Cognitive Reasoning and Learning Mechanisms

Loizos Michael

Open University of Cyprus
`loizos@ouc.ac.cy`

Abstract. With an eye towards the development of systems for common everyday tasks — that operate in a manner that is cognitively-compatible with the argumentative nature of human reasoning — we present mechanisms for reasoning with and learning *cognitive programs*: common sense, symbolically represented in the form of prioritized association rules. The `FLASH` mechanism supports a fast argumentation-flavored type of reasoning, while sidestepping the rigidness of traditional proof procedures in formal logic. The `NERD` mechanism supports the never-ending acquisition of cognitive programs, without human supervision.

1 Introduction

The dawn of the era of cognitive systems and human-computer symbiosis replaces the heretofore prevalent metaphor of "a computer as a multi-purpose *tool*" with the much more apt metaphor of "a computer as a cognitive *assistant*". Humans no longer *use* their computing-enabled devices, but rather *collaborate* with them to solve certain cognitive tasks, with each collaborator being expected to learn, and adapt to, the way of thinking of the other. Importantly, communication and decision-making happens at a level that is transparent to, and cognitively-compatible with, human abilities and limitations.

The framework of *cognitive programming* [Michael *et al.*, 2015] was proposed to study such considerations in the context of everyday cognitive tasks, with symbolically-represented common sense capturing *phenomenologically* the reasoning from percepts to inferences. Unlike certain work in logic-based AI that effectively treats symbolically-represented knowledge as a set of (possibly defeasible) constraints to be satisfied, and approaches reasoning as a proof procedure towards that end, the framework emphasizes the need to acknowledge psychological evidence on the looser form of reasoning that is used for common sense [Evans, 2002], and embraces recent theories suggesting that human reasoning is at its foundation argumentation-driven [Mercier and Sperber, 2011].

The focus of cognitive programming is not on the development of general-purpose cognitive architectures [Anderson and Lebiere, 1998; Laird, 2012] or languages [Giarratano and Riley, 2004; Gelfond and Kahl, 2014] that are able to support decision-making and problem-solving in complex situations. Rather, the framework strives to examine how to best draw useful, and convincing to a human, inferences in everyday scenarios that would generally fall under the auspices of what Kahneman [2011] calls System 1, or the fast thinking system. This focus allows one to *shift the emphasis from deep and elaborated reasoning to richly-structured knowledge* [Michael *et al.*, 2015], where the cognitive assistant undertakes shallow reasoning, "jumps to conclusions" while "knowing when to think twice" [Bach, 1984], and relies on an interaction with a human collaborator, or some other mechanism that supports knowledge acquisition, to enrich the structure of its knowledge towards drawing appropriate inferences.

Indeed, the appropriateness or convincingness of the drawn inferences ultimately rests on the mechanism that acquires the knowledge that supports those inferences. The cognitive programming proposal suggests that a central mechanism of knowledge acquisition is an autodidactic [Michael, 2008] learning process, operating continuously in the background without human supervision. The proposal adopts natural language text as a knowledge source (see, e.g., [Mitchell *et al.*, 2015]), and argues that learned knowledge should capture the structure not of the surface text, but of the reality "between the lines" [Michael, 2009], which will have to be inferred when reasoning (cf. [Dagan *et al.*, 2013]). For our purposes, this argument points to a learning process that can cope with arbitrarily partial information, from which it seeks to approximate (to the extent possible) the structure of the underlying reality [Michael, 2010]. To ensure guarantees on the quality of the acquired knowledge, the proposal points to the *probably approximately correct* semantics [Valiant, 1984; 2000] for guidance, and to recent extensions that establish certain benefits for "simultaneous learning and prediction" [Michael, 2014].

Following these guidelines, we present in this work the `FLASH` and `NERD` mechanisms, which we have developed and implemented to reason with and learn cognitive programs. The remainder of this section discusses how these mechanisms fit into the big picture of related works in Artificial Intelligence. The sections that follow introduce the two mechanisms and their central components in more detail, making connections mostly with areas outside Artificial Intelligence to show how terminology and knowhow from those disciplines translates to concrete formal terms and computational processes.

1.1 Discussion on Reasoning

Reasoning with cognitive programs is tackled in this work through the introduction of the `FLASH` mechanism, a natural algorithm that iteratively applies implication rules on an externally provided percept to derive the inferences that follow. Each rule captures a defeasible regularity in the domain of interest, and the `FLASH` mechanism resolves conflicts that arise when applying these rules by having weaker rules yield to stronger ones, and all rules yield to externally provided evidence. Much like the Chase algorithm in database theory [Abiteboul *et al.*, 1995], each drawn inference is ultimately grounded on the given percept. Unlike the Chase algorithm, the inferences are not drawn monotonically, not only in the inter-percept sense that a certain percept may yield more inferences than another more complete one, but also in the intra-percept sense that while computing the inferences of a particular percept, the inferences that are drawn are provisional (as a result of jumping to conclusions), and may be retracted (as a result of thinking twice) as the computation progresses. Further, the representation and reasoning are such that no computationally hard satisfiability problems need to be solved to draw inferences. In particular, the bodies of rules are assumed to be read-once formulas, which can be evaluated on partial percepts efficiently; even only slightly more expressive classes of formulas are known to lead to intractable reasoning [Michael, 2010].

On some level, the `FLASH` mechanism performs a form of structured argumentation, like the one found, for instance, in the ASPIC+ framework [Prakken, 2010]. The `FLASH` mechanism deviates, however, from the typical argumentation requirement that an acceptable argument (which determines the drawn inferences) must defend all its attacks. Rather, the operational semantics of the `FLASH` mechanism considers only those rules that happen to be "ready to fire" at each iteration, and seeks to defend only those attacks that are thus generated. This more shallow type of reasoning avoids the typically intractable task of determining acceptability in formal argumentation. Unlike *syntactic* restrictions (e.g., on the rule sizes, or on the number of rules included in an argument) that one could impose in formal argumentation to reinstate tractability, our approach can be thought of as a pragmatic *semantic* restriction on the notion of acceptability.

Any deficiencies of this pragmatic restriction on reasoning are expected to be overcome with the expansion of a knowledge base with additional rules that would offer new arguments towards the expected inference. It is by acknowledging the central role of this feedback from a human to a cognitive assistant that the cognitive programming framework supports a shift from deep elaborated reasoning to richly-structured knowledge. We prove later that including new rules is always sufficient as an updating mechanism.

One could argue that the `FLASH` mechanism is subsumed by well-established cognitive architectures such as ACT-R [Anderson and Lebiere, 1998] and Soar [Laird, 2012], or that the mechanism can be easily implemented using declarative programming languages like CLIPS [Giarratano and Riley, 2004] and ASP [Gelfond and Kahl, 2014]. This point is uncontested, if only because these architectures and languages offer sufficient expressivity to implement any computable mechanism that would be of interested in our context. Despite sharing our motivation of cognitive-compatibility, and the underlying mechanism of a production system, these approaches are typically geared towards the development of complex expert systems, with the experts having at their disposal tools such as means-ends analysis and hill-climbing to search a problem space.

Our interpretation of cognitive-compatibility in this work explicitly seeks to *exclude such problem-solving types of reasoning* that would generally fall under the auspices of what Kahneman [2011] calls System 2, or the slow thinking system. One could adopt the view that the knowledge representation and the reasoning mechanism we propose aim to *isolate a fragment* of existing cognitive architectures or declarative programming languages, without committing to their extra features (which might be necessary for System 2 reasoning), so that the fragment would be understandable to a "person in the street", who would be able (perhaps through a natural language interface; cf. [Michael *et al.*, 2015]) to cognitively program their smart device, and be able to make sense of the arguments their device presents in support of, or against, a certain inference or action.

1.2 Discussion on Learning

Choosing an appropriate fragment of existing frameworks is not a trivial task, as it must balance the expressivity of representation, the tractability of reasoning, the cognitive-compatibility with ordinary humans, and ultimately the ability to offer guarantees on the learned knowledge. The `NERD` mechanism that we introduce has been designed with existing PAC learnability results in mind [Valiant, 1984; Kearns and Vazirani, 1994]. Although formally establishing that it, or any algorithm for that matter, is a PAC learner in our context of shallow reasoning with prioritized rules remains an open problem, evidence from the literature highlights that the balance we seek to strike is a fine one.

The NERD mechanism can be roughly thought to perform association rule mining [Agrawal *et al.*, 1993], seeking to identify a collection of rules, each with sufficiently high support (i.e., its body is often true) and confidence (i.e., its head is often true when its body is true). It diverges from this view in that it works in an online fashion, identifies priorities between rules, deals with arbitrarily missing information in its percepts, and learns rules that can be meaningfully reasoned with using the FLASH mechanism.

Its online nature is inspired by the Winnow algorithm [Littlestone, 1988] for learning linear thresholds in a noise-resilient fashion. The NERD mechanism associates a weight with each rule under consideration, which is promoted or demoted with each percept that, respectively, confirms or contradicts the rule. Only when (and as long as) a rule's weight exceeds a certain threshold is the rule considered as having been learned. Weights are updated multiplicatively, since this is known to support attribute-efficient learning [Littlestone, 1988]. The weight promoting / demoting scheme that we adopt differs from that of Winnow only in that it moves towards / away from a given upper bound, rather than moving away / towards a given initial value. The change aims only to help humans interpret a weight as a real-valued confidence level, without meaning that a weight formally corresponds to a probability or some other predetermined notion.

The NERD mechanism eventually seeks to produce a collection of rules, each a loose association of the form "*A offers evidence for B*". An individual rule is, by itself, not qualified by any explicit degree of confidence, and its qualification happens through its interaction with other rules. In particular, the weight associated with a rule is used only as a learning aid to capture succinctly the evidence that the NERD mechanism has found in favor of and against that rule, and to decide when this rule is sufficiently favored.

This distinguishes our approach from linear thresholds, and probabilistic logics such as Markov Logic Networks [Richardson and Domingos, 2006] and Bayesian Networks, where weights associated with formulas or atoms are part of the their intended representation and meaning, and are used explicitly by the reasoning mechanism. Interestingly, Conditional Preference Networks [Boutilier *et al.*, 2004], in a sense the "weightless" or discrete analogue of Bayesian Networks, can be viewed as a collection of prioritized implication rules, and their PAC learnability has been demonstrated formally [Dimopoulos *et al.*, 2009] and empirically [Michael and Papageorgiou, 2013]. We expect that line of work to prove useful in the investigation of PAC learnability in our context.

In terms of learning priorities, the NERD mechanism is inspired by the class of decision lists, which is known to be both learnable [Rivest, 1987] and evolvable [Michael, 2012]. Typically, learning decision lists proceeds by identifying the strongest condition first, or, roughly, the rule with the fewest exceptions. Accordingly, the NERD mechanism considers rules whose weight first reaches the threshold as being the strongest, and proceeds to assign to them a higher priority than subsequently learned rules. Previous attempts to combine multiplicative weight updating and the learnability of decision lists [Nevo and El-Yaniv, 2003] — although without dealing with the additional complications that reasoning brings into our work — show this approach to be tractable and to retain the attribute-efficiency of Winnow, when the depth of the exceptions is bounded.

We are unaware of learning algorithms that learn *partially-ordered* decision lists or rules under the PAC semantics. We later prove, in fact, that the learnability of partially-ordered rules relates to certain negative PAC learnability results under standard complexity assumptions. Such computational evidence suggests that the extra expressivity afforded by the partial ordering of rules, as supported by the FLASH mechanism, might go beyond what one should consider. It also suggests that the totally-ordered rules that are necessarily returned by the NERD mechanism might be the best one could hope for if one expects to eventually prove that the mechanism is a PAC learner. A recent study [Diakidoy *et al.*, 2015a] has identified corroborating psychological evidence, showing that human participants that were asked to produce and then curate rules in the context of story understanding [Diakidoy *et al.*, 2014; 2015b], have rarely invoked the use of priorities between rules. Other works that have investigated the learnability of default concepts [Schuurmans and Greiner, 1994] or exceptions in logic programming [Dimopoulos and Kakas, 1995] have placed less emphasis on the computational complexity, and have not dealt with the learnability of partially-ordered exceptions either.

Dealing with arbitrarily missing information (as needed to learn knowledge from text [Michael, 2009]) manifests itself in two distinct ways. First, the NERD mechanism learns without a predetermined set of atoms, in what is called an infinite-attribute domain. It is known that this lack of information makes learnability harder, but the task still remains possible and enjoys certain natural algorithms [Blum, 1992]. Second, even with a determined set of atoms, the partial percepts faced by the NERD mechanism are known to hinder learnability, with even decision lists being unlearnable under typical worst-case complexity assumptions [Michael, 2010; 2011b]. However, worst-case negative results are inapplicable in everyday scenarios where information is not hidden adversarially. Further, the shift from equivalences, such as decision lists, to prioritized implications further suggests that the non-learnability of the former need not carry over to the latter class. Read-once formulas, as those in the rule bodies, are known, in particular, to remain PAC learnable even in the absence of full information [Michael, 2010].

It is also known that learning from partial percepts is tightly coupled to reasoning, and that a learning mechanism must simultaneously learn and predict to ensure highly-complete inferences without compromising the soundness of the learned rules [Michael, 2014]. Efficiency considerations, then, impose restrictions on the length of the inference trace (number of reasoning steps), which, fortuitously, can be viewed in a positive light as being in line with psychological evidence on the restricted depth of human reasoning [Balota and Lorch, 1986]. Accordingly, the NERD mechanism incorporates inferences from the FLASH mechanism, which itself limits the number of its reasoning steps.

Overall, our chosen fragment seems to lie at the edge of what is / what is not (known to be) learnable. This realization can be viewed favorably in the context of developing cognitively-compatible mechanisms, since, arguably, evolutionary pressure would have pushed for such an optimal choice for the cognitive processing in humans as well.

Among works that seek to combine, and strike a balance between, reasoning and learning in a unified framework, most closely related to ours is work on Robust Logics [Valiant, 2000], which shares our emphasis on PAC learnability and tractable reasoning. The treatment of missing information in that work differs significantly from ours, in that missingness of information on an atom is assumed to be a distinct value that can be perceived and reasoned with (much like in the work of Schuurmans and Greiner [1994]). Further, learned rules are in the form of equivalences, which we have argued against as a choice of representation due to their diminished learnability from partial information.

Inductive Logic Programming [Muggleton, 1991] also tightly integrates reasoning and learning. In its basic form, ILP effectively seeks to identify patterns in a training set of data, and is less concerned with predictive guarantees and efficiency in the PAC sense. Further, as an extension of logic programming, it operates under the closed-world assumption, effectively dismissing the consideration of missing information in percepts, or the possibility of abstaining from drawing an inference on a certain atom. Certain extensions dealing with the learnability of priorities between rules (e.g., [Dimopoulos and Kakas, 1995]) seem to introduce negations in the heads of rules, but these extensions are ultimately translatable to the typical closed-world treatment of logic programming.

2 Knowledge Representation

An agent uses a pre-specified language to assign finite names to ***atoms***, which are used to represent concepts related to its environment. Atoms are not explicitly provided upfront, but are encountered across the agent's lifespan while it perceives its environment. From a neural perspective, an atom might be thought of as a set of neurons (i.e., the rules that support the atom to be inferred) assigned to represent a concept [Valiant, 2006].

A ***scene*** s is a mapping from atoms to the set $\{\texttt{0}, \texttt{1}, *\}$, with the values corresponding to *false*, *true*, *unspecified*. We write $s[\alpha]$ to mean the value associated with atom α, and call atom α ***specified*** in scene s if $s[\alpha] \in \{\texttt{0}, \texttt{1}\}$. Scenes s_1, s_2 ***agree*** on atom α if $s_1[\alpha] = s_2[\alpha]$. Scene s_1 is an ***expansion*** of scene s_2 if s_1, s_2 agree on every atom specified in s_2. Scene s_1 is a ***reduction*** of scene s_2 if s_2 is an expansion of s_1. A scene s is the ***greatest common reduct*** of a set S of scenes if s is the only scene among its expansions that is a reduction of each scene in S. A set S of scenes is ***compatible*** if there exists a particular scene that is an expansion of each scene in S.

In simple psychological terms, a scene can be thought of as representing the contents of an agent's working memory, where the agent's (possibly subjective) perception of the current environment state, and any relevant thereto drawn inferences, are made concrete for further processing. A scene corresponding to externally perceived contents of the working memory will be henceforth called a ***percept***. Following psychological evidence, the maximum number, denoted by $\texttt{w}$, of specified atoms in any scene (or percept) used by the agent can be assumed to be a small constant [Miller, 1956].

A propositional formula ψ is ***true*** (resp., ***false***) and ***specified*** in s if ψ (resp., $\neg\psi$) is classically entailed by the conjunction of: atoms α such that $s[\alpha] = \texttt{1}$, and the negation of atoms α such that $s[\alpha] = \texttt{0}$; otherwise, ψ is ***unspecified*** in s. When convenient, we represent unambiguously a scene as the set of its true literals (atoms or their negations).

A ***rule*** is an expression of the form $\varphi \rightsquigarrow \lambda$, where formula φ is the ***body*** of the rule, and literal λ is the ***head*** of the rule, with φ and λ not sharing any atoms, and with φ being read-once (no atom appears more than once). The intuitive reading of a rule is that when its body is true in a scene, an agent has *evidence* that its head should be true.

A collection of rules may happen to simultaneously offer evidence for conflicting conclusions. To resolve such conflicts, we let rules be qualified based on their priorities. A ***knowledge base*** $\kappa = \langle \varrho, \succ \rangle$ over a set $\mathcal{R}$ of rules comprises a finite collection $\varrho \subseteq \mathcal{R}$ of rules, and an irreflexive antisymmetric ***priority relation*** $\succ$ that is a subset of $\varrho \times \varrho$.

Although we do not make this always explicit, rules in ϱ are named (and copies of rules with different names may exist), and the priority relation $\succ$ is defined over their names.

For a knowledge base $\kappa = \langle \varrho, \succ \rangle$: its ***length*** is the number $\mathtt{l}$ of rules $r \in \varrho$; its ***breadth*** is the maximum number $\mathtt{b}$ of atoms in the body of a rule $r \in \varrho$; its ***depth*** is the maximum number $\mathtt{d}$ such that $r_0 \succ r_1, r_1 \succ r_2, \ldots, r_{\mathtt{d}-1} \succ r_{\mathtt{d}}$, for rules $r_i \in \varrho$.

3 The Reasoning Mechanism

We start by defining how rules are qualified by percepts or other rules, and then establish the operational semantics of reasoning by the repeated application of a step operator.

Definition 1 (Exogenous and Endogenous Qualifications). *Rule r_1 is **applicable** on scene s_i if r_1's body is true in s_i. Rule r_1 is **exogenously qualified** on scene s_i by percept s if r_1 is applicable on s_i and its head is false in s. Rules r_1, r_2 are **conflicting** if their heads are the negations of each other. Rule r_1 is **endogenously qualified** on scene s_i by rule r_2 if r_1, r_2 are applicable on s_i and conflicting, and $r_1 \not\succ r_2$.*

Definition 2 (Step Operator). *The **step operator** for a knowledge base κ and a percept s is a mapping $s_i \overset{\kappa,s}{\dashrightarrow} s_{i+1}$ from a scene s_i to the scene s_{i+1} that is an expansion of the percept s and differs from s only in making true the head of each rule r in κ that: (i) is applicable on s_i, (ii) is not endogenously qualified on s_i by a rule in κ, and (iii) is not exogenously qualified on s_i by s. A rule satisfying (i) and (ii) is called **dominant** in the step, while a dominant rule satisfying (iii) as well is called **effective** in the step.*

Intuitively: The truth-values of atoms specified in percept s remain as perceived, since they are not under dispute (although disputing the percepts can easily be simulated if needed). The truth-values of other atoms in s_i are updated to incorporate in s_{i+1} the inferences drawn by the effective rules, and to drop those inferences that are no longer supported; this is possible since s_{i+1} is an expansion of s, but not necessarily of s_i.

The inferences of a knowledge base on a percept are determined by the set of scenes that are reached, and cannot be escaped from, by repeatedly applying the step operator.

Definition 3 (Inference Trace and Inference Frontier). *The **inference trace** of a knowledge base κ on a percept s is the infinite sequence $\mathtt{trace}(\kappa, s) = s_0, s_1, s_2, \ldots$ of scenes, with $s_0 = s$ and $s_i \overset{\kappa,s}{\dashrightarrow} s_{i+1}$ for each integer $i \geq 0$. The **inference frontier** of a knowledge base κ on a percept s is the set $\mathtt{front}(\kappa, s)$ of all the scenes that appear in $\mathtt{trace}(\kappa, s)$ infinitely often (or, equivalently, appear in $\mathtt{trace}(\kappa, s)$ more than once).*

Example 1. Consider a knowledge base κ with the rules r_1 : *Penguin* $\rightsquigarrow \neg$*Flying*, r_2 : *Bird* $\rightsquigarrow$ *Flying*, r_3 : *Penguin* $\rightsquigarrow$ *Bird*, r_4 : *Feathers* $\rightsquigarrow$ *Bird*, r_5 : *Antarctica* $\wedge$ *Bird* $\wedge$ *Funny* $\rightsquigarrow$ *Penguin*, r_6 : *Flying* $\rightsquigarrow$ *Wings*, r_7 : $\neg$*Flying* $\rightsquigarrow \neg$*Feathers*, and the priority $r_1 \succ r_2$. If percept $s = \{$*Antarctica*, *Funny*, *Feathers*$\}$, then $\mathtt{trace}(\kappa, s) =$

$\{$*Antarctica*, *Funny*, *Feathers*$\}$,
$\{$*Antarctica*, *Funny*, *Feathers*, *Bird*$\}$,
$\{$*Antarctica*, *Funny*, *Feathers*, *Bird*, *Flying*, *Penguin*$\}$,
$\{$*Antarctica*, *Funny*, *Feathers*, *Bird*, $\neg$*Flying*, *Penguin*, *Wings*$\}$,

$\{Antarctica, Funny, Feathers, Bird, \neg Flying, Penguin\}$,
$\{Antarctica, Funny, Feathers, Bird, \neg Flying, Penguin\}, \ldots$

and $\texttt{front}(\kappa, s)$ is the singleton set whose unique member is the (infinitely repeated in $\texttt{trace}(\kappa, s)$) scene $\{Antarctica, Funny, Feathers, Bird, \neg Flying, Penguin\}$.

Observe the back and forth while computing $\texttt{trace}(\kappa, s)$. Initially *Bird* is inferred, giving rise to *Flying*, and then to *Wings*, effectively *jumping to this conclusion* given the evidence available that far. When *Penguin* is later inferred, it leads rule r_1 to oppose, and in fact override and negate, the inference *Flying* from rule r_2, effectively capturing the ability to *think twice*, when new evidence emerges. As a result of this overriding of *Flying*, inference *Wings* is no longer supported through rule r_6, and is also dropped, even though no other rule directly opposes it; cf. Figure 1. $\neg$*Feathers* is never inferred, despite its support through rule r_7, since it is directly opposed by percept s.

Thus, the inference trace captures *the evolving contents of an agent's working memory*, while the inference frontier captures the working memory's final (possibly fluctuating) contents. Since the inference frontier generally includes multiple scenes, one can define multiple natural notions for entailment, some of which we explore below.

Definition 4 (Entailment Notions). *A knowledge base κ applied on a percept s **entails** a formula ψ if ψ is: (N_1) true in some scene in $\texttt{front}(\kappa, s)$; ($N_2$) true in some scene in $\texttt{front}(\kappa, s)$ and not false in any of the other scenes in $\texttt{front}(\kappa, s)$; ($N_3$) true in every scene in $\texttt{front}(\kappa, s)$; ($N_4$) true in the greatest common reduct of $\texttt{front}(\kappa, s)$.*

Of particular interest is the subtle difference that exists between the entailment notions N_3 and N_4: under N_4 an entailed formula needs to be not only true in every scene in $\texttt{front}(\kappa, s)$, but true *for the same reason*. This excludes reasoning by case analysis, where an inference can follow if it does in each of a set of collectively exhaustive cases. When $\texttt{front}(\kappa, s) = \{\{\alpha\}, \{\beta\}\}$, for instance, the formula $\alpha \vee \beta$ is true in every scene in $\texttt{front}(\kappa, s)$ by case analysis, and is entailed under N_3, but not under N_4.

When the inference frontier comprises only a single scene, all aforementioned (and generally distinct) entailment notions coincide. It could be hypothesized that such "stable" inference frontiers (i.e., working memory contents) are typical in naturally represented domains. Empirical results in Section 5 give some credence to this hypothesis.

3.1 Fast and Loose Inferencing

We propose next an algorithm for reasoning with knowledge bases, as defined in the context of this work. Algorithm 1 receives as input a knowledge base κ, and a percept s, and proceeds to compute their inference frontier. The positive integer *effort* bounds the maximum number of steps in the inference trace to be computed before giving up.

During its first stage, the algorithm expands the inference trace, analogously to what presumably happens with neuron activations in the brain. Figure 1 illustrates this computation as the spreading of activation over a network. Each oval node corresponds to one memory cell. At each moment in time certain nodes are set to true or false, encoding the current contents of the working memory. The initial values of the nodes are *permanently* set by the percept s. Each rule / neuron checks concurrently to see whether

Algorithm 1 Fast and Loose Argumentation for Scene enHancement

```
1:  function FLASH ($\kappa, s$, effort)
2:      Set $i := 0$, set $s_i := s$, and set $\texttt{trace}(\kappa, s) := s_i$.
3:      repeat
            # Stage 1: Compute next step in the inference trace.
4:      Let $s_{i+1} := s$ be a set of Priority CRCW memories.
5:      For each rule $\varphi \rightsquigarrow \lambda$ in $\kappa$, do concurrently:
6:          Let $v := \texttt{1/0}$ if $\lambda = \alpha/\neg\alpha$, for the atom $\alpha$ in $\lambda$.
7:          Check if $\varphi$ is true in $s_i$, and $\alpha$ is unspecified in $s$.
8:          If the check passes, attempt to write $v$ to $s_{i+1}[\alpha]$.
9:      Set $i := i + 1$, and append $s_i$ to $\texttt{trace}(\kappa, s)$.
            # Stage 2: Check if the inference frontier is identified.
10:     Set $f := \min\{j \mid s_j \in \texttt{trace}(\kappa, s), s_j = s_i\}$.
11:     Set $\texttt{front}(\kappa, s) := \{s_j \mid s_j \in \texttt{trace}(\kappa, s), f \leq j < i\}$.
12:     until either $\texttt{front}(\kappa, s) \neq \emptyset$ or $i \geq$ effort.
13:     Return $\texttt{front}(\kappa, s)$.
14: end function
```

it is applicable on the current scene s_i, and if its head is not specified in the percept s (i.e., if its inference does not contradict the fixed values of the nodes). Each rule that passes the check fires, and attempts to write concurrently a value (corresponding to the polarity of its head) to the node whose atom is predicted by the rule. The memory cell of each node allows only the rule with the highest priority (according to the knowledge base) to set the node's value. This guarantees that only effective rules set the contents of the working memory, correctly computing a scene s_{i+1} such that $s_i \overset{\kappa,s}{\dashrightarrow} s_{i+1}$.

An architecture, as the one assumed above, that supports the concurrent writing in memory cells according to some specified priority among the writing processes is well-studied in the relevant literature, and is known as a Priority CRCW (Concurrent Read Concurrent Write) PRAM (Parallel Random-Access Machine) architecture [Cormen *et al.*, 2009]. We assume that in the case where no rule has the highest priority (and, hence, no rule is dominant), the contents of the respective memory cell remain unaffected.

Going from here to identifying the inference frontier requires checking for repeated scenes in the inference trace, which is done during the algorithm's second stage. If only "stable" inference frontiers are of interest, the check can be completed much faster, as it reduces to checking whether $s_{i-1} = s_i$, without traversing the entire inference trace.

In computing the inference frontier, the FLASH mechanism operates in a distinctively credulous manner. It jumps to inferences as long as there is sufficient evidence to do so, and no immediate / local reason to qualify the evidence. If reasons emerge later that oppose an inference drawn earlier, those are considered as they become available.

This fast and loose inferencing follows Bach [1984], who argues for approaching default reasoning as *"inference to the first unchallenged alternative"*. It is also remi-

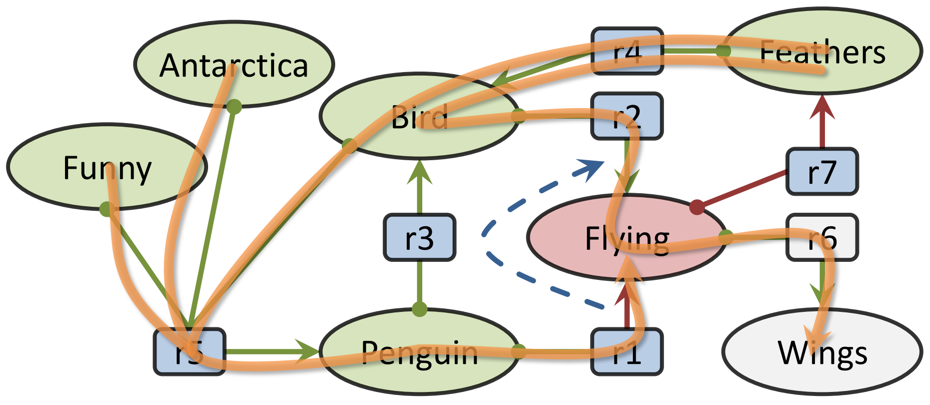

Fig. 1. Representation of the knowledge base and the single scene in the inference frontier from Example 1. Green, red, and grey ovals correspond to atoms that are true, false, and unspecified in the scene. Solid lines beginning with a circle connect an atom to a rule that includes that atom in its body, and solid lines ending with an arrowhead connect a rule to the single atom in the rule's head. Atoms that appear negated in the body or the head of a rule are connected to the rule via red lines, whereas the non-negated atoms are connected to the rule via green lines. Dashed blue lines between rules point to the rule with strictly less priority. Transparent orange traces from percepts to inferences show examples of arguments implicitly constructed by the `FLASH` algorithm.

niscent of the spreading-activation theory [Collins and Loftus, 1975], which can inform further extensions to make the framework and the mechanisms that we consider even more psychologically-valid (e.g., reducing the inference trace length by including a decreasing gradient in rule activations). For now, the restriction on the length of the inference traces is captured in Algorithm 1 through the inclusion of the parameter *effort*. Computational [Michael, 2014] and psychological [Balota and Lorch, 1986] evidence suggests, in fact, that this parameter takes necessarily only a small constant value.

Thinking about everyday situations where we often seem to draw inferences directly from the evidence in our percepts, one might even be tempted to argue that the value of *effort* is 1, in that we apply rules only once, without chaining them to draw further inferences. Harman [1974] objects to this view, and insists that we should not infer that intermediate steps do not occur when reasoning simply because we do not notice them, and that our inability to notice them might be due to the sheer speed with which we go through them. The efficiency of the `FLASH` mechanism certainly supports this scenario.

Theorem 1 (`FLASH` Analysis). *The `FLASH` mechanism returns* $\texttt{front}(\kappa, s)$ *assuming that the value of effort is sufficiently large, and returns* $\emptyset$ *otherwise. On a Priority CRCW PRAM architecture with priorities as specified by the knowledge base* κ*, `FLASH` runs in time at most quadratic in* $\texttt{w}$*, effort, and the breadth* $\texttt{b}$ *of* κ*.*

Proof (sketch). For memory priorities as those in κ correctness follows. For each of *effort* repetitions: Read-once formulas are evaluated on scenes in time linear in their respective sizes $\texttt{b}$ and $\texttt{w}$. Remaining steps take time linear in *effort* and $\texttt{w}$. □

Recalling the evidence suggesting that `w`, *effort*, `b` are all small constants[1], the `FLASH` mechanism runs effectively in constant time on a Priority CRCW PRAM architecture.

3.2 Flavor of Argumentation

Argumentation [Dung, 1995] has revealed itself as a powerful formalism, within which several AI problems have been investigated (see, e.g., [Bench-Capon and Dunne, 2007; Baroni *et al.*, 2011]). We show that the `FLASH` mechanism has an argumentation flavor.

We start by explaining how the reasoning mechanism can be viewed as implicitly constructing arguments and attacks among collections of rules that support inferences.

Definition 5 (Arguments). *An* ***argument*** *A for the literal λ given a knowledge base κ and a percept s is a subset-minimal set of explanation-conclusion pairs of the form $\langle e, c\rangle$ ordered such that: if e equals s, then c is a literal that is true in s; if e equals a rule r in κ, then c is the head of the rule, and the rule's body is classically entailed by the conjunction of conclusions in preceding pairs in A; c equals λ for the last pair in A.*

We consider below two natural notions for attacks.

Definition 6 (Attack Notions). *An argument A_1 for literal λ_1* ***attacks*** *an argument A_2 for literal λ_2 given a knowledge base κ and a percept s if there exist $\langle e_1, c_1\rangle \in A_1$ and $\langle e_2, c_2\rangle \in A_2$ such that $c_1 = \lambda_1$, $c_2 = \neg\lambda_1$, e_2 is a rule in κ, and either $e_1 = s$ or: (N_1) e_1 is a rule in κ and $e_2 \not\succ e_1$; (N_2) for every $\langle e, c\rangle \in A_1$ such that e is a rule in κ, it holds that $e_2 \not\succ e$.*

Definition 7 (Argumentation Framework). *The* ***argumentation framework*** *$\langle \mathbb{A}, \mathbb{R}\rangle$ associated with a knowledge base κ and a percept s comprises the set $\mathbb{A}$ of all arguments for any literal given κ and s, and the attacking relation $\mathbb{R} \subseteq \mathbb{A} \times \mathbb{A}$ such that $\langle A_1, A_2\rangle \in \mathbb{R}$ if A_1 attacks A_2 given κ and s.*

Example 2. Consider the knowledge base κ, and the percept s from Example 1. One can identify two arguments, depicted in Figure 1 as traces from percepts to inferences: $A_1 = \{\langle s, \mathit{Feathers}\rangle, \langle r_4, \mathit{Bird}\rangle, \langle r_2, \mathit{Flying}\rangle, \langle r_6, \mathit{Wings}\rangle\}$, and $A_2 = \{\langle s, \mathit{Antarctica}\rangle, \langle s, \mathit{Feathers}\rangle, \langle r_4, \mathit{Bird}\rangle, \langle s, \mathit{Funny}\rangle, \langle r_5, \mathit{Penguin}\rangle, \langle r_1, \neg\mathit{Flying}\rangle\}$, with A_2 attacking A_1 under either N_1 or N_2. Quite appropriately, the inference frontier $\texttt{front}(\kappa, s)$ does not include the literal *Wings* that is supported only by the defeated argument.

The back and forth during the computation of $\texttt{trace}(\kappa, s)$, as discussed in Example 1, captures the provisional drawing of inferences that are supported by simpler arguments, and their retraction *if and when* more complex counterarguments are later identified, *without proactively trying to defend the drawing of an inference*.

Indeed, reasoning ideally and explicitly from premises to conclusions is dismissed by Bach [1984], as not being a good cognitive policy. Rather, he stipulates that: *"When our reasoning to a conclusion is sufficiently complex, we do not survey the entire argument for validity. We go more or less step by step, and as we proceed, we assume that if*

[1] The bound `w` on the size of scenes in $\texttt{trace}(\kappa, s)$ can be ensured, for instance, through some process that keeps only their coherent subpart as determined by κ [Murphy and Medin, 1985].

each step follows from what precedes, nothing has gone wrong[.]". The FLASH mechanism makes concrete exactly this point of view. A more detailed comparison of our reasoning semantics to existing argumentation frameworks remains to be undertaken.

3.3 Revision through Structure

To illustrate how richly-structured knowledge can compensate for the lack of deep and elaborated reasoning, we show that the addition of new rules is always sufficient as a way to nullify the effect of any chosen part of a given knowledge base, if this happens to be desirable, without a "surgery" to the existing knowledge [McCarthy, 1998].

Two knowledge bases κ_1, κ_2 (whose rules are possibly defined over two different sets of atoms) are ***equivalent*** if for every percept s, $\texttt{front}(\kappa_1, s) = \texttt{front}(\kappa_2, s)$. If $\kappa_1 = \langle \varrho_1, \succ_1 \rangle$, $\kappa_2 = \langle \varrho_2, \succ_2 \rangle$, then we write $\kappa_1 \subseteq \kappa_2$ to mean $\varrho_1 \subseteq \varrho_2$ and $\succ_1 \subseteq \succ_2$.

Theorem 2 (Additive Elaboration Tolerance). *Consider two knowledge bases κ_0, κ_1. Then, there exists a knowledge base κ_2 such that $\kappa_1 \subseteq \kappa_2$ and κ_0, κ_2 are equivalent.*

Proof (sketch). Initially set $\kappa_2 := \kappa_1$. For each rule $r : \varphi \rightsquigarrow \lambda$ in κ_1, add to κ_2 the rule $f_1(r) : \varphi \rightsquigarrow \neg\lambda$ with a fresh name $f_1(r)$. For each rule $r : \varphi \rightsquigarrow \lambda$ in κ_0, add to κ_2 the rule $f_0(r) : \varphi \rightsquigarrow \lambda$ with a fresh name $f_0(r)$. Give priority to rule $f_0(r)$ over every other rule that appears in κ_2 because of κ_1. For every priority $r_i \succ_0 r_j$ in κ_0, add to κ_2 the priority $f_0(r_i) \succ_2 f_0(r_j)$. $\square$

The simplicity of the result is important, as ordinary humans are expected to engage in such revisions of knowledge bases. Roughly, when an unexpected inference is drawn by a cognitive assistant, a human offers feedback by simply considering the effective rules in the inference trace that gave rise to that inference, and overriding those that are found to be unconvincing, by including new more preferred rules. Over time, this interactive addition of rules and priorities increases the structure of the knowledge base.

4 The Learning Mechanism

We assume that the environment is at each moment in time in a ***state***, which cannot be directly accessed by agents. Rather, each agent has a stochastic ***perception process*** perc determining for each state a probability distribution over a compatible subset of scenes; these capture the possible ways in which the particular state can be perceived by that particular agent. The agent senses the underlying environment state t by obtaining a ***percept*** s drawn from $\texttt{perc}(t)$. Not assigning an *a priori* meaning to states obviates the need to commit to an objective representation of the environment, and accommodates cases where an agent's perception process determines not only what the agent does (or can possibly) perceive, but also the subjective interpretation of the underlying state.

An agent is initialized with some (possibly empty, externally programmed, or previously learned) knowledge base κ. Whenever a state t is perceived and mapped into a percept s, the agent reasons to decide what else holds in t that so happened not to be perceived in s, and updates κ to incorporate newly available information. The influx of states proceeds indefinitely, and new atoms may be encountered with the lapse of time, without the agent having foreknowledge of all atoms that will be eventually perceived.

The performance of the agent on a certain state depends on its ability to draw sound and complete inferences with respect to a certain set of atoms. Given a set P of atoms, the ***P-projection*** of a scene s is the scene $s|_P \triangleq \{\lambda \mid \lambda \in s$ and the atom of λ is in $P\}$; the ***P-projection*** of a set S of scenes is the set $S|_P \triangleq \{s|_P \mid s \in S\}$.

Definition 8 (Resoluteness). *Given a knowledge base κ, and a percept s drawn from* $\texttt{perc}(t)$*, κ is **P-resolute** on s if the P-projection of* $\texttt{front}(\kappa, s)$ *is a singleton set.*

Definition 9 (Completeness). *Given a knowledge base κ, and a percept s drawn from* $\texttt{perc}(t)$*, κ is **P-complete** on s if every $s_f \in$* $\texttt{front}(\kappa, s)$ *specifies every atom in P.*

Since the state t against which one wishes to make sound predictions is inaccessible, soundness is more subtly defined against all possible ways in which t can be perceived.

Definition 10 (Soundness). *Given a knowledge base κ, a percept s drawn from* $\texttt{perc}(t)$*, and the compatible subset S of scenes determined by* $\texttt{perc}(t)$*, κ is **P-sound** on s against t if for every $s_f \in$* $\texttt{front}(\kappa, s)$*, $(\{s_f\} \cup S)|_P$ is compatible.*

Effectively, S in Definition 10 captures the agent's subjective interpretation of the state t that underlies percept s. The seemingly ill-defined requirement to be sound against the *unknown* compatible subset S can be achieved optimally in a defined sense by (and only by) ensuring that the drawn inferences are consistent with the percept s itself [Michael, 2010, Theorem 2.2]; or, that the rules used by the reasoning process are not exogenously qualified. Establishing such a relation between soundness and consistency in our context can proceed analogously to the proof of the cited result. We leave this for future work, and restrict ourselves to making precise the notion of consistency.

Definition 11 (Consistency). *Given a knowledge base κ, and a percept s drawn from* $\texttt{perc}(t)$*, κ is **P-consistent** on s if for every $s_f \in$* $\texttt{front}(\kappa, s)$*, every dominant rule in the step $s_f \overset{\kappa;s}{\dashrightarrow} s_{f+1}$ whose head atom is in P is effective (i.e., not exogenously qualified).*

Note that although the reasoning process can cope with exogenous qualification, this ability should be used in response to unexpected / exceptional circumstances, and only as a last resort. It is the role of the learning process *to minimize the occurrences of exogenous qualifications, and to turn them into endogenous qualifications*, through which the agent internally can explain why a certain rule failed to draw an inference.

Interestingly, the position above echoes evidence from the behavioral and brain sciences, asserting that the human brain is ultimately a predictive machine that learns (and even acts) in a manner that will minimize surprisal in its percepts [Clark, 2013]. Our analysis reveals that surprisal minimization is not necessarily an *end* in itself and a goal of the learning process, but rather a *means* to the reliability of the reasoning process.

4.1 Never-Ending Rule Discovery

We propose next an algorithm for learning knowledge bases. Algorithm 2 receives as input a knowledge base κ, which it proceeds to update according to a received percept

Algorithm 2 Never-Ending Rule Discovery

```
1: function NERD ($\kappa, s, s_t, F, \texttt{b}, \texttt{d}, \textit{change}, \textit{max}, \textit{intolerance}$)
2:     Set $\textit{rise} := \textit{change}^{1/\textit{intolerance}}$, and set $\textit{fall} := 1/\textit{change}$.
       # Stage 1: Addition of new rules for coverage.
3:     Set $\textit{eff} :=$ rules in $\kappa$ that are effective on $s_t$ given $s$.
4:     Set $\textit{add} := \texttt{CoverAtoms}(s_t, s|_F \setminus \texttt{Heads}(\textit{eff}), \texttt{b})$.
5:     Add to $\kappa$ the rules in add that are structurally novel.
       # Stage 2: Weight promotion for completeness.
6:     Set $\textit{app} :=$ rules in $\kappa$ that are applicable on $s_t$.
7:     Set $\textit{pro} :=$ rules in app that concur with $s|_F$.
8:     Set $\textit{act-neg} :=$ rules in pro that are not active.
9:     Adjust in $\kappa$ the weight of rules in pro by $(\textit{rise}, \textit{max})$.
       # Stage 3: Weak priority for newly active rules.
10:    Set $\textit{wkr} :=$ rules in $\kappa$ that are active and in act-neg.
11:    Make in $\kappa$ the rules in wkr $\texttt{d}$-weaker than active rules.
       # Stage 4: Weight fast demotion for consistency.
12:    Set $\textit{eff} :=$ rules in $\kappa$ that are effective on $s_t$ given $s$.
13:    Set $\textit{app} :=$ rules in $\kappa$ that are applicable on $s_t$.
14:    Set $\textit{dem} :=$ rules in app that oppose $s|_F \setminus \texttt{Heads}(\textit{eff})$.
15:    Set $\textit{act-pos} :=$ rules in dem that are active.
16:    Adjust in $\kappa$ the weight of rules in dem by $(\textit{fall}, \textit{max})$.
       # Stage 5: No priority for newly inactive rules.
17:    Set $\textit{ntr} :=$ rules in $\kappa$ that are not active and in act-pos.
18:    Remove from $\kappa$ the priorities of rules in ntr.
19:    Return $\kappa$.
20: end function
```

s. Along with κ, s, the algorithm receives some scene $s_t \in \texttt{trace}(\kappa, s)$. Inputs $F, \texttt{b}, \texttt{d}$ restrict the ***focus*** (atoms in rule heads), breadth, and depth of the knowledge base to be returned, while inputs *change*, *max*, *intolerance* specify how rule weights are updated.

In the context of Algorithm 2, we make allowances for the following: Each rule is associated with a ***weight***, a value from 0 to $\textit{max} > 1$. Rules with a weight of at least 1 are ***active***, and for the purposes of reasoning only the active rules are considered. Set membership is determined by a rule's name. Two rules are ***structurally equivalent*** if they have the same body and the same head, even if their names and / or weights differ.

Algorithm 2 proceeds through five stages: Initially, it covers new atoms by including rules (with at most $\texttt{b}$ body atoms) that are applicable on s_t and whose heads belong in $s|_F$ and are not inferred by effective rules. To improve completeness, the algorithm promotes (slowly, by *rise*) all applicable rules that concur with $s|_F$, nudging them towards activation. Rules that *become* active as a result of weight promotion are made weaker than existing active rules with which they are conflicting; such rules are discarded, however, if their addition would cause the depth of the knowledge base to increase beyond $\mathtt{d}$. To improve consistency, the algorithm demotes (quickly, by *fall*) all applicable rules that oppose $s|_F$; however, demotion is foregone for rules that oppose literals inferred by effective rules, since the latter rules explain the literals and qualify the opposing rules. Rules that *become* inactive as a result of weight demotion are made priority-neutral, eliminating the event of "carry-over" priorities in a possible subsequent re-activation.

The process `CoverAtoms` aims to introduce new rules in the pool of rules that the `NERD` mechanism maintains and seeks evidence for. The process can be defined in a number of ways without essentially affecting our analysis. For concreteness, however, we shall assume it returns a single rule $\varphi \rightsquigarrow \lambda$ (i.e., *add* is a singleton) for a conjunction formula φ such that: λ is chosen uniformly at random from $s|_F \setminus \mathtt{Heads}(\textit{eff})$; an integer number is chosen uniformly at random between 0 and $\min(\mathtt{b}, |s_t \setminus \{\lambda\}|)$, and those many literals are then chosen uniformly at random without replacement from $s_t \setminus \{\lambda\}$ to be the conjuncts of φ; the rule's initial weight is set to zero. In essence, the process `CoverAtoms` introduces a randomly chosen rule that is supported by the current scene s_t, and whose inference (had the rule been active) would explain the value of an atom in the percept $s|_F$ that is not currently explained / inferred by another effective rule.

The result of ***adjusting*** the current weight *weight* of a rule by $(\textit{value}, \textit{max})$ is defined to be $\textit{max} - (\textit{max} - \textit{weight})/\textit{value}$ if the result is positive, or zero otherwise. The adjustment operates on the distance of the weight from its maximum possible value of *max*. Thus: if $\textit{value} > 1$ then the distance is decreased by a multiplicative factor, moving the weight towards *max*; if $\textit{value} < 1$ then the distance is increased by a multiplicative factor, moving the weight towards zero, but never beyond that. The parameter *intolerance* effectively balances the desire for high completeness (i.e., the size of a promotion step) against the intolerance to high inconsistency (i.e., the size of a demotion step).

Rules are promoted and demoted irrespectively of whether they are active. An active rule is still strengthened if it draws a correct inference, even though it is already strong enough to do so. Analogously, an inactive rule is still penalized if it would have drawn an incorrect inference had it been active, even though it does not draw that inference.

Priorities exist only between active rules. However, both active and inactive rules are "protected" from demotion when an effective rule explains the percept. The former are necessarily weaker from the effective rule, whereas the latter would be weaker had they become active after the effective rule. Figure 2 illustrates a typical scenario.

Theorem 3 (`NERD` Analysis). *The `NERD` mechanism runs in time polynomial in* $\mathtt{b}, c^{\mathtt{d}}$, *and the sizes of* κ, s, s_t, F, *for some constant* c.

Proof (sketch). Usual mathematical operations take unit time. Step 4 avoids enumerating candidate rule bodies. Computing κ's depth at Step 11 can be shown to be NP-hard, but it is also fixed-parameter tractable [Chen *et al.*, 2007] for parameter $\mathtt{d}$. □

With the availability of an underlying architecture analogous to the one discussed in the context of the `FLASH` mechanism, the dependence of the running time on the size of κ can be eliminated, since rules can be checked and updated concurrently.

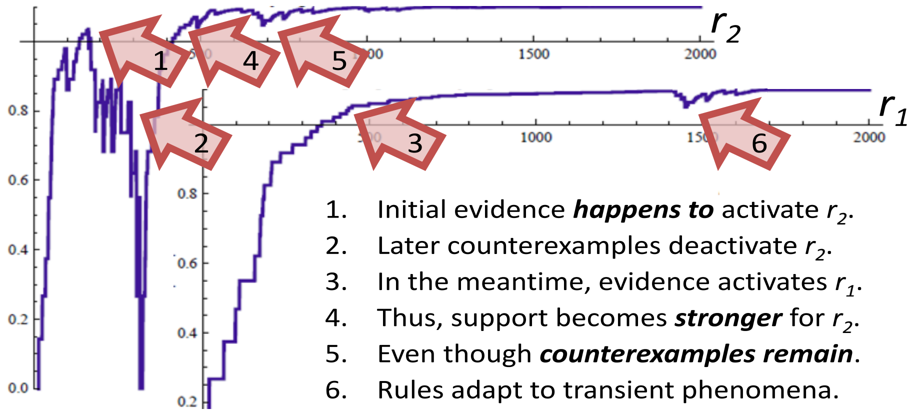

Fig. 2. Weight updating for rules r_1, r_2 from Example 1 with successive calls of the `NERD` mechanism. Jumps show promotions and demotions. Once rule r_1 becomes active, rule r_2 is "protected" from demotion whenever encountering a percept for a bird that is not flying, as long as the percept also specifies that the bird is a penguin. The never-ending nature of the `NERD` mechanism ensures the adaptation of rules to transient phenomena (e.g., a period with percepts of flying penguins).

4.2 Boundaries of Learnability

In addition to evidence from the literature, including from Computational Learning Theory and Cognitive Psychology, that we have provided in support of various choices we have made, we prove below formal results that offer additional corroborating evidence.

First, we recall that PAC learning polynomial-sized DNF formulas (even with complete information in scenes) is a major open problem in Computational Learning Theory [Kearns and Vazirani, 1994]. Our next result shows that an extremely bounded-depth knowledge base of prioritized implication rules can express any such DNF formula, suggesting that the learnability of the former is as hard as the learnability of the latter.

Theorem 4 (Expressing DNF Formulas with Knowledge Bases). *For any DNF formula u over $\mathtt{b}$ variables, and of size polynomial in $\mathtt{b}$, there exists a set A of $\mathtt{b}+1$ atoms (one atom for each variable, and an extra atom α) and a knowledge base κ over A, and of depth 1, breadth at most $\mathtt{b}$, and length polynomial in $\mathtt{b}$ such that: on any scene s that specifies all atoms / variables except α, the value of u on s is true / false if and only if κ applied on s entails α / $\neg\alpha$. The result holds for all entailment notions in Definition 4.*

Proof (sketch). Given a formula u, construct a knowledge base κ as follows: κ includes the rule $r_0 : \top \rightsquigarrow \neg\alpha$; for each disjunct φ in u, κ includes the rule $r : \varphi \rightsquigarrow \alpha$ and the priority $r \succ r_0$. Clearly, all entailment notions in Definition 4 coincide. □

The situation is roughly analogous when considering knowledge base breadth. We recall that monotone-term 1-Decision Lists are not PAC learnable under typical complexity assumptions, if scenes are (even very lightly) partial, and one insists on proper learning (i.e., that the learner outputs a monotone-term 1-Decision List as its learned hypothesis) [Michael, 2010]. Our next result shows that an extremely bounded-breadth knowledge base of prioritized implication rules can express any such Decision List, suggesting that the learnability of the former is as hard as the learnability of the latter.

Theorem 5 (Expressing Decision Lists with Knowledge Bases). *For any monotone-term 1-Decision List u over $\mathtt{d}$ variables, and of size at most $\mathtt{d}$, there exists a set A of $2\mathtt{d}+1$ atoms (two atoms for each variable, and an extra atom α) and a knowledge base κ over A, and of depth at most $\mathtt{d}$, breadth 1, and length polynomial in $\mathtt{d}$ such that: on any scene s that specifies any atoms / variables except the last $\mathtt{d}+1$ ones, the value of u on s is true / false / abstention if and only if κ applied on s entails α / $\neg\alpha$ / neither α nor $\neg\alpha$. The result holds for all entailment notions in Definition 4.*

Proof (sketch). Given a decision list u, construct a knowledge base κ as follows: for every condition c_i with a corresponding decision d_i in u, κ includes the rules $r_{i,\top} : \top \rightsquigarrow \alpha(d_i)$ and $r_i : \neg c_i \rightsquigarrow \neg\alpha(d_i)$; further, for every condition c_j preceding condition c_i in u, κ includes the rule $r_{j,i} : c_j \rightsquigarrow \neg\alpha(d_i)$; for each decision d_i in u, κ includes the rule $r_{i,\alpha} : \alpha(d_i) \rightsquigarrow \alpha/\neg\alpha$ depending on whether d_i is true / false. $\square$

Computational considerations suggest, then, that knowledge bases should be simultaneously of small breadth and of small depth (and, therefore, of small overall length or size), which would then ensure the efficiency of the `NERD` mechanism (cf. Theorem 3).

Note that Theorem 5 relies critically on the expressivity of the knowledge bases to represent rules without specifying a priority between them, so that when two rules are in conflict, neither takes precedence and they both end up abstaining. We have commented earlier on how this ability gives extra expressivity beyond what can be handled by the `NERD` mechanism, which necessarily returns totally-ordered rules. These observations are consistent with our eventual goal to show that the `NERD` mechanism is a PAC learner.

5 Empirical Demonstration

In what follows we offer yet another piece of evidence that our developed mechanisms work as expected. It is beyond the aims of this work to carry out a full empirical evaluation. We focus on demonstrating through a simple and understandable example (that humans can work through) the operation of the mechanisms. We do not discuss running times other than to say that about 2 seconds (on a Lenovo X201 Tablet with Intel Core i7) sufficed for processing 2000 scenes for learning, including the time needed to reason with these scenes, and the time to print status information after each processed scene.

The knowledge base κ^* in Figure 3 captures certain commonsensical rules of inference in relation to meningitis, its two most common types, and their respective treatability prospects. Observe that κ^* is P-resolute on each percept s from those considered below, for any choice of P. The `FLASH` mechanism correctly computes the unique scene in $\mathtt{front}(\kappa^*, s)$, i.e., $\{\neg \mathit{vm}, \neg \mathit{bm}, \neg m\}$ when $s = \{\}$; $\{\mathit{vm}, \neg \mathit{bm}, m, \neg f, t, \mathit{ws}\}$ when $s = \{\mathit{vm}\}$; $\{\neg \mathit{vm}, \mathit{bm}, m, f, \neg t, \neg \mathit{ws}\}$ when $s = \{\mathit{bm}\}$; and $\{\neg \mathit{vm}, \mathit{bm}, m, f, \neg t, \mathit{it}, \mathit{ws}\}$ when $s = \{\mathit{bm}, \mathit{it}\}$, properly integrating the unexpected input *it* and its ramification *ws*. For an empty percept s, inferences are drawn from rules with a tautology as their body.

For the perception process, we fix six states corresponding to the fatal / non-fatal / treated fatal cases of meningitis / non-meningitis, choose one at random, and partly

`true implies -viral_meningitis.`	($\top \rightsquigarrow \neg \mathit{vm}$)
`true implies -bacterial_meningitis.`	($\top \rightsquigarrow \neg \mathit{bm}$)
`viral_meningitis implies meningitis.`	($\mathit{vm} \rightsquigarrow \mathit{m}$)
`bacterial_meningitis implies meningitis.`	($\mathit{bm} \rightsquigarrow \mathit{m}$)
`true implies -meningitis.`	($\top \rightsquigarrow \neg \mathit{m}$)
`bacterial_meningitis implies fatal.`	($\mathit{bm} \rightsquigarrow \mathit{f}$)
`meningitis implies -fatal.`	($\mathit{m} \rightsquigarrow \neg \mathit{f}$)
`fatal implies -treatable.`	($\mathit{f} \rightsquigarrow \neg \mathit{t}$)
`meningitis implies treatable.`	($\mathit{m} \rightsquigarrow \mathit{t}$)
`is_treated implies will_survive.`	($\mathit{it} \rightsquigarrow \mathit{ws}$)
`fatal implies -will_survive.`	($\mathit{f} \rightsquigarrow \neg \mathit{ws}$)
`-fatal implies will_survive.`	($\neg \mathit{f} \rightsquigarrow \mathit{ws}$)

Fig. 3. Representation of the "meningitis" domain, with rules listed earlier having higher priority than subsequently-listed ones. The representation is a simplified version of the actual syntax (with named rules and partially-ordered priorities) supported by the implemented `FLASH` mechanism.

obscure it to produce a percept s with at most three randomly chosen specified atoms. Such percepts are sequentially given to the `NERD` mechanism, with an initially empty knowledge base κ, scene s_t chosen on each call at random from the inference frontier $\texttt{front}(\kappa, s)$, and values for the other parameters as follows: $F = \{\mathit{vm}, \mathit{bm}, \mathit{m}, \mathit{f}, \mathit{t}, \mathit{ws}\}$, $\texttt{b} \in \{1, 2\}$, $\texttt{d} = 2$, $\mathit{change} \in \{2, 3, 4\}$, $\mathit{max} = 1.1$, $\mathit{intolerance} \in \{1, 3, 5, 7, 9\}$.

Figure 4 shows how different parameters trade off soundness for completeness: in the second and third rows the learned knowledge base κ is, respectively, more complete and more sound than the hand-coded knowledge base κ^*; the knowledge bases are also overwhelmingly resolute. Such outcomes are possible because learning "picks up" rules not in κ^* that are, nonetheless, supported by the data: the contrapositive $\neg \mathit{f} \rightsquigarrow \neg \mathit{bm}$, the shortcut $\mathit{bm} \rightsquigarrow \neg \mathit{t}$, the cautionary $\mathit{m} \wedge \neg \mathit{bm} \rightsquigarrow \neg \mathit{f}$, or the coincidental $\mathit{bm} \rightsquigarrow \neg \mathit{vm}$.

Additional experiments, with varying domains and parameters, have been carried out, which, unfortunately, cannot be presented herein. The `NERD` mechanism is implemented in `C`, and exploits advanced data structures to boost computational efficiency.

6 Conclusions and Outlook

We have presented cognitive mechanisms for reasoning and learning, which could be utilized for the development of cognitive assistants in the context of the cognitive programming framework [Michael *et al.*, 2015]. Extensions of the mechanisms to represent a simple form of time may facilitate the handling of temporal narratives, in addition to atemporal scenes. Work on reasoning about actions and change [Kakas *et al.*, 2011], and recent approaches that use argumentation for narrative comprehension [Michael, 2013b; Diakidoy *et al.*, 2014] can guide such extensions. Theory-wise, the main next step is to establish the PAC nature of the proposed `NERD` mechanism and its potential extension to the problem of learning causal, in addition to atemporal, rules [Michael, 2011a].

In terms of knowledge acquisition, and complementing existing large-scale efforts in this direction [Mitchell *et al.*, 2015], the proposed (and extended) mechanisms can be

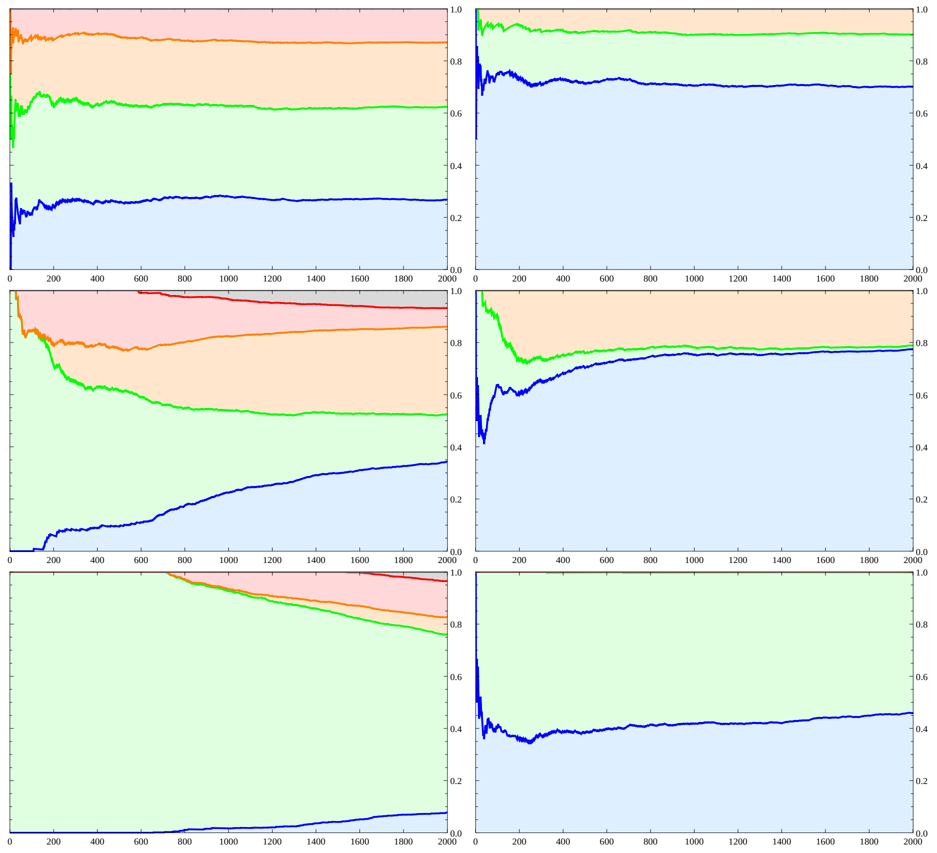

Fig. 4. Evaluation of the `NERD` mechanism on the "meningitis" domain. The (up to) five colored areas in each plot (stacked along the y-axis) show the percentage of all states / percepts seen that far (in the order determined along the x-axis), on which the learned knowledge base κ is (denoted `+`) or is not (denoted `-`) P-resolute (denoted `r`), P-sound (denoted `s`), or P-complete (denoted `c`) for the following combinations of these metrics: `+r+s+c` is represented by the blue area, `+r+s-c` is represented by the green area, `+r-s+c` is represented by the orange area, `+r-s-c` is represented by the red area, and `-r` is represented by the grey area. The two columns of plots correspond, left-to-right, to: $P = \{vm, bm, m, f, t, ws\}$, $P = \{ws\}$. The three rows of plots correspond, top-to-bottom, to: evaluating κ^* instead of κ, $\mathtt{b} = 2$ and low intolerance, $\mathtt{b} = 1$ and high intolerance.

applied on textual story corpora to extract phenomenological knowledge, after upgrading the propositional representation to a relational one via standard reductions [Valiant, 2000], as demonstrated in earlier work [Michael and Valiant, 2008; Michael, 2008; 2013a]. This application may also offer an experimental arena to compare our mechanisms to ones for association rule mining [Agrawal *et al.*, 1993] and ILP [Muggleton, 1991], in terms of how efficiently, reliably, and comprehensively knowledge is acquired.

On the psychological front, an interesting question is whether it can be shown that phenomena like Hebbian learning or Pavlovian conditioning can be simulated. Ongoing psychological work [Diakidoy *et al.*, 2015a] already examines empirically how humans activate knowledge in the form of rules, and the structure of the knowledge base they offer in terms of its breath and depth (and hence the use of priorities). We expect additional empirical studies to be informed from this work, and inform its future development.

References

[Abiteboul *et al.*, 1995] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing, Boston, MA, U.S.A., 1st edition, 1995.

[Agrawal *et al.*, 1993] R. Agrawal, T. Imieliński, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, DC, U.S.A., 1993.

[Anderson and Lebiere, 1998] J. R. Anderson and C. Lebiere. *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Mahwah, NJ, U.S.A., 1998.

[Bach, 1984] K. Bach. Default Reasoning: Jumping to Conclusions and Knowing When to Think Twice. *Pacific Philosophical Quarterly*, 65(1):37–58, 1984.

[Balota and Lorch, 1986] D. A. Balota and R. F. Lorch. Depth of Automatic Spreading Activation: Mediated Priming Effects in Pronunciation but Not in Lexical Decision. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 12(3):336–345, 1986.

[Baroni *et al.*, 2011] P. Baroni, M. Caminada, and M. Giacomin. An Introduction to Argumentation Semantics. *Knowledge Engineering Review*, 26(4):365–410, 2011.

[Bench-Capon and Dunne, 2007] T. J. M. Bench-Capon and P. E. Dunne. Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10–15):619–641, 2007.

[Blum, 1992] A. Blum. Learning Boolean Functions in an Infinite Attribute Space. *Machine Learning*, 9(4):373–386, 1992.

[Boutilier *et al.*, 2004] C. Boutilier, R. Brafman, C. Domshlak, H. Hoos, and D. Poole. CP-Nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research*, 21(1):135–191, 2004.

[Chen *et al.*, 2007] J. Chen, S. Lu, S.-H. Sze, and F. Zhang. Improved Algorithms for Path, Matching, and Packing Problems. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 298–307, New Orleans, LA, U.S.A., 2007.

[Clark, 2013] A. Clark. Whatever Next? Predictive Brains, Situated Agents, and the Future of Cognitive Science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[Collins and Loftus, 1975] A. M. Collins and E. F. Loftus. A Spreading-Activation Theory of Semantic Processing. *Psychological Review*, 82(6):407–428, 1975.

[Cormen *et al.*, 2009] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, U.S.A., 3rd edition, 2009.

[Dagan *et al.*, 2013] I. Dagan, D. Roth, M. Sammons, and F. M. Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2013.

[Diakidoy *et al.*, 2014] I.-A. Diakidoy, A. Kakas, L. Michael, and R. Miller. Story Comprehension through Argumentation. In *Proceedings of the 5th International Conference on Computational Models of Argument*, pages 31–42, Pitlochry, Scotland, U.K., 2014.

[Diakidoy *et al.*, 2015a] I.-A. Diakidoy, A. Kakas, and L. Michael. Knowledge Structure and Activation in Story Comprehension. In *Symposium on Bridging Cognitive Psychology and Artificial Intelligence, 19th Conference of the ESCoP*, Paphos, Cyprus, 2015.

[Diakidoy *et al.*, 2015b] I.-A. Diakidoy, A. Kakas, L. Michael, and R. Miller. STAR: A System of Argumentation for Story Comprehension and Beyond. In *Proceedings of the 12th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 64–70, Palo Alto, CA, U.S.A., 2015.

[Dimopoulos and Kakas, 1995] Y. Dimopoulos and A. Kakas. Learning Non-Monotonic Logic Programs: Learning Exceptions. In *Proceedings of the 8th European Conference on Machine Learning*, pages 122–137, Heraclion, Crete, Greece, 1995.

[Dimopoulos *et al.*, 2009] Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris Paribus Preference Elicitation with Predictive Guarantees. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1890–1895, Pasadena, CA, U.S.A., 2009.

[Dung, 1995] P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games. *Artificial Intelligence*, 77(2):321–357, 1995.

[Evans, 2002] J. S. Evans. Logic and Human Reasoning: An Assessment of the Deduction Paradigm. *Psychological Bulletin*, 128(6):978–996, 2002.

[Gelfond and Kahl, 2014] M. Gelfond and Y. Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, New York, NY, U.S.A., 1st edition, 2014.

[Giarratano and Riley, 2004] J. C. Giarratano and G. D. Riley. *Expert Systems: Principles and Programming*. Course Technology, 4th edition, 2004.

[Harman, 1974] G. Harman. *Thought*. Princeton University Press, Princeton, NJ, U.S.A., 1974.

[Kahneman, 2011] D. Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, New York, NY, U.S.A., 2011.

[Kakas *et al.*, 2011] A. Kakas, L. Michael, and R. Miller. Modular-E and the Role of Elaboration Tolerance in Solving the Qualification Problem. *Artificial Intelligence*, 175(1):49–78, 2011. Special Issue on 'Logical Approaches to Artificial Intelligence: Papers in Honor of John McCarthy'.

[Kearns and Vazirani, 1994] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, Cambridge, MA, U.S.A., 1994.

[Laird, 2012] J. E. Laird. *The Soar Cognitive Architecture*. The MIT Press, Cambridge, MA, U.S.A., 2012.

[Littlestone, 1988] N. Littlestone. Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm. *Machine Learning*, 2(4):285–318, 1988.

[McCarthy, 1998] J. McCarthy. Elaboration Tolerance. In *Proceedings of the 4th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 198–216, London, England, U.K., 1998.

[Mercier and Sperber, 2011] H. Mercier and D. Sperber. Why Do Humans Reason? Arguments for an Argumentative Theory. *Behavioral and Brain Sciences*, 34(02):57–74, 2011.

[Michael and Papageorgiou, 2013] L. Michael and E. Papageorgiou. An Empirical Investigation of Ceteris Paribus Learnability. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1537–1543, Beijing, P. R. China, 2013.

[Michael and Valiant, 2008] L. Michael and L. G. Valiant. A First Experimental Demonstration of Massive Knowledge Infusion. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning*, pages 378–389, Sydney, Australia, 2008.

[Michael *et al.*, 2015] L. Michael, A. Kakas, R. Miller, and G. Turán. Cognitive Programming. In *Proceedings of the 3rd International Workshop on Artificial Intelligence and Cognition*, pages 3–18, Turin, Italy, 2015.

[Michael, 2008] L. Michael. *Autodidactic Learning and Reasoning*. Doctoral Dissertation, Harvard University, Cambridge, MA, U.S.A., 2008.

[Michael, 2009] L. Michael. Reading Between the Lines. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1525–1530, Pasadena, CA, U.S.A., 2009.

[Michael, 2010] L. Michael. Partial Observability and Learnability. *Artificial Intelligence*, 174(11):639–669, 2010.

[Michael, 2011a] L. Michael. Causal Learnability. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1014–1020, Barcelona, Catalonia, Spain, 2011.

[Michael, 2011b] L. Michael. Missing Information Impedements to Learnability. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 825–827, Budapest, Hungary, 2011.

[Michael, 2012] L. Michael. Evolvability via the Fourier Transform. *Theoretical Computer Science*, 462:88–98, 2012.

[Michael, 2013a] L. Michael. Machines with WebSense. In *Proceedings of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning*, Ayia Napa, Cyprus, 2013.

[Michael, 2013b] L. Michael. Story Understanding... Calculemus! In *Proceedings of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning*, Ayia Napa, Cyprus, 2013.

[Michael, 2014] L. Michael. Simultaneous Learning and Prediction. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning*, pages 348–357, Vienna, Austria, 2014.

[Miller, 1956] G. A. Miller. The Magic Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63(2):81–97, 1956.

[Mitchell *et al.*, 2015] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-Ending Learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2302–2310, Austin, TX, U.S.A., 2015.

[Muggleton, 1991] S. H. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

[Murphy and Medin, 1985] G. L. Murphy and D. L. Medin. The Role of Theories in Conceptual Coherence. *Psychological Review*, 92(3):289–316, 1985.

[Nevo and El-Yaniv, 2003] Z. Nevo and R. El-Yaniv. On Online Learning of Decision Lists. *Journal of Machine Learning Research*, 3:271–301, 2003.

[Prakken, 2010] H. Prakken. An Abstract Framework for Argumentation with Structured Arguments. *Argument and Computation*, 1(2):93–124, 2010.

[Richardson and Domingos, 2006] M. Richardson and P. Domingos. Markov Logic Networks. *Machine Learning*, 62(1):107–136, 2006.

[Rivest, 1987] R. L. Rivest. Learning Decision Lists. *Machine Learning*, 2(3):229–246, 1987.

[Schuurmans and Greiner, 1994] D. Schuurmans and R. Greiner. Learning Default Concepts. In *Proceedings of the 10th Canadian Conference on Artificial Intelligence*, pages 99–106, Banff, Alberta, Canada, 1994.

[Valiant, 1984] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Valiant, 2000] L. G. Valiant. Robust Logics. *Artificial Intelligence*, 117(2):231–253, 2000.

[Valiant, 2006] L. G. Valiant. A Quantitative Theory of Neural Computation. *Biological Cybernetics*, 95(3):205–211, 2006.