x, and y are the schema arguments, and preconditions and effects are atoms formed by predicate symbols and some of the schema arguments. The schema for $move$ can be expressed in turn as:

$move(x, y, x', y')$:
  Prec: $at_r(x, y), adjacent(x, y, x', y')$
  Eff: $at_r(x', y'), \neg at_r(x, y)$

where at_r and at are the predicates that encode the location of the agent and the packages, and $adjacent$ encodes the grid topology. These $pick$ and $move$ action schemas, along with the $drop$ scheme and the predicates involved in

[3] http://www.cambridgeblog.org/2012/07/qa-with-judea-pearl-part-one.

[4] (Pearl and Mackenzie 2018).

them, are precisely the "reusable pieces" over all Delivery instances, and hence if we want to learn a dynamic model from some instances that generalizes to other instances, we will be well advised to learn a representation of this type.

There are indeed important similarities between planning languages and structural causal models: SCMs provide a *compact and invariant description* of the effects of interventions on probability distributions, while planning languages provide a *compact and invariant description* of the effects of interventions on states. Compact, first-order languages for defining probabilistic graphical models, MDPs, and POMDPs have also been developed (Raedt et al. 2016; van den Broeck et al. 2021; Younes et al. 2005; Vallati et al. 2015), and if we want to learn *models that generalize*, such languages would be good targets for learning as well.

Languages vs. Inductive Biases

> *It's hard to understand why we should struggle to understand deep learning instead of learning deep understanding*. J. Pearl

There is a compelling reason for why learning approaches are either model-free and learn no models, or are model-based but do not learn language-based representations (i.e., do not learn symbolic representations). The reason is that learning such representations appears to require humans in the loop, something that gets in the way of the automated learning pipeline. This impression, however, is wrong: while *languages* like those of SCMs and planning have been developed to be used by humans, this does not mean that the *representations* over such languages can only be provided by humans and cannot be learned from data. Certainly, there are obstacles to overcome for achieving this and a key one is the identification of the *(state) variables* from unstructured data, but this is a technical problem that can be solved. I'll show that there are indeed crisp solutions to this problem that exploit the natural inductive biases of language-based representations.

Current DRL approaches can learn in principle policies that solve problems such as Delivery or Minigrid for any value of the parameters, but even then, it is not clear why this is so. For representations learned over languages designed to support modularity and reuse, the answers to these questions follow from their semantics.

While the use of formal languages for learning representations is not common in deep learning, there is an increasing trend to reflect intuitions about the representations sought in the architectures. For example, RIM networks assume a dynamics determined by sparse object interactions (Goyal et al. 2020). Yet, informal talk of sparse interactions is no substitute for a language with a clear semantics that can represent the range of possible sparse interactions and lead to representations that can be understood in that way.

As mentioned before, suitable target languages yield meaningful learning biases, as generalization is most often the result of learning compact descriptions. In Bayesian networks, compactness comes from sparse graphs, while in SCMs, compactness comes from the language and semantics of interventions. In planning, compact descriptions result from the language of action schemas and the predicates used in them. Compact descriptions are easier to learn and afford a powerful generalization, implying that representation size is a key bias in language-based representations learning. While there is no similar notion in deep learning, the "right" biases in neural nets would be the ones that deliver compact and reusable representations of this type.

Related Research

Language-based representations, most often first order, are at the heart of the models and solvers studied in AI (Geffner 2014). The problem of learning such representations from data is active in some of these settings, like learning causal representations, mentioned above, and learning general action models, to be discussed below. Methods designed to learn symbolic representations from data provide a natural way for integrating learning and reasoning (Konidaris, Kaelbling, and Lozano-Perez 2018; Evans et al. 2021). Neuro-symbolic methods make use of prior symbolic knowledge (Serafini and Garcez 2016; Yang, Ishay, and Lee 2020), possibly in terms of first-order probabilistic models (Raedt et al. 2016; Manhaeve et al. 2021). The use of logical languages has been central in recent characterizations of the expressive power of graph neural networks (Barceló et al. 2020; Grohe 2020). Domain specific task languages have been used in a number of settings (Lake et al. 2017; Silver et al. 2020; Tsividis et al. 2021), including the dynamics of MDPs (Diuk, Cohen, and Littman 2008). These languages are domain specific in the sense that they assume a particular vocabulary. Language-based representations of rewards have been explored too (Camacho et al. 2019; De Giacomo et al. 2019). Some deep learning approaches aim to approximate first-order formulas like conjunctions of atoms (Shanahan et al. 2020) or rules (Goyal et al. 2021), while others draw intuitions from them (Garnelo and Shanahan 2019; Goyal and Bengio 2020). Generally, symbolic methods like those considered in inductive logic programming (Muggleton and De Raedt 1994) assume and exploit background knowledge, while deep learning approches do not use and do not produce background knowledge; i.e., knowledge that can be reused. This is their advantage and also their limitation.

Action Models, Policies, and Decompositions

We consider next three concrete examples of domain-independent languages for acting and planning, and how representation over them can be learned.

Language for General Action Models

Classical planning problems $P = \langle D, I \rangle$ are described in terms of a planning domain D involving action schemas and predicates, and instance information I detailing the objects, the initial situation, and the goal. A planning instance P defines a unique graph $G(P)$ where the nodes s stand for the states over P and edges (s, s') express that there is a ground action a in P that maps the state s into s'. The states s are the possible truth valuations over the ground atoms in P. The general *model learning problem over this language* can be expressed then as the following inverse problem (Bonet and Geffner 2020):

> Given plain graphs $G_1, \ldots, G_k$, find the simplest domain D and instances $P_i = \langle D, I_i \rangle$ over D such that the given graphs G_i and $G(P_i)$ are isomorphic, $i = 1, \ldots, k$.

Variations of this problem have also been considered where the edges in the input graphs G_i are labeled with action types (e.g., $pick$, $drop$, $move$), and edges may be missing or observations may be noisy (Rodriguez et al. 2021). The domains learned from some instances can then be used to predict the effects of actions in other, unseen instances. This learning formulation has been used to obtain the predicates and action schemas for domains such Blocks, Tower of Hanoi, and others. For example, the following domain description for Hanoi is learned from a single graph, produced by an instance with 3 disks (predicate and variable names are ours):

$Move(d, fr, to)$
Sta: $neq(d, to), neq(d, fr), neq(to, fr), \neg larger(to, d)$
Pre: $\neg p(to, d), \neg p(fr, fr), p(d, d), p(to, to), p(fr, d)$
Eff: $\neg p(to, to), \neg p(fr, d), p(t, do), p(fr, fr)$.

The domain learned can be shown to be correct for instances of any size (any number of disks and pegs), and use the predicate $p(x_1, x_2)$ for two different purposes: for capturing the relation $on(x_2, x_1)$ when $x_2 \neq x_1$, and for capturing clear(x_1) otherwise. The three schema arguments d, fr, to represent disks, and "Sta(tic)" refers to precondition atoms that are not affected by the actions. One can actually test that this domain description D is correct experimentally, using (validation) graphs G obtained from other instances, and checking if there is an I such that G and $G(P)$ for $P = \langle D, I \rangle$ are isomorphic (a simpler version of the learning problem above). The learned representation also identifies the *state variables* of the problem through the $p(d, d')$ atoms that encode the location of each of the disks d. As discussed by Bonet and Geffner (2020), the use of a first-order target language with action schemas is critical for learning such state variables, as propositional representations cannot be reused in the same way and hence do not admit the same learning bias.

The language of action models (action schemas and predicates) is suitable for learning in this setting, not just because it supports representations that generalize to other instances, but also because it defines a *heavily biased hypothesis space*, with the space of possible domains being characterized by a small number of parameters with small integer values, like the number of action schemas and predicates and their arities. Provided with bounds on these values, the learning problem becomes a combinatorial optimization problem that can be solved in a number of ways, in many cases optimally. The optimization criterion used by Rodriguez et al. (2021), for example, minimizes the sum of actions and arities. Asai (2019), on the other hand, learn first-order action representations using deep learning, while Cresswell, McCluskey, and West (2013) learn first-order representations heuristically assuming that action arguments in state transitions are observable. Many other works learn similar representations but assuming that the domain predicates are known.

Language for General Policies

The target languages for learning in many settings can be taken off the shelf like the languages for representing actions and causal models discussed above. But for other tasks, new domain-independent languages with the right properties may have to be created. For example, in the Minigrid problem, DRL approaches are not after general dynamic models, but after general policies: policies that can deal with *any* instance of the domain. What is then a good language for representing such policies that is not tied to this particular domain? This question has been considered in the area of *generalized planning* (Srivastava, Immerman, and Zilberstein 2008; Hu and De Giacomo 2011), and the language below follows the one introduced by Bonet and Geffner (2018).

A general policy π for a class of domain instances $\mathcal{Q}$ is a set of *policy rules* of the form $C \mapsto E$ where C contains boolean conditions of the form p, $\neg p$, $n = 0$, or $n > 0$, and E contains effects of the form p, $\neg p$, $p?$, $n\!\downarrow$, $n\!\uparrow$, $n?$, where p and n stand for boolean and numerical *features*. Features are functions over states. Boolean features p can have value *true* or *false*, and numerical features n can have any non-negative integer value. Conditions in C like p and $n = 0$ are true in a state when p has value true, and n has value 0 respectively, and effects in E like p ($\neg p$), $n\!\downarrow$ ($n\!\uparrow$), and $p?$ ($n?$) indicate that p must be made true (resp. false), that n must decrease (resp. increase), and that p (resp. n) can change in any way. Features not appearing in the effects of a rule must keep their values. The value of all the features Φ in a state s is expressed as $f(s)$, and f without a state argument refers to an arbitrary feature valuation.

A *pair of feature valuations* (f, f') satisfies a policy rule $C \mapsto E$ if f makes the conditions in C true, and the change in feature values from f to f' is compatible with E. A *state transition* (s, s') in P is compatible with a policy π if $(f(s), f(s'))$ satisfies a policy rule, and a *state trajectory* $s_0, \ldots, s_n$ is compatible with the policy in P if s_0 is the initial state of P and each transition (s_i, s_{i+1}) is compatible with π. Finally, the policy π *solves* P if every maximal state trajectory compatible with π reaches a goal state of P, and it solves $\mathcal{Q}$ if it solves every instance P in $\mathcal{Q}$.

For example, the policy π over the features $\Phi = \{H, n\}$, captured by the following two rules

$$\begin{array}{ll} \{\neg H, n > 0\} \mapsto \{H, n\!\downarrow\}; & \text{pick block above } x \\ \{H, n > 0\} \mapsto \{\neg H\}; & \text{put block away} \end{array}$$

where H is true if holding a block and n is the number of blocks above a block x, solves the class $\mathcal{Q}$ of Blocksworld problems where the goal is $clear(x)$, regardless of the number or initial configuration of blocks. The first rule in the policy says to do any action that makes H true and decreases the value of n, provided that H is false and n is positive, while the second rule says to do any action that makes H false and does not affect the value of n, provided that H is true and n is positive.

More interestingly, a general policy for the Delivery domain above can be defined using the features $\Phi = \{H, p, t, n\}$ for "holding", "distances to nearest package and target", and "number of undelivered packages", as:

$$\begin{array}{ll}
\{\neg H, p > 0\} \mapsto \{p\downarrow, t?\}; & \text{go to nearest pkg} \\
\{\neg H, p = 0\} \mapsto \{H\}; & \text{pick it up} \\
\{H, t > 0\} \mapsto \{t\downarrow\}; & \text{go to target} \\
\{H, n > 0, t = 0\} \mapsto \{\neg H, n\downarrow, p?\}; & \text{drop pkg.}
\end{array}$$

The first rule says to do any action that decreases the distance p to the nearest package when not holding a package and the distance is positive, whatever the effect on the distance t to the target. The reading of the other rules is similar. These are policies written by hand though, and the question is how such policies can be learned? As before, this learning problem has been formulated and solved as a combinatorial optimization problem by creating a large but finite set $\mathcal{F}$ of possible boolean and numerical features *from the domain predicates*, using a standard grammar based on description logics, which is a fragment of 2-variable logics (Baader, Horrocks, and Sattler 2008). Provided with this set $\mathcal{F}$ where each feature is given a cost (the number of grammar rules used to derive it), the learning task becomes:

Given a domain D, instances $P_1, \ldots, P_k$ of $\mathcal{Q}$, and a finite pool of features $\mathcal{F}$, each with a cost, find the cheapest set of features $\Phi \subset \mathcal{F}$ and a policy π over them such that π solves the instances $P_1, \ldots,$ and P_k.

This is a combinatorial optimization problem that is cast and solved as a Weighted Max-SAT task (Francès, Bonet, and Geffner 2021). Once again, the language in which representations are sought provides a *strongly biased hypothesis space* where policies that involve few simple features (in terms of the domain predicates) are preferred. As before, nothing precludes the use of deep learning to provide an alternative computational method, potentially more scalable and robust (Toyer et al. 2020; Garg, Bajpai, and Mausam 2020). A formal step in this direction is the computation of general optimal value functions using graph neural networks (Ståhlberg, Bonet, and Geffner 2021), that exploits a correspondence between 2-variable logics and GNNs (Barceló et al. 2020; Grohe 2020).

Language for Decomposing Goals into Subgoals

The problem of *expressing* and *using* the common subgoal structure of a collection of problems $\mathcal{Q}$ has been important in AI since the 1960s (Newell and Simon 1963; Erol, Hendler, and Nau 1994), while the problem of *learning* such structure (in the form of intrinsic rewards) has become important in recent RL research as well (Zheng et al. 2020). We are interested in a similar problem but want to learn such structure over a suitable formal language.

A *policy sketch* or simply a *sketch* is a set of sketch rules $C \mapsto E$ of the same form as policy rules. But while policy rules filter 1-step transitions; namely, when in a state s, select a 1-step transition to any s' such that the feature valuations $(f(s), f(s'))$ satisfy a policy rule, sketch rules define *subproblems:* when in a state s, reach a state s', *not necessarily in one step*, such that the feature valuations $(f(s), f(s'))$ satisfy a sketch rule (Bonet and Geffner 2021).

Sketches *decompose* problems into *subproblems* without prescribing how these subproblems should be solved (going from s to s'). One is interested, however, in sketches that yield subproblems that can be solved efficiently, in low polynomial time (in the number of problem variables), and this is guaranteed when subproblems have a low, bounded width (Lipovetzky and Geffner 2012). In that case, the sketch has a bounded width, and all the problems in $\mathcal{Q}$ can be solved in polynomial time.

For example, a sketch R_1 for Delivery that involves the single feature n which tracks the number of packages not yet delivered, is given by the rule

$$R_1 : \{\{n > 0\} \mapsto \{n\downarrow\}\}$$

that expresses a decomposition where, in states s where $n > 0$, states s' should be reached where the value of n is lower than in s. One can show that the resulting subproblems have a width bounded by 2. Likewise, a sketch R_2 over the features n and H with the same meaning as above, can be given with two rules:

$$R_2 : \{\{\neg H\} \mapsto \{H\}\,,\, \{n > 0, H\} \mapsto \{n\downarrow, \neg H\}\}\ .$$

The rule on the left says that if not holding a package, get hold of one, while the other rule, that if holding a package, deliver it. The sketch R_2 has width 1 meaning that all subproblems and hence all Delivery instances are rendered solvable in linear time. The use of hand-crafted sketches has been addressed by Drexler, Seipp, and Geffner, and work on learning sketches automatically is next.

Summary

Deep learning and deep reinforcement learning are incredibly powerful techniques that struggle with structural generalization. While researchers assume that the right inductive bias in the architecture is all that is needed, no much guidance is offered to get there. In this paper, I've argued that learning representations over suitably designed formal languages with a semantics provides a research path that is crisp and meaningful, and illustrated the approach by focusing on the problems of learning general action models, policies, and subgoals (intrinsic rewards). This is all compatible with Bengio's vision that systematic generalization arises from an "efficient factorization of knowledge into recomposable pieces", but complements it by assuming that the pieces are expressed in a language. Learning language-based representations from data, indeed, is not incompatible with the use of deep learning techniques. Moreover, the integration of language-based representations and deep learning, one describing what needs to be learned, and the other, delivering it at scale, has the potential to inform the design of deep learning methods that are more transparent and which can be assessed in ways that go beyond performance curves.

Acknowledgments

I thank Judea Pearl, and not just for the quotations. I also thank Blai Bonet for comments and collaboration, the RLeap team, and the reviewers. The research is partially supported by an ERC Advanced Grant (No 885107), the EU Horizon 2020 project TAILOR (Grant No 952215), and the Knut and Alice Wallenberg (KAW) Foundation, under the WASP Program.

References

Asai, M. 2019. Unsupervised Grounding of Plannable First-Order Logic Representation from Images. In *Proc. ICAPS*.

Baader, F.; Horrocks, I.; and Sattler, U. 2008. *Handbook of Knowledge Representation*, chapter Description Logics. Elsevier.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J.; and Silva, J. P. 2020. The logical expressiveness of graph neural networks. In *ICLR*.

Bengio, Y.; Lecun, Y.; and Hinton, G. 2021. Deep learning for AI. *Communications of the ACM*, 64(7): 58–65.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1–2): 5–33.

Bonet, B.; and Geffner, H. 2018. Features, Projections, and Representation Change for Generalized Planning. In *Proc. IJCAI*, 4667–4673.

Bonet, B.; and Geffner, H. 2020. Learning first-order symbolic representations for planning from the structure of the state space. In *Proc. ECAI*.

Bonet, B.; and Geffner, H. 2021. General Policies, Representations, and Planning Width. In *Proc. AAAI*.

Bronstein, M. M.; Bruna, J.; Cohen, T.; and Veličković, P. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*.

Camacho, A.; Icarte, R. T.; Klassen, T. Q.; Valenzano, R. A.; and McIlraith, S. A. 2019. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In *IJCAI*, 6065–6073.

Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T. H.; and Bengio, Y. 2019. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning. In *ICLR*.

Chevalier-Boisvert, M.; Willems, L.; and Pal, S. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

Cresswell, S. N.; McCluskey, T. L.; and West, M. M. 2013. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review*, 28(2): 195–213.

Darwiche, A. 2018. Human-level intelligence or animal-like abilities? *Communications of the ACM*, 61(10): 56–67.

De Giacomo, G.; Iocchi, L.; Favorito, M.; and Patrizi, F. 2019. Foundations for restraining bolts: Reinforcement learning with LTLf/LDLf restraining specifications. In *ICAPS*, volume 29, 128–136.

Diuk, C.; Cohen, A.; and Littman, M. L. 2008. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 240–247.

Drexler, D.; Seipp, J.; and Geffner, H. 2021. Expressing and Exploiting the Common Subgoal Structure of Classical Planning Domains Using Sketches. In *KR*. ArXiv preprint arXiv:2105.04250.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *AAAI*, volume 94, 1123–1128.

Evans, R.; Bošnjak, M.; Buesing, L.; Ellis, K.; Pfau, D.; Kohli, P.; and Sergot, M. 2021. Making sense of raw input. *Artificial Intelligence*, 299.

Francès, G.; Bonet, B.; and Geffner, H. 2021. Learning General Planning Policies from Small Examples Without Supervision. In *Proc. AAAI*, 11801–11808.

Garg, S.; Bajpai, A.; and Mausam. 2020. Generalized Neural Policies for Relational MDPs. In *Proc. ICML*.

Garnelo, M.; and Shanahan, M. 2019. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29: 17–23.

Geffner, H. 2014. Artificial Intelligence: From programs to solvers. *AI Communications*, 27(1): 45–51.

Geffner, H. 2018. Model-free, model-based, and general intelligence. In *IJCAI*, 10–17.

Geffner, H.; and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers.

Ghallab, M.; Nau, D.; and Traverso, P. 2016. *Automated planning and acting*. Cambridge U.P.

Goyal, A.; and Bengio, Y. 2020. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*.

Goyal, A.; Didolkar, A.; Ke, N. R.; Blundell, C.; Beaudoin, P.; Heess, N.; Mozer, M.; and Bengio, Y. 2021. Neural Production Systems. *arXiv preprint arXiv:2103.01937*.

Goyal, A.; Lamb, A.; Hoffmann, J.; Sodhani, S.; Levine, S.; Bengio, Y.; and Schölkopf, B. 2020. Recurrent Independent Mechanisms. In *ICLR*.

Grohe, M. 2020. The Logic of Graph Neural Networks. In *Proc. of the 35th ACM-IEEE Symp. on Logic in Computer Science*.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool.

Hu, Y.; and De Giacomo, G. 2011. Generalized planning: Synthesizing plans that work for multiple environments. In *Proc. IJCAI*, 918–923.

Konidaris, G.; Kaelbling, L. P.; and Lozano-Perez, T. 2018. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61: 215–289.

Lake, B.; Ullman, T.; Tenenbaum, J.; and Gershman, S. 2017. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521(7553): 436.

Lipovetzky, N.; and Geffner, H. 2012. Width and serialization of classical planning problems. In *Proc. ECAI*, 540–545.

Manhaeve, R.; Dumančić, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2021. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298: 103504.

Marcus, G. 2018. Deep Learning: A Critical Appraisal. *arXiv preprint arXiv:1801.00631*.

McCarthy, J. 1960. Programs with common sense. In *Proc. Teddington Conf. on the Mechanization of Thought Processes*. Reprinted in M. Minsky (Ed.), Semanfic Informafion Processing, 1968, MIT Press, Cambridge, Mass.

McCarthy, J. 1987. Generality in artificial intelligence. *Communications of the ACM*, 30(12): 1030–1035.

McDermott, D. 1999. Using regression-match graphs to control search in planning. *Artificial Intelligence*, 109(1-2): 111–159.

McDermott, D. 2000. The 1998 AI Planning Systems Competition. *Artificial Intelligence Magazine*, 21(2): 35–56.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529.

Muggleton, S.; and De Raedt, L. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19: 629–679.

Newell, A.; and Simon, H. 1963. GPS: a program that simulates human thought. In Feigenbaum, E.; and Feldman, J., eds., *Computers and Thought*, 279–293. McGraw Hill.

Pearl, J. 2009. *Causality: Models, Reasoning, and Inference (2nd Edition)*. Cambridge University Press.

Pearl, J. 2021. Radical empiricism and machine learning research. *Journal of Causal Inference*, 9(1): 78–82.

Pearl, J.; and Bareinboim, E. 2011. Transportability of causal and statistical relations: A formal approach. In *AAAI*.

Pearl, J.; and Mackenzie, D. 2018. *The book of why: the new science of cause and effect*. Basic books.

Raedt, L. D.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. *Statistical relational artificial intelligence: Logic, probability, and computation*. Morgan & Claypool Publishers.

Rodriguez, I. D.; Bonet, B.; Romero, J.; and Geffner, H. 2021. Learning First-Order Representations for Planning from Black-Box States: New Results. In *KR*. ArXiv preprint arXiv:2105.10830.

Schölkopf, B.; Locatello, F.; Bauer, S.; Ke, N. R.; Kalchbrenner, N.; Goyal, A.; and Bengio, Y. 2021. Toward causal representation learning. *Proceedings of the IEEE*, 109(5): 612–634.

Serafini, L.; and Garcez, A. S. d. 2016. Learning and reasoning with logic tensor networks. In *Conference of the Italian Association for Artificial Intelligence*, 334–348. Springer.

Shanahan, M.; Nikiforou, K.; Creswell, A.; Kaplanis, C.; Barrett, D.; and Garnelo, M. 2020. An explicitly relational neural network architecture. In *International Conference on Machine Learning*, 8593–8603.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017a. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017b. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354.

Silver, T.; Allen, K. R.; Lew, A. K.; Kaelbling, L. P.; and Tenenbaum, J. 2020. Few-Shot Bayesian Imitation Learning with Logical Program Policies. In *Proc. AAAI*, 10251–10258.

Srivastava, S.; Immerman, N.; and Zilberstein, S. 2008. Learning generalized plans using abstract counting. In *Proc. AAAI*, 991–997.

Ståhlberg, S.; Bonet, B.; and Geffner, H. 2021. Learning General Optimal Policies with Graph Neural Networks: Expressive Power, Transparency, and Limits. *arXiv preprint arXiv:2109.10129*.

Toyer, S.; Thiébaux, S.; Trevizan, F.; and Xie, L. 2020. ASNets: Deep Learning for Generalised Planning. *Journal of Artificial Intelligence Research*, 68: 1–68.

Tsividis, P. A.; Loula, J.; Burga, J.; Foss, N.; Campero, A.; Pouncy, T.; Gershman, S. J.; and Tenenbaum, J. B. 2021. Human-level reinforcement learning through theory-based modeling, exploration, and planning. *arXiv preprint arXiv:2107.12544*.

Vallati, M.; Chrpa, L.; Grześ, M.; McCluskey, T. L.; Roberts, M.; Sanner, S.; et al. 2015. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3): 90–98.

van den Broeck, G.; Kersting, K.; Natarajan, S.; and Poole, D. 2021. *Introduction to Lifted Probabilistic Inference*. MIT Press.

Yang, Z.; Ishay, A.; and Lee, J. 2020. NeurASP: Embracing Neural Networks into Answer Set Programming. In *IJCAI*, 1755–1762.

Younes, H. L.; Littman, M. L.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24: 851–887.

Zheng, Z.; Oh, J.; Hessel, M.; Xu, Z.; Kroiss, M.; Van Hasselt, H.; Silver, D.; and Singh, S. 2020. What can learned intrinsic rewards capture? In *ICML*, 11436–11446.