Neuro-Symbolic Artificial Intelligence: The State of the Art

Pascal Hitzler[a] and Md Kamruzzaman Sarker[b]
[a] *Kansas State University*
[b] *University of Hartford*

Contents

Chapter 1

Logic meets Learning: From Aristotle to Neural Networks

Vaishak Belle, University of Edinburgh, UK

The tension between deduction and induction is perhaps the most fundamental issue in areas such as philosophy, cognition and artificial intelligence. In this chapter, we survey work that provides evidence for the long-standing and deep connections between logic and learning. After a brief historical prelude, our narrative is then structured in terms of three strands of interaction: logic versus learning, machine learning for logic, and logic for machine learning, but with ample overlap.

1.1. Introduction

Deduction is the process of drawing conclusions from acquired or specified knowledge. So we focus on questions about the expressiveness of formal languages for representing knowledge, together with proof systems for reasoning from that knowledge. Induction is the process of extracting knowledge from observations. So we focus on the semantics of generalization from partial descriptions, together with the target language for representing and the algorithms for computing that generalization. Consequently, frameworks for integrating the two mechanisms in service of artificial or natural cognition is of fundamental theoretical and practical import. Conceptually, the matter is clearly relevant to areas such as philosophy, cognitive science and artificial intelligence, but on a practical level, it impacts science broadly as it attempts to balance expert knowledge with empirical data.

Somewhat surprisingly, historically, approaches to reasoning and learning in AI have attempted to make progress on the topic of cognition almost independently of each other. However, advances in cross-over areas such as *statistical relational learning* [1, 2], *high-level control* [3, 4] and most recently, *neuro-symbolic systems* [5, 6, 7, 8] have illustrated that the dichotomy is not very constructive, and perhaps even ill-formed. Such areas correctly argue that logic emphasizes high-level reasoning, and encourages structuring the world in terms of objects, properties, and relations. In contrast, much of the inductive machinery assume random variables to be independent and identically distributed, which can be problematic when attempting to reason about (physical and social) relationships between groups of objects.

The need to integrate logic and learning can also be motivated without resorting to the apparent flexibility that logical syntax offers for modeling relations and hierarchies. To tackle artificial cognition, it seems we might need to consider approaches that symbolic logic and machine learning struggle to duplicate in functionality individually. For example, although there is much debate about what precisely commonsense knowledge might look like, it is widely acknowledged that concepts such as time, space, abstraction and causality are essential [9, 10]. Low-level sensory data can be processed by machine learning, but beyond that, (classical, or perhaps non-classical) logic can provide the formal machinery to reason about such concepts in a rigorous way. Indeed, it is worth noting that attempts to logically formalize generalizations of sensory data (vision, audio, and so on) have not been successful – they have either proven to be too coarse or too brittle to deal with variations seen in the real-world [11]. So a logical theory for unifying top-down and bottom-up processing also seems infeasible.

In more immediate terms, despite the success of methods such as deep learning, it is now increasingly recognized that owing to a number of reasons, including model re-use, transferability, explainability and data efficiency, those methods need to be further augmented with logical, symbolic and/or programmatic artifacts [12, 13, 14]. Likewise, for building intelligent agents, low-level, data-intensive, reactive computations needs to be tightly integrated with high-level, deliberative computations [3, 4, 15], the latter possibly also engaging in hypothetical and counterfactual reasoning. Here, a parallel is often drawn to Kahneman's so-called *System 1* versus *System 2* processing in human cognition [16], in the sense that experiential and reactive processing (learned behavior) needs to be coupled with cogitative processing (reasoning, deliberation and introspection) for sophisticated machine intelligence. (Of course, this is not to suggest that developments in one equates to insights in the other, or even that abstract models can easily imbibe cultural and sociopolitical norms in a straightforward manner.)

The purpose of this chapter is not to resolve this debate, but rather briefly survey work and distill representative ideas that provide evidence for the deep connections between logic and learning. We begin with historical developments before turning to the ways we will distill those ideas.

1.2. Lineage

Our goal here is to trace a very rough lineage of ideas. It would be impractical to discuss the intricate philosophical and mathematical issues that arise in logic, learning and their integration. The historical developments of inductive reasoning alone is significant and complex, surely worthy of a book in itself. Likewise, there are various book-length treatments on subfields such as statistical relational learning [2, 1], and of course, the current book on neuro-symbolic learning.

So we are organized as follows. We begin with the type of formal reasoning that distinguishes this enterprise from conventional machine learning: the presence of logical artifacts, their use for postulating hypotheses by arranging symbols, and deriving conclusions in a truth-preserving manner. Formal logic, in the vast majority of applications, is concerned with *deduction* after all. This then provides the background for introducing *induction*, some of the modern flavors of which are the focus of the rest of the chapter. We also briefly position *abduction*, a reasoning approach for most likely explanations.

We remark that this article is undoubtedly a biased view, as the body of related work is large, touching on multiple sub-areas of AI, computer science, philosophy and cognitive science [17]. Readers are encouraged to refer to discussions in our references, among others, to get a sense of the breadth of the area.

1.2.1. Deduction

Deductive reasoning goes back at least to 4th century BC. In a collection of works, Aristotle suggested the use of syllogisms as a means to arrive at *new* and *necessary* truths. New, because the conclusion is not present in the premise, and necessary, because the conclusions are inescapable – provided the rules of inference are irrefutable. As a mechanism for reasoning about discourse and arguments, it is clearly attractive to carefully work out what to infer from a set of premises and it ultimately influenced philosophy considerably [18, 19]. Leibniz, in particular, felt that just as the rules of arithmetic manipulate abstract numbers symbolically, the rules of logic would manipulate abstract ideas symbolically too [20]. Thus, logical reasoning had long been identified with rational thought [21, 22].

It was only much later, in the 19th century primarily, that efforts to algebraize and equationalize logical reasoning by Boole, Frege and others led to formal systems like the ones in use today in computer science and AI. Nowadays, we associate deductive reasoning with at least the following three rules of inference [23, 24, 25]:

- **Modus Ponens.** Suppose p gives q. Assume p. Therefore q.
- **Modus Tollens.** Suppose p gives q. Assume $\neg q$. Therefore $\neg p$.
- **Syllogisms.** Suppose p gives q. Suppose q gives r. Assume p. Therefore r.

(Syllogisms could also take the form of Modus Ponens.) These could be further combined with quantifiers, which allow rules such as:

- **Universal instantiations.** Assume $\forall x\, P(x)$. Therefore $P(t)$ for any term t. Terms are constants, or variables, or functions applied to constants and variables.

It is the application of universal instantiations with syllogisms that allows Aristotelean arguments such as: all humans are mortal; Socrates is human; therefore, Socrates is mortal.

Given a set of sentences, or theory, T in a logic L, we are interested in the following question: assume T; is $\alpha \in L$ a logical consequence of T by way of deduction? We could arrive at α either by substituting sentences from T in the rules of inference – i.e., proof-theoretically, or by trying out truth value substitutions to the atoms in T – i.e., semantically. In the latter case, every substitution where T evaluates to true must also be one where α evaluates to true. The celebrated Gödel completeness theorem established the correspondence between these two views: for any T, α in first-order logic, if α is a semantic consequence of T, written $T \models \alpha$, then there is a proof of α from T, written $T \vdash \alpha$.

In the 1960s, *resolution*, which compacts syllogistic reasoning, was established as a single rule of inference that is complete in the above sense for first-order logic. Together with its straightforward application in Horn clausal logic, the development of logic programming and Prolog burgeoned [26]. While all of this is proof-theoretic machinery, semantic solvers have also matured. SAT (Satisfiability) and SMT (Satisfiability Modulo Theory) solvers attempt to find satisfying assignments to propositional and fragments of first-order logic [27]. Like resolution, semantic approaches determine $T \models \alpha$ by refutation: $T \models \alpha$ if and only if $T \wedge \neg\alpha$ has no satisfying assignment. Early SAT solvers used backtracking to search the space of all possible truth value substitutions; modern solvers use a range of advanced techniques from local search (sound but not complete), conflict analysis and parallelization [27].

It is interesting to note that Gödel's other ground-breaking result, the incompleteness theorem, might have led Turing to believe that intelligent machines need a mechanism for induction [28]. The incompleteness theorem states that any first-order logical theory T, which includes Peano's axioms for natural numbers, is either inconsistent or incomplete.

We now briefly review induction, i.e., learning by generalizing from observations. Note that this is not to be confused with mathematical induction, which is used to prove properties for an infinite sequence of objects, and is a form of deduction.

1.2.2. Induction

Not surprisingly, inductive reasoning has been a core issue for the logical worldview, as we need a mechanism for obtaining axiomatic knowledge. Here too, discussions date back to the early Greek philosophers: Aristotle suggested that we need the means for generalizing an argument *from the particular to the universal*. That is, deduction produces *new* knowledge in the form of conclusions not explicitly stored in the knowledge base as beliefs, and induction produces *new* knowledge via statements evidenced only in individual instances. But this knowledge is not *necessary* in the sense of being irrefutable. If a counterexample were provided, the induced statement would have to be amended or in the worst case retracted. The often used example here is that of believing that swans are white until one encounters the less common black swan. It is this fragility of induced beliefs that Hume alluded to in his criticisms of inductive reasoning [29].

However, the usefulness of learning from observations as a mechanism for producing knowledge has led to numerous accounts of induction. For example, through the development of the mathematical area of Statistics, assessing hypotheses has a rigorous regime in many scientific fields. But prominent philosophers, such as De Finetti and Carnap, were also interested in formal accounts of induction [30, 31, 32, 33]. Carnap in particular considered how such statistical notions of evaluating hypotheses could be formulated in first-order logic. Indeed, rather than seeking a generalization (or hypothesis) H that explains all the observations O, we might construct a hypothesis H' such that $\Pr(O \mid H')$ is maximized, where the probability function Pr captures the degree of support for H'.

Since the birth of the field of AI, Plotkin [34], Michalski [35], Shapiro [36], Muggleton [28], De Raedt [37] and others established practical inductive reasoning systems in first-order logic based on generalizing clauses that entail the examples, but variant accounts were developed too, such as the information-theoretic approach of Quinlan [38]. These could be contrasted with much of early (but also many modern) machine learning frameworks (or leaners) that are propositional. They also lack a general solution for the provision of axiomatic background knowledge.

1.2.3. Abduction

Although we solely focus on the interaction between deduction and induction here, one other type of logical reasoning is very relevant in our context. Developed by Charles Sanders Pierce [39], *abduction* allows us to infer a hypothesis as a plausible explanation for an observation. The "deductive" capabilities that the fictional detective Sherlock Holmes was most famous for are, in fact, instances of abduction [40]. From a logical perspective, abductive hypotheses might appear to have a similar semantical requirement as inductive hypotheses, in the sense of constructing a sentence such that together with the background knowledge, the examples are entailed. However, while in induction we seek a clause that generalizes examples, in abduction, given a clause that makes a general claim and observations that agree with the conclusion of that claim, we offer the premise of the clause as a plausible explanation. For example, if all swans from Australia are black, and we observe a swan that is (unusually) black as opposed to the white swans we usually encounter, we conclude that the swan must be from Australia.

We will not dwell on abduction anymore, but note that, for example, in [41, 42], a case is made for neuro-symbolic integration as an abductive process. Even earlier to this, Poole [43, 44] established a semantic foundation for a probabilistic logic programming language called ICL (independent choice logic) via abduction. ICL is a precursor of many probabilistic logical modeling languages seen in statistical relational learning [1].

1.2.4. A Modern Desiderata

In the recent years, there has been tremendous development on the interface between learning and logic, and symbolic structures more generally [45]. The impetus for these developments stems from the fact that symbolic logic served as the calculus for modeling information in artificial intelligence and computer science for several decades [46, 47]. But the sheer intricacy and noise in the real-world coupled with the incompleteness of specifications and requirements at the start of any problem solving endeavor has motivated modelers to look to data for completeness and robustification.

This is not without its challenges, however. A great deal of effort is needed to adapt conventional learners based on labelled-data to the involved syntactic structures widely used in computer science and artificial intelligence, including exotic logics and programs. In fact, just the extension of learning schemes from atomic random variables to clausal logic and probabilistic extensions of clausal logic has led to the subfields of inductive logic programming and statistical relational learning. These subfields continue to enjoy active development [48, 49].

But leaving such syntactic matters aside, there are a number of fundamental questions that probe at the heart of the logic-learning enterprise, such as:

- *What knowledge does a system need to have in advance – i.e., provided by the modeler – versus what can be acquired by observations?*
  That is, are we only learning the probabilities of a given knowledge base, or are we also learning the knowledge base itself? Do we need to assume the provision of some background knowledge before the learning process is initiated (i.e., the question of what is innate to the agent before it can effectively learn from its environment)? Is this determined on a case-by-case basis, or can we arrive at broader conditions for this delineation?

- *What is the language for representing and reasoning with the background knowledge and observations?*
  That is, is it propositional, first-order, modal, or some probabilistic extension thereof? Do we assume the provision of labeled training data denoting the observations, or are we considering systems that purposefully act in an environment, and then receive observations and rewards as a result of those actions?
- *What kind of semantics governs the updating of a priori knowledge given new and possibly conflicting observations?*
  That is, do we assume all observations are provided before the hypothesis is generated? What is the appropriate mechanism for learning the rules and repairing them as more, and possibly, conflicting observations are made? Do we have a means to quantify the generalization capabilities of the learner with respect to unknown ground truth?
- *What are the rules of inference that apply to the resulting knowledge base?*
  That is, is the query language unrestricted wrt logical connectives? Do we permit proofs of arbitrary depth? What is the mechanism for requesting "shallow" versus "deep" reasoning (e.g., quickly reacting to a tiger versus reflecting on the nature of the universe)?
- *How does the system generalize from low-level observations to high-level structured knowledge?*
  That is, do we assume the vocabulary of the high-level language is available a priori, or do we need to discover a (possibly hierarchical) mapping between high-level predicates and low-level atomic observations? Is there a means to generate examples to support or refute a claim made using high-level predicates?

To a large extent, much of the recent work on logic and learning, including deep learning, are attempting to address such questions by formulating methodological or algorithmic solutions. It would be impossible to report on every such development, of course, so in this chapter we mainly look at some representative ideas. Our narrative is structured in terms of three strands, inspired by [50]:

1. *Logic versus Machine Learning:* the study of problems that can be solved using either logical techniques or via machine learning;
2. *Machine Learning for Logic:* the learning of logical formulas and logic programs, but also the use of machine learning to guide search; and
3. *Logic for Machine Learning:* the use of logic to enhance machine learning, identify the boundary between tractable and intractable learning problems, and as a declarative language for expressing machine learning problems.

At the outset, we note that the focus is on logical languages, in the sense of a formal framework embodying inference rules in the manner discussed earlier, and not simply the induction of symbols lacking a formal semantics. In a way, symbolic induction is much larger and more ambiguous problem space: after all, even conventional neural network outputs can be interpreted symbolically.

Through the course of our discussion, we also focus on the following sore point: there is a common misconception that logic is for discrete properties, whereas probability theory and machine learning, more generally, is for continuous properties. It is true that logical formulas are discrete structures, but they can very easily also express properties about countably infinite or even uncountably many objects. Consequently, in this article we survey some recent results that tackle the integration of logic and learning in infinite domains.

It is worth remarking that the importance of *logic programming* to the thrust of this article *cannot be overstated*. Indeed, it is logic programming that set the stage for numerous deduction-based programming languages, from declarative modeling [26, 51] to reasoning about choices and probabilities [43, 1] to reasoning about actions and plans [52]. Likewise, inductive logic programming set the stage for major practical advances on learning theories and programs [28, 49]. Probabilistic extensions to logic programming and inductive logic programming [1], which we also touch upon in the below subsection, continue to invite new integrations of relational and graph-based modeling with learning. Many neural learning schemes to integrate logical reasoning do so via logic programming [14]. Thus, although various other languages are being developed too, logic programming provides a convenient and fairly accessible apparatus to integrate reasoning and learning, and study questions of the sort discussed earlier.

1.2.5. Logic & Probability

Before turning to the three strands, let us briefly reflect on a very related enterprise, that of bridging logic and probability. In so much as one would view induction as a probabilistic process, our discussions above already suggest deep connections between logic and probability. But there is much more to be said about the role of probabilities for logic, and vice versa.

Philosophers from the 19th century, including John Stuart Mill and George Boole, felt (human) reasoning would involve both logic and probability. In the earliest days of AI, McCarthy [53] put forward a profound idea: he posited that what the system needs to know could be represented in a *formal language*, and a general-purpose algorithm would then conclude the necessary actions needed to solve the problem at hand. The main advantage is that the representation can be scrutinized and understood by external observers, and the system's behavior could be improved by making statements to it. All of this was ultimately grounded in logic, of course. But pervasive uncertainty in almost every domain of interest led to alternative formalisms for representing knowledge. Uncertainty could range from measurement errors (e.g. readings from a thermometer) to the absence of categorical assertions (e.g. smoking may be a factor for cancer, but cancer is not an absolute consequence for smokers) to "latent" factors that the modeler may simply not have taken into account, all of which question the legitimacy of the expert knowledge. Moreover, classical logic is seen as being "rigid" (sentences always evaluate to true or false) and "brittle" (sentences in the knowledge base must be true in all possible worlds). Although this in turn led to probabilistic formalisms such as Bayesian and Markov networks, as argued the relational structure of the world has meant we have now come a full circle: probabilistic logics are logical languages that admit probabilistic assertions [54, 55, 56]. Semantically, in classical logic, $T \models \alpha$ implies that every truth assignment where T is true is also one where α is true. In probabilistic logics, logical and probabilistic assertions in T mean that there is a measure on the the set of truth assignments, and we are often interested in the probability of α being true.

Currently, the most popular flavors of such probabilistic logics are syntactic extensions to probabilistic formalisms, as seen in statistical relational learning [1]. That is, the use of logical syntax allows one to define an involved probabilistic model, but it can often be interpreted at a propositional level with probabilities over propositional interpretations. For example, the weighted formula [1]:

$$0.9 \quad \forall x, y.\ (\mathit{Smokers}(x) \wedge \mathit{Friends}(x, y) \supset \mathit{Smokers}(y))$$

over a finite universe corresponds to a fully interconnected Markov network; the random variables in the network are obtained from all the ground atoms. This sentence says that the likelihood of friends of smokers being smokers themselves is very high. Likewise, when logic programs are decorated with probabilities, it enables reasoning about possible worlds and can capture intricate probabilistic models involving deterministic and stochastic constraints [43, 57]. For a slightly more complex version of the friends-smokers example from above in ProbLog [57], one such logic programming language, consider the below program:

```
0.1::stress(X) :- person(X).
0.2::friend(X,Y) :- person(X), person(Y).
smokes(X) :- stress(X).
smokes(X) :- friend(X,Y), smokes(Y).
```

This program says that the likelihood of a person being stressed is low, as is the likelihood of two people taken at random being friends. However, stress categorically leads to smoking, and friends of smokers are certainly smokers too.

Interestingly, such logical languages for probabilistic modeling also enables the use of logic-based inference technology, which we will be review further in the subsection below.

Formulating probabilistic graphical models as weighted logical formulas is an elegant approach to decouple the constraints and dependencies from the probabilistic parameters. However, at a conceptual level, there is little gained as we are still embedded in the framework of standard probability theory. When the representation language is thought of as a model of a system's mental state, we need to be able to reason about probabilistic events in a more general way. For example, we may need to compare the probabilities of hypothetical outcomes, or analyze the behavior of non-terminating action sequences. In some cases, we may even be ignorant about the actual probabilities of events. Such concerns were raised already by McCarthy and Hayes [58], who argued that assigning probabilities to quantifiers and logical connectives is not straightforward, and it should not be required that every statement be accorded a probability. This then leads to extensions of first-order logic that allow classical first-order sentences to be declared alongside weighted assertions, in addition to any dynamic and causal laws about how objects and actions interact in the world [59, 55, 60, 61]. For example, such languages can express statements such as:

- $\Pr(p) > \Pr(q)$ and $\Pr(p) < [a]\Pr(p)$
- $\Pr(\forall x\, Q(x)) \neq 0$
- $\Box(\Pr(p \vee q) = \Pr(p) + \Pr(q) - \Pr(p \wedge q))$

These say that (respectively): the probability of p is more than that of q, that of p becomes lesser after doing action a, the probability ascribed to all instances of Q being true is non-zero, and after any number of actions, the axiom of probability regarding the union of events necessarily holds. Consider, for example, p to denote that a robot is close to the wall; if a is the action of moving away from the wall then surely $\Pr(p) < [a]\Pr(p)$.

Although such logics are often intractable when taken in their full generality, the view taken is that as a language, it should be as general as possible: we are then in a position to investigate fragments that enjoy reasonable computational properties.

Before concluding this interlude, it is worth remarking about two classes of development. First, there are a number of results that allude to the statistical properties of logic. For example, an influential result by Fagin [62] established the Zero-One Law for first-order logic. For any first-order formula ϕ, consider the probability $\Pr_n$ of ϕ being true in first-order interpretations of size n. As n tends to infinity, $\Pr_n(\phi)$ will either be 0 or 1. Likewise, the discovery of a phase transition phenomena for propositional satisfiability [63] also implies profound probabilistic properties underlying logical reasoning. Second, probabilities are but one approach to quantify uncertainty, perhaps the most common and an obvious choice for many modeling situations. Of course, there are other approaches for specifying priors and defining posteriors [64]. One proposal that attempts to redefine logical truth altogether is fuzzy logic and related proposals [65]. These relax the truth value to atoms: in classical logic, we assign either a value of 1 or 0; in these logics, they can be any real number in the interval [0,1]. Consequently, the truth value of non-atomic formulas is then defined as an arithmetic function over such real numbers. As processing functions over such real numbers is simpler in some situations than reasoning about possible worlds as required by probability theory, real-valued logics are very popular in both statistical relational learning [66] and neuro-symbolic learning [67].

In what follows, we mostly restrict our attention to either classical or probabilistic logical languages, but as the above discussion indicates, other approaches are available, and they come with their own distinctive features.

1.3. Logic versus Machine Learning

To appreciate the role and impact of logic-based solvers for machine learning systems, it is perhaps useful to consider the core computational problem underlying (probabilistic) machine learning: the problem of inference, including evaluating the partition function and conditional probabilities of a probabilistic graphical model such as a Bayesian network.

When leveraging Bayesian networks for machine learning tasks [68], the networks are often learned using local search to maximize a likelihood or a Bayesian quantity. For example, given data D and the current guess for the network N, we might estimate the "goodness" of the guess by means of a score: $score(N,D) \propto \log \Pr(D \mid N) - size(N)$. That is, we want to maximize the fit of the data wrt the current guess, but we would like to penalize the model complexity, to avoid overfitting. Then, we would opt for a second guess N' only if $score(N',D) > score(N,D)$. Note that in as much as N represents a (scientific) hypothesis, and D represents the observations, we are very much following in the spirit of early work by Carnap and others on evaluating hypotheses via $\Pr(O \mid H)$.

1.3.1. Weighted Model Counting

Needless to say, even with a reasonable local search procedure for guessing hypotheses, the most significant computational effort here is that of probabilistic inference. Reasoning in such networks becomes especially challenging with logical syntax. Intuitively, where previously we wished to draw samples $z_1, \ldots, z_k$ from a distribution, we would now need to ensure that these samples are consistent with logical constraints that might need to be always true. These constraints can be thought of as making arbitrarily complex regions of the probability space invalid, and so can very challenging to efficiently sample from.

The prevalence of large-scale social networks, machine reading domains, and other types of relational knowledge bases has led to numerous formalisms that borrow the syntax of predicate logic for probabilistic modeling [69, 70, 71, 72]. This has led to a large family of solvers for the *weighted model counting* (WMC) problem [73, 74]. The idea is this: given a Bayesian network, a relational Bayesian network, a factor graph, or a probabilistic program [75], one considers an encoding of the formalism as a *weighted propositional theory*, consisting of a propositional theory Δ and a weight function w that maps literals in Δ to $\mathbb{R}^+$. Recall that SAT is the problem of finding an assignment to such a Δ, whereas #SAT counts the number of assignments for Δ. WMC extends #SAT by computing the sum of the weights of all assignments: that is, given a set of models that satisfy Δ, we evaluate the quantity $W(\Delta) = \sum_{M \models \Delta} w(M)$ where $w(M)$ is factorized in terms of the literals true at M. To obtain the conditional probability of a query q against evidence e (wrt the theory Δ), we define $\Pr(q \mid e) = W(\Delta \wedge q \wedge e)/W(\Delta \wedge e)$. (Although widely practiced, it turns out that factorization of the weight function in terms of literals is limiting: not only does it necessitate the need for additional variables and clauses, but it also constraints the space of discrete probability distributions that can be captured by WMC. See [76, 77] for discussions on alleviating this restriction.)

The popularity of WMC can be explained as follows. Its formulation elegantly decouples the logical or symbolic representation from the numeric representation, which is encapsulated in the weight function. When building solvers, this allows us to reason about logical equivalence and reuse SAT solving technology (such as constraint propagation and clause learning). WMC also makes it more natural to reason about deterministic, hard constraints in a probabilistic context [74]. For ideas on generating such representations randomly to assess scalability and compare inference algorithms, see [78], for example.

Exact WMC solvers are based on knowledge compilation [79, 80], exhaustive DPLL search [81] or pseudo-Boolean function manipulation [82, 76, 77]. Approximate WMC algorithms use local search [83] or sampling [84]. Here, knowledge compilation is worth particularly highlighting, where the logical theory is transformed to a data structure called a *circuit* [74]. Knowledge compilation [85] arose as a way to represent logical theories in a manner where certain kinds of computations (e.g., checking satisfiability) is significantly more effective, often polynomial in the size of the circuit. In the context of probabilistic inference, the idea was to then position probability estimation to also be computable in time polynomial in the size of the circuit [74, 86]. In the recent years, knowledge compilation has also been extended to probabilistic relational models (but in a finite domain setting) [87], referred to as *lifted inference.*

In sum, the discussion above suggests using WMC to compute $\Pr(D \mid N)$, especially if N has logical syntax. Implicit in the suggestion is that the probabilistic estimation machinery of WMC could also be used for tasks such as *parameter learning*. Roughly, if the hypothesis is a probabilistic representation, as is the case in this section, the *structure* determines the conditional dependencies between random variables, and the *parameters* determines the probability function associated with those variables. Parameter learning involves computing the likelihood of a guessed weight against the data, and so WMC can be used for that probabilistic estimation. But, of course, it is also possible to consider *structure learning* while using WMC for appropriate probabilistic estimation tasks. In particular, if N is logical artifact, then this is the case of using machine learning for logic – our next section – where we will consider this task in more detail. In the first instance, for example, [88] provides an end-to-end treatment of parameter and structure learning with probabilistic logical representations.

1.3.2. Beyond Propositional Languages

Lifted reasoning techniques attempt to leverage the relational structure of probabilistic logical models so as to avoid the exponentially large propositional theory that may result if the variables are substituted by constants. For example, the formula $\forall x(P(x) \vee Q(x))$ has 3^n models (out of the 4^n possible assignments for a domain of size n leading to $2n$ propositions in the ground theory). But still n is a fixed natural number. Thus, there is a computational benefit, but logically, we might as well assume a propositional language, and the formulation is limited to discrete random variables. At least on the logical front, a similar observation can be made for SAT, which for the longest time could only be applied in discrete domains. This changed with the increasing popularity of *satisfiability modulo theories* (SMT) [89], which enable us to, for example, reason about the satisfiability of linear constraints over the reals. Extending earlier insights on piecewise-polynomial weight functions [90, 91], the formulation of *weighted model integration* (WMI) was proposed in [92]. WMI extends WMC by leveraging the idea that SMT theories can represent mixtures of Boolean and continuous variables: for example, a formula such as $p \wedge (x > 5)$ denotes the logical conjunction of a Boolean variable p and a real-valued variable x taking values greater than 5. For every assignment to the Boolean and continuous variables, the WMI problem defines a weight. The total WMI is computed by integrating these weights over the domain of solutions to Δ, which is a mixed discrete-continuous (or simply *hybrid*) space. Consider, for example, the special case when Δ has no Boolean variables, and the weight of every model is 1. Then, the WMI simplifies to computing the volume of the polytope encoded in Δ. When we additionally allow for Boolean variables in Δ, this special case becomes the hybrid version of #SAT, known as #SMT [93]. In particular, previously we assumed that w mapped literals in Δ to $\mathbb{R}^+$, but now we will allow the mapping to any arithmetic expression to capture density functions. For example, mapping an interval $0 < x < 5$ to a number 0.3 would capture a piecewise-constant density, but if mapped to x^3, we would be admitting piecewise-polynomial densities. WMI is defined in much the same way as WMC as the sum of weights of models, except that we would need to additionally integrate the density function.

Since that proposal, numerous advances have been made on building efficient WMI solvers (e.g., [94, 95, 96]), including the development of knowledge compilation strategies for the extended language [97, 98, 99]. In [100], a lifted strategy for WMI is considered.

WMI proposes an extension of WMC for uncountably infinite (i.e., continuous) domains. What about countably infinite domains? The latter type is particularly useful for reasoning in quantified first-order settings, where we may say that a property such as $\forall x, y, z(\mathit{parent}(x, y) \wedge \mathit{parent}(y, z) \supset \mathit{grandparent}(x, z))$ applies to every possible x, y and z. Of course, in the absence of the finite domain assumption, reasoning in the first-order setting suffers from undecidability properties, and so various strategies have emerged for reasoning about an *open universe* [101]. One popular approach is to perform *forward reasoning*, where samples needed for probability estimation are obtained from the facts and declarations in the probabilistic model [101, 102]. Each such sample corresponds to a possible world. But there may be (countably or uncountably) infinitely many worlds, and so exact inference is usually sacrificed. A second approach is to restrict the model wrt the query and evidence atoms and define estimation from the resulting finite submodel [103, 104, 105], which may also be substantiated with exact inference in some cases [106, 107]. For example, in [106, 107], the following property is proved: given a first-order clausal theory Δ where the universal quantifier ranges over a countably infinite set and a ground query α, $\Delta \models \alpha$ can be determined via refutation in a propositional language with finitely many propositional variables. In general, if we are able to establish results where first-order properties in a restricted fragment can be reduced to propositional reasoning, we might look to consider reasoning and learning with probabilistic extensions to such fragments.

In the next section, we will be interested in the learning of logical artifacts. Given the successes of logic-based solvers for inference and probability estimation, the next section will also feature the application of such solvers to learning models with relational features and deterministic constraints.

1.3.3. Neurally-guided Search

Before turning to the next section, it is worth briefly reflecting where recent advances in machine learning, particularly deep learning, might play a role. This avenue is burgeoning so rapidly it would be difficult given the scope of this article to discuss representative ideas in any suitable detail. Thus, we will only mention a few interesting trends, for each of the three strands of logic vs learning (discussed immediately below), learning for logic and logic for learning (in subsequent subsections).

To begin with, the learning of hypothesis and representations is clearly one venue where mainstream learners could prove useful, but we defer this to the next section. As far as inference is particularly concerned, there have been initiatives to search for logically consistent artifacts via deep learning. (The precursor of this is perhaps classical clausal search in inductive logic programming, defined by the generality of the clauses, over some structured hypothesis space.) These artifacts could be satisfying assignments, program completions, and more generally, theorems. For example, in [108] the discovery of first-order theorems is enabled by empowering the search of proofs with learning. In [109], the search for solutions for NP-complete problems is explored via graph neural networks [7]. In [110], the completion of programs against a partial specification – i.e., partially-specified program – is augmented using learning. The key trade-off worth noting here is the balance of prediction against explicit symbolic reasoning to achieve maximal logical consistency. Finally, in [111], the approximation of WMC is explored via deep learning.

1.4. Machine Learning for Logic

Expert knowledge is hard to come by, difficult to articulate at different levels of granularity and is often brittle. We need a mechanism for obtaining axiomatic knowledge.

1.4.1. Entailment versus Bayesian Scoring

The learning of logical and symbolic artifacts is an important issue in AI, and computer science more generally [45]. There is a considerable body of work on learning propositional and relational formulas, and in context of probabilistic information, learning weighted formulas [50, 112, 113, 1]. Approaches can be broadly lumped together as follows:

1. *Entailment-based scoring:* Given a logical language L, background knowledge $B \subset L$, examples D (usually a set of L-atoms), find a hypothesis $H \in \overline{H}, H \subset L$ such that $B \cup H$ entail the instances in D. Here, the set $\overline{H}$ places restrictions of the syntax of H so as to control model complexity and generalization. (For example, $H = D$ is a trivial hypothesis that satisfies the entailment stipulation.)
2. *Likelihood-based scoring:* Given L, B and D as defined above, find $H \subset L$ such that $score(H,D) > score(H',D)$ for every $H' \neq H$. As discussed before, we might define $score(H,D) \propto \log \Pr(D \mid H) - size(H)$. Here, like $\overline{H}$ above, $size(H)$ attempts to the control model complexity and generalization.

Generally speaking, the idea is to find hypotheses that do not assume more information than provided, but nonetheless capture all of the examples. In that regard, entailment-based approaches attempt to find the *least general* clause, in the sense that the hypothesis is a clause that entails all and only the examples. Likelihood-based scoring attempt to find a suitable representation that captures the *distribution of the data*, in the sense that sampling from the representation will recover the data.

In the most standard setting, Bayesian network learning algorithms would be of the first type, and inductive logic programming of the second type. But in many recent works, the distinction is not strict. For example, we may use entailment-based inductive synthesis for an initial estimate of the hypothesis, and then resort to Bayesian scoring models to refine that hypothesis [70]. The synthesis step might invoke neural machinery [14]. We might not require that the hypothesis entails every example in D but only the largest consistent subset, which is sensible when we expect the examples to be noisy [113], or if the search for such a comprehensive hypothesis is infeasible or computationally expensive. We might compile B to circuit, and perform Bayesian scoring on that structure [114], and so B could be seen as deterministic domain-specific constraints. Finally, we might stipulate the conditions under which a "correct" hypothesis may be inferred wrt unknown ground truth, only a subset of which is provided in D. This is perhaps best represented by the (probably approximately correct) PAC-semantics that captures the quality possessed by the output of learning algorithm whilst costing for the number of (noisy) examples that need to be observed [115, 116, 117].

Broadly speaking, then, there are *three major flavors of learning*: statistical relational learning, inductive logic programming, and learning under PAC-semantics. Not surprisingly, there are significant differences in terms of the learning regime, the notion of correctness, and the underlying algorithmic machinery, although as suggested above, there continues to be a cross-pollination of ideas. For example, fragments of inductive logic programming can be shown to be PAC-learnable [118], in the sense of postulating the conditions under which the learning is tractable. However, as mentioned, inductive logic programming treats the input examples as defining a domain for which we aim to construct a hypothesis. In PAC-Semantics, on the one hand, we only seek to bound the probability that queries are true. But on the other, the framework allows for some unknown distribution over models of the underlying theory, from which we can sample. We never assume access to complete knowledge of this theory, or even the distribution. This makes the setting significantly different from inductive logic programming.

It is worth noting that many of these ideas carry over to the learning of symbolic (but not explicitly logical) artifacts, such as programs [45, 110, 119].

1.4.2. Learning in Infinite Spaces

Recall the interplay between SAT and probabilistic inference, on the one hand, and probabilistic inference and learning, on the other. By means of logical and probabilistic reasoning tools for arithmetic fragments such as SMT and WMI, it should be clear that it now becomes possible to extend learning schemes to continuous and mixed discrete-continuous spaces. All three flavours of learning have, in fact, been adapted to the continuous setting in the recent years.

For example, one could learn SMT formulas by guessing a sentence that entails all and only the examples [120]. For an account that also tries to evaluate a SMT formula hypothesis that is correct wrt unknown ground truth via the PAC-semantics, see [121]. In [122], it is further shown how the PAC-semantics approach could be compared against other induction schemes for such arithmetic fragments.

If the overall objective is to obtain a distribution of the data, other possibilities present themselves. For example, extending the work on learning SMT formulas, the learning of weighted SMT formulas is also considered in [120]. That is, just as learning probabilistic propositional and relational sentences corresponds to representations for which we can compute WMC, the learning of weighted SMT sentences essentially corresponds to learning WMI formulations.

Such ideas could be integrated with circuits, where as discussed, probability estimation was positioned to be computable in time polynomial in the size of the structure [74, 86]. In [123, 124], by means of likelihood-based scoring, probabilistic circuits are constructed, where the internal nodes of the structure would represent arithmetic formulas, that might be additionally decorated with probability mass or density functions.

WMI, and closely-related notions, can also be considered with rule learning. In [125], real-valued data points are first lumped together to obtain atomic continuous random variables. From these, relational formulas are constructed so as to yield hybrid probabilistic programs. The rule learning is based on likelihood scoring. In [126], the real-valued data points are first intervalized, and polynomials are learned for those intervals based on likelihood scoring. These weighted atoms are then used for learning clauses by entailment judgements, which is an extension to the discrete setting [113].

What about countably infinite domains? There is a long history in philosophical logic on learning universals [31, 33], as indicated in our discussion on Aristotle. Early in the history of inductive synthesis, numerous proposals were considered in terms of learning in the limit [127, 36, 128]. In most pragmatic instances on learning logical artifacts, however, the difference between the uncountable and countably infinite setting is this: in the former, we see finitely many real-valued samples as being drawn from an (unknown) interval, and we could inspect these samples to crudely infer a lower and upper bound. In the latter, based on finitely many relational atoms, we would need to infer a universally quantified clause, such as $\forall x, y, z(parent(x, y) \wedge parent(y, z) \supset grandparent(x, z))$. If we are after a hypothesis that is simply guaranteed to be consistent wrt the observed examples, then standard rule induction strategies might suffice [112], and we could naively interpret the rules as quantifying over a countably infinite domain. But this is somewhat unsatisfactory, as there is no distinction between the rules learned in the standard finite setting and its supposed applicability to the infinite setting. What is really needed is an analysis of what rule learning would mean wrt the infinitely many examples that have *not* been observed. This was recently considered via the PAC-semantics in [129], by appealing to ideas on reasoning with open universes discussed earlier [106].

1.4.3. Gradient-based Structure Learning

Because data can be noisy, and there is a need to build representations at scale, neural approaches to rule induction are on the rise. In [14, 130], for example, inductive logic programming is reconsidered in a neural setting, by extracting the program from a latent representation. In [131], a latent representation of a probabilistic relational model is considered. In [67], a fuzzy logical semantics is shown to capture the representation and reasoning of neural network outputs.

Before concluding this section, it is worth noting that although the above discussion is primarily related to the learning of logical artifacts, it can equivalently be seen as a class of machine learning methods that leverage symbolic domain knowledge. Indeed, logic-based probabilistic inference over deterministic constraints, and entailment-based induction augmented with background knowledge are instances of such a class [132, 6]. Analogously, the automated construction of relational and statistical knowledge bases [133, 134] by combining background knowledge with extracted tuples (obtained, for example, by applying natural language processing techniques to large textual data) is another instance of such a class [135, 136]. In the next section, we will consider yet another way in which logical and symbolic artifacts can influence learning: we will see how such logical artifacts are useful to enable tractability, correctness, modularity and compositionality.

1.5. Logic for Machine Learning

There are two obvious ways in which a logical framework can provide insights on machine learning theory. First, consider that computational tractability is of central concern when applying logic in computer science, knowledge representation, database theory and search [137, 138, 139]. Thus, the natural question to wonder is whether these ideas would carry over to probabilistic machine learning. On the one hand, probabilistic extensions to tractable knowledge representation frameworks could be considered [140]. But on the other, as discussed previously, ideas from knowledge compilation, and the use of circuits, in particular, are proving very effective for designing tractable paradigms for machine learning. While there has always been an interest in capturing tractable distributions by means of low tree-width models [141], knowledge compilation has provided a way to also represent high tree-width models and enable exact inference for a range of queries [79, 142, 86, 114]. See [143] for a comprehensive view on the use of knowledge compilation for machine learning.

The other obvious way logic can provide insights on machine learning theory is by offering a formal apparatus to reason about *context*. Machine learning problems are often positioned as atomic tasks, such as a classification task where regions of images need to be labeled as cats or dogs. However, even in that limited setting, we imagine the resulting classification system as being deployed as part of a larger system, which includes various modules that communicate or interface with the classification system. We imagine an implicit accountability to the labelling task in that the detected object is either a cat or a dog, but not both. If there is information available that all the entities surrounding the object of interest have been labelled as lions, we might want to (heuristically) accord a high probability to the object being a cat, possibly a wild cat. There is a very low chance of the object being a dog, then. If this is part of a vision system on a robot, we should ensure that the robot never tramples on the object, regardless of whether it is a type of cat or a dog. To inspect such patterns, and provide meta-theory for machine learning, it can be shown that symbolic, programmatic and logical artifacts are extremely useful. We will specifically consider correctness, modularity and compositionality to explore the claim.

We remark that there is not much to be said about the distinction between finite versus infinite wrt most of these properties. In principle, many of these ideas are likely amenable to extensions to an infinite setting in the ways discussed in the previous sections (e.g., considering constraints of a continuous or a countably infinite nature). So we will not dwell on the size of the domain in quite an explicit manner as before.

1.5.1. Correctness

On the topic of correctness, the classical framework in computer science is *verification*: can we provide a formal specification of what is desired, and can the system be checked against that specification? In machine learning, we might ask whether the system, during or after training, satisfies a specification. The specification here might mean constraints about the physical laws of the domain, or notions of perturbation in the input space while ensuring that the labels do not change, or insisting that the prediction does not label an object as being both a cat and a dog, or otherwise ensuring that outcomes are not subject to, say, gender bias. Although there is a broad body of work on such issues, touching more generally on *trust* [144], we discuss approaches closer to the thrust of this article. For example, [145] show that a trained neural network can be verified by means of an SMT encoding of the network. In recent work, [146] show that the loss function of deep learning systems can be adjusted to logical constraints by insisting that the distribution on the predictions is proportional to the weighted model count of those constraints. In [114], prior (logical) constraints are compiled to a circuit to be used for probability estimation. In [147], circuits are shown to be amenable to training against probabilistic and causal prior constraints, including assertions about fairness, for example. In [148], fairness is formalized in a probabilistic relational language to enable equitable properties in, say, social networks by explicitly relating the (human) entities of the domain.

1.5.2. Modularity

In [149, 15], a somewhat different approach to respecting domain constraints is taken: the low-level prediction is obtained as usual from a machine learning module, which is then interfaced with a probabilistic relational language and its reasoning apparatus. That is, the reasoning is positioned to be tackled directly by the symbolic engine. In a sense, such approaches cut across the three strands: the symbolic engine uses weighted model counting, the formulas in the language could be obtained by (say) entailment-based scoring, and the resulting language supports modularity and compositionality (discussed below).

On the topic of modularity, recall that the general idea is to reduce, simplify or otherwise abstract a (probabilistic) computation as an atomic entity, which is then to be referenced in another, possibly more complex, entity. (In this regard, modularity also aids in assessing the correctness of systems since one can inspect the modules independently too, in addition to the overall behavior.) In standard programming languages, this might mean the compartmentalization and interrelation of computational entities. For machine learning, approaches such as probabilistic programming [150, 151] support probabilistic primitives in the language, with the intention of making learning modules re-usable and modular. It can be shown, for example, that the computational semantics of some of these languages reduce to WMC [152, 153]. Thus, in the infinite case, a corresponding reduction to WMI follows [154, 126, 155].

A second dimension to modularity is the notion of *abstraction*. Here, we seek to model, reason and explain the behavior of systems in a more tractable search space, by omitting irrelevant details. The idea is widely used in natural and social sciences. Think of understanding the political dynamics of elections by studying micro level phenomena (say, voter grievances in counties) versus macro level events (e.g., television advertisements, gerrymandering). In particular, in computer science, it is often understood as the process of mapping one representation onto a simpler representation by suppressing irrelevant information. In fact, integrating low-level behavior with high-level reasoning, exploiting relational representations to reduce the number of inference computations, and many other search space reduction techniques can all loosely be seen as instances of abstraction [156].

While there has been significant work on abstraction in deterministic systems [157], for machine learning, however, a probabilistic variant is clearly needed. In [158], an account of abstraction for loop-free propositional probabilistic programs is provided, where certain parts of the program (possibly involving continuous properties) can be reduced to a Bernoulli random variable. For example, suppose every occurrence of the continuous random variable x, drawn uniformly on the interval [0,1], in a program is either of the form $x \leq 7$ or of the form $x > 7$. Then, we could use a discrete random variable b with a 0.7 probability of being true to capture $x \leq 7$; and analogously, $\neg b$ to capture $x > 7$. The resulting program is likely to be simpler. In [156], an account of abstraction for probabilistic relational models is considered, where the notion of abstraction also extends to deterministic constraints and complex formulas. For example, a single probabilistic variable in the abstracted model could denote a complex logical formula in the original model. Moreover, the logical properties that enable verifying and inducing abstractions are also considered, and it is shown how WMC is sufficient for the computability of these properties (also see [153]).

Incidentally, abstraction brings to light a reduction from infinite to finite: it is shown in [156] that the modeling of piecewise densities as weighted propositions, which is leveraged in WMI [92, 154], is a simple case of the more general account. Therefore, it is worthwhile to investigate whether this or other accounts of abstraction could emerge as general-purpose tools that allow us to inspect the conditions under which infinitary statements reduce to finite computations.

A broader point here is the role abstraction might play in generating explanations [159, 160]. For example, a user's understanding of the domain is likely to be different from the low-level data that a machine learning system interfaces with [161], and so, abstractions can establish a map these two levels in a formal way.

1.5.3. Compositionality

Finally, we turn to the topic of compositionality, which, of course, is closely related to modularity in that we want to distinct modules to come together to form a complex composition. Not surprisingly, this is of great concern in AI, as it is widely acknowledged that most AI systems will involve heterogeneous components, some of which may involve learning from data, and others reasoning, search and symbol manipulation [9]. In continuation with the above discussion, probabilistic programming is one such endeavor that purports to tackle this challenge by allowing modular components to be composed over programming and/or logical connectives [150, 151, 70, 15, 149, 162, 163, 164, 165, 61]. (See [139, 166, 167] for ideas in deterministic systems.) However, probabilistic programming only composes probabilistic computations, but does not offer an obvious means to capture other types of search-based computations, such as SAT, and integer and convex programming.

Recall that the computational semantics of probabilistic programs reduces to WMC [152, 153]. Following works such as [168, 169], an interesting observation made in [170] is that by appealing to a sum of products computation over different semiring structures, we can realize a large number of tasks such as satisfiability, unweighted model counting, sensitivity analysis, gradient computations, in addition to WMC. It was then shown in [171] that the idea could be generalized further for infinite domains: by defining a measure on first-order models, WMI and convex optimization can also be captured. As the underlying language is a logical one, composition can already be defined using logical connectives. But an additional, more involved, notion of composition is also proposed, where a sum of products over different semirings can be concatenated. To reiterate, the general idea behind these proposals [169, 170, 171] is to arrive at a principled paradigm that allows us to interface learned modules with other types of search and optimization computations for the compositional building of AI systems. See also [172] for analogous discussions, but where a different type of coupling for the underlying computations is suggested. Overall, we observed that a formal apparatus (symbolic, programmatic and logical artifacts) help us define such compositional constructions by providing a meta-theory.

1.5.4. Empowering Neural Artifacts

As argued above, concerns of correctness, modularity and compositionality apply to almost all of machine learning. Given the vast number of parameter and architectural choices in deep learning, such concerns are particularly prominent. We have reviewed corresponding advances in the above sections, but we reiterate the key ideas for the sake of completeness.

On the topic of correctness, ensuring that the training of neural networks respect safety and physical constraints has received considerable attention [145, 146], ranging from the use of SMT encodings to verify the sensitivity of trained models to the use of WMC for capturing domain knowledge whilst training models.

On the topic of modularity and compositionality, a general observation is that it is clearly advantageous to combine probabilistic computations from various modules, neural or otherwise. We previously argued for the use of probabilistic programming and related proposals to systematically capture such interactions. Consequently, probabilistic programming languages that allow neural components are in active development [173]. In [169, 174], for example, neural computations are positioned as one of the many possible types of computations in a semiring framework. This is very much in the spirit of semiring-based generalizations to WMC discussed earlier [170, 171]. Likewise, abstraction can be seen as a simple schema for unifying deep learning and traditional symbolic reasoning. For example, much of the work on neuro-symbolic systems attempt to integrate neural network outputs as abstract (and external) predicates in a logical framework [5, 67, 6, 8, 175].

Naturally, much work remains to be done on how the integration between neural and logical computations can be made deeper, in the sense of making the best possible tradeoffs between learning and reasoning. But, from our discussions, it is clear that logical methods provide a powerful and rigorous regime to establish those tradeoffs and develop meta-theory for constructing hybrid models.

1.6. Conclusions

In this article, we surveyed work that provides further evidence for the connections between logic and learning. Our narrative was structured in terms of three strands: logic versus learning, machine learning for logic, and logic for machine learning, but naturally, there was considerable overlap.

We covered a large body of work on what these connections look like, including, for example, pragmatic concerns such as the use of hard, domain-specific constraints and background knowledge, all of which considerably eases the requirement that all of the agent's knowledge should be derived from observations alone. (See discussions in [176, 11] on the limitations of learned behavior, for example.) Where applicable, we placed an emphasis on how extensions to infinite domains are possible. Moreover, logical artifacts can help in constraining, simplifying and/or composing machine learning entities, and in providing a principled way to study the underlying representational and computational issues.

In general, this type of work could help us move beyond the narrow focus of the current learning literature so as to deal with time, space, abstraction, causality, quantified generalizations, relational abstractions, unknown domains, unforeseen examples, among other things, in a principled fashion. In fact, what is being advocated is the tackling of problems that symbolic logic and machine learning might struggle to address individually. One could even think of the need for a recursive combination of the strands: purely reactive components interact with purely cogitative elements, but then those reactive components are learned against domain constraints, and the cogitative elements are induced from data, and so on. More broadly, making progress towards a formal realization of *System 1* versus *System 2* processing might also contribute to our understanding of human intelligence, and if not that, at least capture the human-like capability of balancing reaction versus deliberation in automated systems.

Acknowledgements. This chapter is a significantly extended version of a previous article by the author [177]. The author was supported by a Royal Society University Research Fellowship. The author is grateful to Paulius Dilkas, Kwabena Nuamah, Alan Bundy, and Luc De Raedt for comments.

References

[1] Raedt LD, Kersting K, Natarajan S, et al. Statistical relational artificial intelligence: Logic, probability, and computation. Synthesis Lectures on Artificial Intelligence and Machine Learning. 2016;10(2):1–189.
[2] Getoor L, Taskar B, editors. An introduction to statistical relational learning. MIT Press; 2007.
[3] Lakemeyer G, Levesque HJ. Cognitive robotics. In: Handbook of knowledge representation. Elsevier; 2007. p. 869–886.
[4] Kaelbling LP, Lozano-Pérez T. Integrated task and motion planning in belief space. I J Robotic Res. 2013;32(9-10):1194–1227.
[5] Garcez Ad, Gori M, Lamb LC, et al. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. arXiv preprint arXiv:190506088. 2019;.
[6] De Raedt L, Manhaeve R, Dumancic S, et al. Neuro-symbolic= neural+ logical+ probabilistic. In: NeSy'19@ IJCAI, the 14th International Workshop on Neural-Symbolic Learning and Reasoning; 2019. p. 1–4.
[7] Lamb L, Garcez A, Gori M, et al. Graph neural networks meet neural-symbolic computing: A survey and perspective. arXiv preprint arXiv:200300330. 2020;.
[8] Sarker MK, Zhou L, Eberhart A, et al. Neuro-symbolic artificial intelligence current trends. arXiv preprint arXiv:210505330. 2021;.
[9] Marcus G, Davis E. Rebooting ai: Building artificial intelligence we can trust. Pantheon; 2019.
[10] Zellers R, Bisk Y, Schwartz R, et al. Swag: A large-scale adversarial dataset for grounded commonsense inference. arXiv preprint arXiv:180805326. 2018;.
[11] Kambhampati S. Polanyi's revenge and ai's new romance with tacit knowledge. Communications of the ACM. 2021;64(2):31–32.
[12] Bunel R, Hausknecht M, Devlin J, et al. Leveraging grammar and reinforcement learning for neural program synthesis. arXiv preprint arXiv:180504276. 2018;.
[13] Xu K, Li J, Zhang M, et al. What can neural networks reason about? arXiv preprint arXiv:190513211. 2019;.
[14] Evans R, Grefenstette E. Learning explanatory rules from noisy data. Journal of Artificial Intelligence Research. 2018;61:1–64.
[15] Manhaeve R, Dumancic S, Kimmig A, et al. Deepproblog: Neural probabilistic logic programming. In: Advances in Neural Information Processing Systems; 2018. p. 3749–3759.
[16] Kahneman D. Thinking, fast and slow. Macmillan; 2011.
[17] Feeney A, Heit E. Inductive reasoning: Experimental, developmental, and computational approaches. Cambridge University Press; 2007.
[18] Russell B. History of western philosophy: Collectors edition. Routledge; 2013.
[19] Beaney M. The oxford handbook of the history of analytic philosophy. Oxford University Press; 2013.
[20] Levesque HJ. Thinking as computation: A first course. MIT Press; 2012.
[21] Hanna R. Rationality and logic. MIT Press; 2009.
[22] van Benthem J, Ter Meulen A. Handbook of logic and language. Elsevier; 1996.
[23] Gabbay DM, Woods JH. Handbook of the history of logic. Vol. 2009. Elsevier North-Holland; 2004.
[24] Smullyan R. First-order logic. Dover Publications; 1995.
[25] Hughes GE, Cresswell MJ. A companion to modal logic. Routledge; 1984.

[26] Kowalski RA. Predicate logic as programming language. In: IFIP Congress; 1974. p. 569–574.

[27] Biere A, Heule M, van Maaren H. Handbook of satisfiability. Vol. 185. IOS press; 2009.

[28] Muggleton S. Inductive logic programming. New generation computing. 1991;8(4):295–318.

[29] Lange M. Hume and the problem of induction. In: Handbook of the history of logic. Vol. 10. Elsevier; 2011. p. 43–91.

[30] Carnap R, Jeffrey RC. Studies in inductive logic and probability. Vol. 2. Univ of California Press; 1980.

[31] Hintikka J. Carnap and essler versus inductive generalization. Erkenntnis. 1975;9(2):235–244.

[32] Carnap R. Logical foundations of probability. Routledge and Kegan Paul London; 1951.

[33] Zabell SL. Carnap and the logic of inductive inference. In: Handbook of the history of logic. Vol. 10. Elsevier; 2011. p. 265–309.

[34] Plotkin GD. A note on inductive generalization. Machine intelligence. 1970;5(1):153–163.

[35] Michalski RS. A theory and methodology of inductive learning. In: Machine learning. Elsevier; 1983. p. 83–134.

[36] Shapiro EY. Inductive inference of theories from facts. Yale University, Department of Computer Science; 1981.

[37] De Raedt L, Frasconi P, Kersting K, et al., editors. Probabilistic inductive logic programming: theory and applications. Berlin, Heidelberg: Springer-Verlag; 2008.

[38] Quinlan JR. Learning first-order definitions of functions. Journal of artificial intelligence research. 1996;5:139–161.

[39] Peirce CS. Collected papers of charles sanders peirce. Vol. 7. Harvard University Press; 1974.

[40] Carson D. The abduction of sherlock holmes. International Journal of Police Science &amp; Management. 2009;11(2):193–202.

[41] Tsamoura E, Michael L. Neural-symbolic integration: A compositional perspective. arXiv preprint arXiv:201011926. 2020;.

[42] Dai WZ, Muggleton SH. Abductive knowledge induction from raw data. arXiv preprint arXiv:201003514. 2020;.

[43] Poole D. Abducing through negation as failure: Stable models within the independent choice logic. The Journal of Logic Programming. 2000;44(1-3):5–35.

[44] Poole D. Probabilistic horn abduction and bayesian networks. Artificial Intelligence. 1993;64(1):81–129.

[45] Gulwani S. Dimensions in program synthesis. In: Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming; 2010. p. 13–24.

[46] Genesereth MR, Nilsson NJ. Logical foundations of artificial intelligence. Morgan Kaufmann; 2012.

[47] Bradley AR, Manna Z. The calculus of computation: decision procedures with applications to verification. Springer Science & Business Media; 2007.

[48] Kersting K, Natarajan S, Poole D. Statistical relational AI: Logic, probability and computation. 2011;.

[49] Cropper A, Dumancic S. Inductive logic programming at 30: a new introduction. Journal of Artificial Intelligence Research. 1993;1:1–15.

[50] Benedikt M, Kersting K, Kolaitis PG, et al. Logic and learning (dagstuhl seminar 19361). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik; 2020.

[51] Brewka G, Delgrande JP, Romero J, et al. asprin: Customizing answer set preferences without a headache. In: AAAI; 2015. p. 1467–1474.

[52] Levesque H, Reiter R, Lespérance Y, et al. Golog: A logic programming language for dynamic domains. Journal of Logic Programming. 1997;31:59–84.

[53] McCarthy J. Programs with common sense. In: Semantic Information Processing. MIT Press; 1968. p. 403–418.

[54] Nilsson NJ. Probabilistic logic. Artificial intelligence. 1986;28(1):71–87.

[55] Halpern J. An analysis of first-order logics of probability. Artificial Intelligence. 1990;46(3):311–350.

[56] Bacchus F. Representing and reasoning with probabilistic knowledge. MIT Press; 1990.

[57] Raedt LD, Kimmig A, Toivonen H. Problog: A probabilistic prolog and its application in link discovery. In: Proc. IJCAI; 2007. p. 2462–2467.

[58] McCarthy J, Hayes PJ. Some philosophical problems from the standpoint of artificial intelligence. In: Machine Intelligence; 1969. p. 463–502.

[59] Kooi B. Probabilistic dynamic epistemic logic. Journal of Logic, Language and Information. 2003; 12(4):381–408.

[60] Fagin R, Halpern JY. Reasoning about knowledge and probability. J ACM. 1994;41(2):340–367.

[61] Belle V. Logic meets probability: Towards explainable AI systems for uncertain worlds. In: IJCAI; 2017.

[62] Fagin R. Probabilities on finite models. The Journal of Symbolic Logic. 1976;41(1):50–58.

[63] Mitchell DG, Selman B, Levesque HJ. Hard and easy distributions of sat problems. In: Proc. AAAI; 1992. p. 459–465.

[64] Shafer G. Perspectives on the theory and practice of belief functions. International Journal of Approximate Reasoning. 1990;4(5–6):323 – 362.

[65] Giles R. Łukasiewicz logic and fuzzy set theory. International Journal of Man-Machine Studies. 1976; 8(3):313–327.

[66] Bach SH, Broecheler M, Huang B, et al. Hinge-loss markov random fields and probabilistic soft logic. arXiv preprint arXiv:150504406. 2015;.

[67] Serafini L, Garcez Ad. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. arXiv preprint arXiv:160604422. 2016;.

[68] Koller D, Friedman N. Probabilistic graphical models - principles and techniques. MIT Press; 2009.

[69] Niu F, Ré C, Doan A, et al. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. Proceedings of the VLDB Endowment. 2011;4(6):373–384.

[70] Richardson M, Domingos P. Markov logic networks. Machine learning. 2006;62(1):107–136.

[71] Poole D. First-order probabilistic inference. In: Proc. IJCAI; 2003. p. 985–991.

[72] Suciu D, Olteanu D, Ré C, et al. Probabilistic databases. Synthesis Lectures on Data Management. 2011;3(2):1–180.

[73] Gomes CP, Sabharwal A, Selman B. Model counting. In: Handbook of satisfiability. IOS Press; 2009.

[74] Chavira M, Darwiche A. On probabilistic inference by weighted model counting. Artificial Intelligence. 2008;172(6-7):772–799.

[75] Renkens J, Shterionov D, Van den Broeck G, et al. ProbLog2: From probabilistic programming to statistical relational learning. In: Roy D, Mansinghka V, Goodman N, editors. Proceedings of the NIPS Probabilistic Programming Workshop,; Dec.; 2012.

[76] Dilkas P, Belle V. Weighted model counting without parameter variables. In: SAT; 2021.

[77] Dilkas P, Belle V. Weighted model counting with conditional weights for Bayesian networks. In: UAI; 2021.

[78] Dilkas P, Belle V. Generating random logic programs using constraint programming. In: CP; 2020.

[79] Chavira M, Darwiche A. Compiling Bayesian networks with local structure. In: IJCAI; Vol. 19; 2005. p. 1306.

[80] Muise C, McIlraith S, Beck J, et al. D sharp: Fast d-DNNF compilation with sharpSAT. Advances in Artificial Intelligence. 2012;:356–361.

[81] Sang T, Beame P, Kautz HA. Performing bayesian inference by weighted model counting. In: AAAI; 2005. p. 475–482.

[82] Dudek JM, Phan VH, Vardi MY. Dpmc: Weighted model counting by dynamic programming on project-join trees. In: International Conference on Principles and Practice of Constraint Programming; Springer; 2020. p. 211–230.

[83] Wei W, Selman B. A new approach to model counting. In: Theory and Applications of Satisfiability Testing; Springer; 2005. p. 96–97.

[84] Chakraborty S, Fremont DJ, Meel KS, et al. Distribution-aware sampling and weighted model counting for SAT. In: AAAI; 2014. p. 1722–1730.

[85] Darwiche A, Marquis P. A knowledge compilation map. J Artif Intell Res. 2002;17:229–264.

[86] Poon H, Domingos P. Sum-product networks: A new deep architecture. UAI. 2011;:337–346.

[87] Van den Broeck G. Lifted Inference and Learning in Statistical Relational Models [dissertation]. KU Leuven; 2013.

[88] Haaren JV, den Broeck GV, Meert W, et al. Lifted generative learning of markov logic networks. Machine Learning. 2016;103(1):27–55.

[89] Barrett C, Sebastiani R, Seshia SA, et al. Satisfiability modulo theories. In: Handbook of satisfiability. Chapter 26. IOS Press; 2009. p. 825–885.

[90] Shenoy P, West J. Inference in hybrid Bayesian networks using mixtures of polynomials. International Journal of Approximate Reasoning. 2011;52(5):641–657.

[91] Sanner S, Abbasnejad E. Symbolic variable elimination for discrete and continuous graphical models. In: AAAI; 2012.

[92] Belle V, Passerini A, Van den Broeck G. Probabilistic inference in hybrid domains by weighted model

integration. In: IJCAI; 2015. p. 2770–2776.

[93] Chistikov D, Dimitrova R, Majumdar R. Approximate counting in SMT and value estimation for probabilistic programs. In: Tacas. Vol. 9035; 2015. p. 320–334.

[94] Morettin P, Passerini A, Sebastiani R. Advanced smt techniques for weighted model integration. Artificial Intelligence. 2019;275:1–27.

[95] Merrell D, Albarghouthi A, D'Antoni L. Weighted model integration with orthogonal transformations. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence; 2017.

[96] Zeng Z, Van den Broeck G. Efficient search-based weighted model integration. arXiv preprint arXiv:190305334. 2019;.

[97] Kolb S, Mladenov M, Sanner S, et al. Efficient symbolic integration for probabilistic inference. In: IJCAI; 2018.

[98] Kolb S, Morettin P, Zuidberg Dos Martires P, et al. The pywmi framework and toolbox for probabilistic inference using weighted model integration. IJCAI. 2019;.

[99] Zuidberg Dos Martires P, Dries A, De Raedt L. Knowledge compilation with continuous random variables and its application in hybrid probabilistic logic programming. arXiv preprint arXiv:180700614. 2018;.

[100] Feldstein J, Belle V. Lifted reasoning meets weighted model integration. In: UAI; 2021.

[101] Russell SJ. Unifying logic and probability. Commun ACM. 2015;58(7):88–97.

[102] Gutmann B, Thon I, Kimmig A, et al. The magic of logical inference in probabilistic programming. Theory and Practice of Logic Programming. 2011;11(4-5):663–680.

[103] Milch B, Marthi B, Sontag D, et al. Approximate inference for infinite contingent bayesian networks. In: AISTATS; 2005. p. 238–245.

[104] Singla P, Domingos PM. Markov logic in infinite domains. In: UAI; 2007. p. 368–375.

[105] Grohe M, Lindner P. Probabilistic databases with an infinite open-world assumption. In: Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems; 2019. p. 17–31.

[106] Belle V. Open-universe weighted model counting. In: AAAI; 2017. p. 3701–3708.

[107] Belle V. Weighted model counting with function symbols. In: UAI; 2017.

[108] Loos S, Irving G, Szegedy C, et al. Deep network guided proof search. arXiv preprint arXiv:170106972. 2017;.

[109] Prates M, Avelar PH, Lemos H, et al. Learning to solve np-complete problems: A graph neural network for decision tsp. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 33; 2019. p. 4731–4738.

[110] Nye M, Hewitt L, Tenenbaum J, et al. Learning to infer program sketches. In: International Conference on Machine Learning; PMLR; 2019. p. 4861–4870.

[111] Abboud R, Ceylan I, Lukasiewicz T. Learning to reason: Leveraging neural networks for approximate dnf counting. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 34; 2020. p. 3097–3104.

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

[113] De Raedt L, Dries A, Thon I, et al. Inducing probabilistic relational rules from probabilistic examples. In: Twenty-Fourth International Joint Conference on Artificial Intelligence; 2015.

[114] Liang Y, Bekker J, Van den Broeck G. Learning the structure of probabilistic sentential decision diagrams. In: Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI); 2017.

[115] Valiant LG. Robust logics. Artificial Intelligence. 2000;117(2):231–253.

[116] Cohen WW. PAC-learning nondeterminate clauses. In: AAAI; 1994. p. 676–681.

[117] Grohe M, Ritzert M. Learning first-order definable concepts over structures of small degree. In: 2017 32nd annual ACM/IEEE symposium on logic in computer science (LICS); IEEE; 2017. p. 1–12.

[118] De Raedt L, Džeroski S. First-order *jk*-clausal theories are PAC-learnable. Artificial Intelligence. 1994; 70(1):375–392.

[119] Ellis K, Solar-Lezama A, Tenenbaum J. Sampling for bayesian program learning. In: NIPS; 2016. p. 1289–1297.

[120] Kolb S. Learn+ solve: Learning and solving constrained hybrid inference problems [dissertation]. KU Leuven; 2019.

[121] Mocanu IG, Belle V, Juba B. Polynomial-time implicit learnability in smt. In: ECAI; 2020.

[122] Rader AP, Mocanu IG, Belle V, et al. Learning implicitly with noisy data in linear arithmetic. IJCAI.

2021;.

[123] Molina A, Vergari A, Di Mauro N, et al. Mixed sum-product networks: A deep architecture for hybrid domains. In: Thirty-second AAAI conference on artificial intelligence; 2018.

[124] Bueff A, Speichert S, Belle V. Tractable querying and learning in hybrid domains via sum-product networks. KR Workshop on Hybrid Reasoning. 2018;.

[125] Nitti D, Ravkic I, Davis J, et al. Learning the structure of dynamic hybrid relational models. In: Proceedings of the Twenty-second European Conference on Artificial Intelligence; IOS Press; 2016. p. 1283–1290.

[126] Speichert S, Belle V. Learning probabilistic logic programs in continuous domains. In: ILP; 2019.

[127] Shapiro EY. An algorithm that infers theories from facts. In: IJCAI; Citeseer; 1981. p. 446–451.

[128] Gold EM. Language identification in the limit. Information and control. 1967;10(5):447–474.

[129] Belle V, Juba B. Implicitly learning to reason in first-order logic. In: Advances in Neural Information Processing Systems; 2019. p. 3376–3386.

[130] Campero A, Pareja A, Klinger T, et al. Logical rule induction and theory learning using neural theorem proving. arXiv preprint arXiv:180902193. 2018;.

[131] Marra G, Kuželka O. Neural markov logic networks. arXiv preprint arXiv:190513462. 2019;.

[132] Domingos P. The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books; 2015.

[133] Niu F, Zhang C, Ré C, et al. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. VLDS. 2012;12:25–28.

[134] Carlson A, Betteridge J, Kisiel B, et al. Toward an architecture for never-ending language learning. In: AAAI; 2010. p. 1306–1313.

[135] Cohen W, Yang F, Mazaitis KR. Tensorlog: A probabilistic database implemented using deep-learning infrastructure. Journal of Artificial Intelligence Research. 2020;67:285–325.

[136] Minervini P, Bošnjak M, Rocktäschel T, et al. Differentiable reasoning on large knowledge bases and natural language. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 34; 2020. p. 5182–5190.

[137] Liu Y, Levesque H. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In: Proc. IJCAI; 2005. p. 522–527.

[138] Levesque HJ, Brachman RJ. Expressiveness and tractability in knowledge representation and reasoning. Computational Intelligence. 1987;3:78–93.

[139] Mitchell DG, Ternovska E. A framework for representing and solving NP search problems. In: AAAI; 2005. p. 430–435.

[140] Koller D, Levy A, Pfeffer A. P-classic: a tractable probablistic description logic. In: Proc. AAAI / IAAI; 1997. p. 390–397.

[141] Bach FR, Jordan MI. Thin junction trees. In: Advances in Neural Information Processing Systems; 2002. p. 569–576.

[142] Darwiche A. A differential approach to inference in Bayesian networks. Journal of the ACM (JACM). 2003;50(3):280–305.

[143] Darwiche A. Three modern roles for logic in ai. In: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems; 2020. p. 229–243.

[144] Rudin C, Ustun B. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. Interfaces. 2018;48(5):449–466.

[145] Huang X, Kwiatkowska M, Wang S, et al. Safety verification of deep neural networks. In: International Conference on Computer Aided Verification; Springer; 2017. p. 3–29.

[146] Xu J, Zhang Z, Friedman T, et al. A semantic loss function for deep learning with symbolic knowledge. In: International Conference on Machine Learning; 2018. p. 5502–5511.

[147] Papantonis I, Belle V. Closed-form results for prior constraints in sum-product networks. Frontiers in Artificial Intelligence. 2021;.

[148] Farnadi G, Babaki B, Getoor L. Fairness in relational domains. In: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society; 2018. p. 108–114.

[149] Dries A, Kimmig A, Davis J, et al. Solving probability problems in natural language. In: IJCAI; 2017.

[150] Goodman ND, Mansinghka VK, Roy DM, et al. Church: A language for generative models. In: Proceedings of UAI; 2008. p. 220–229.

[151] De Raedt L, Kimmig A. Probabilistic (logic) programming concepts. Machine Learning. 2015; 100(1):5–47.

[152] Fierens D, Van den Broeck G, Thon I, et al. Inference in probabilistic logic programs using weighted CNF's. In: UAI; 2011. p. 211–220.

[153] Holtzen S, Van den Broeck G, Millstein T. Dice: Compiling discrete probabilistic programs for scalable inference. arXiv preprint arXiv:200509089. 2020;.

[154] Dos Martires PZ, Dries A, De Raedt L. Exact and approximate weighted model integration with probability density functions using knowledge compilation. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 33; 2019. p. 7825–7833.

[155] Albarghouthi A, D'Antoni L, Drews S, et al. Quantifying program bias. CoRR. 2017;abs/1702.05437.

[156] Belle V. Abstracting probabilistic models: Relations, constraints and beyond. Knowledge-Based Systems. 2020;:105976.

[157] Banihashemi B, De Giacomo G, Lespérance Y. Abstraction in situation calculus action theories. In: AAAI; 2017. p. 1048–1055.

[158] Holtzen S, Millstein T, Van den Broeck G. Probabilistic program abstractions. In: UAI; 2017.

[159] Gunning D. Explainable artificial intelligence (xai). DARPA/I20; 2016.

[160] Belle V, Papantonis I. Principles and practice of explainable machine learning. Frontiers in Big Data. 2021;.

[161] Sreedharan S, Srivastava S, Kambhampati S. Hierarchical expertise level modeling for user specific contrastive explanations. In: IJCAI; 2018. p. 4829–4836.

[162] Nitti D, Belle V, De Laet T, et al. Planning in hybrid relational mdps. Machine Learning. 2017; 106(12):1905–1932.

[163] Bundy A, Nuamah K, Lucas C. Automated reasoning in the age of the internet. In: International Conference on Artificial Intelligence and Symbolic Computation; Springer; 2018. p. 3–18.

[164] Belle V, Levesque HJ. Allegro: Belief-based programming in stochastic dynamical domains. In: IJCAI; 2015.

[165] Halpern JY. Reasoning about uncertainty. MIT Press; 2003.

[166] Ensan A, Ternovska E. Modular systems with preferences. In: IJCAI; 2015. p. 2940–2947.

[167] Lierler Y, Truszczynski M. An abstract view on modularity in knowledge representation. In: AAAI; 2015. p. 1532–1538.

[168] Bistarelli S, Montanari U, Rossi F. Semiring-based constraint logic programming: syntax and semantics. TOPLAS. 2001;23(1):1–29.

[169] Eisner J, Filardo NW. Dyna: Extending Datalog for modern AI. In: Datalog reloaded. (LNCS; Vol. 6702). Springer; 2011. p. 181–220.

[170] Kimmig A, Van den Broeck G, De Raedt L. Algebraic model counting. J Appl Log. 2017;22:46–62.

[171] Belle V, De Raedt L. Semiring programming: A declarative framework for generalized sum product problems. AAAI Workshop: Statistical Relational Artificial Intelligence. 2020;.

[172] Kordjamshidi P, Roth D, Kersting K. Systems ai: A declarative learning based programming perspective. In: IJCAI; 2018. p. 5464–5471.

[173] Le TA, Baydin AG, Wood F. Inference compilation and universal probabilistic programming. In: Artificial Intelligence and Statistics; PMLR; 2017. p. 1338–1348.

[174] Obermeyer F, Bingham E, Jankowiak M, et al. Functional tensors for probabilistic programming. arXiv preprint arXiv:191010775. 2019;.

[175] Riegel R, Gray A, Luus F, et al. Logical neural networks. arXiv preprint arXiv:200613155. 2020;.

[176] Levesque HJ. Common sense, the turing test, and the quest for real ai. MIT Press; 2017.

[177] Belle V. Symbolic logic meets machine learning: A brief survey in infinite domains. In: International Conference on Scalable Uncertainty Management; Springer; 2020. p. 3–16.