Non-Monotonic Bounded Reasoners

:: Paolo Baldi and Fabio D'Asaro

Abstract

We put forward a proposal for logics handling both non-monotonic reasoning and boundedly rational agents. We propose a hierarchy of depth-bounded non-monotonic logics. As the parameter k, which controls the level of reasoning depth, increases, non-monotonicity decreases while inferential power, in a classical sense, increases. Ultimately, these logics aim to provide non-monotonic approximations of classical logic. We present arguments and evidence supporting the adoption of this framework.

Keywords

Logic programming; depth-bounded boolean logics; non-monotonic reasoning; bounded rationality;

Sumbitted 11 December 2024
Revised 15 January 2025
Accepted 16 January 2025

How to Cite

Baldi, P., & D'Asaro, F. A. Non-monotonic bounded reasoners. The Reasoner, 19(1). https://doi.org/10.54103/1757-0522/27548

Introduction

This paper focuses on two key issues that have attracted significant attention in non-classical logical investigations: the *non-monotonic* nature of cognitive agents and their *bounded* reasoning capabilities.

Cognitive agents typically infer, from their available information *more* than what is allowed by classical logic. In this "jumping to conclusion", the information that they infer should also be reasonably retracted upon the arrival of new information. This is the *non-monotonic* aspect of common sense reasoning and contrasts with classical (Tarskian) logic, where once a conclusion is reached, it remains valid regardless of the additional information. Logic programming in particular has proved to be a powerful computational tool to model this dynamic of non-monotonic reasoning.

On the other hand, cognitive agents also infer *less* than what is prescribed by classical logic. This is because classical logical inferences are computationally hard and real reasoning agents have *bounded* cognitive resources. Among the various logical formalisms introduced to model bounded reasoning agents, here we focus in particular on the approach based on *Depth-Bounded Boolean Logics* (DBBLs in the following), see D'Agostino, Gabbay, et al. (2024: *Depth-Bounded Reasoning: Volume I: Classical Propositional Logic*, College Publications). This approach provides a whole family of logical systems that are meant to approximate classical logic, with each system tractable, in computational terms.

In this paper we propose a formal framework for combining non-monotonicity with bounded reasoning, and more precisely the ideas of logic programming with those of DBBLs.

DBBLs classify logical inferences in a hierarchy, based on how much they employ information that goes beyond the actually available information.

As a typical example of such inferences involving additional information, consider reasoning by cases: to prove ψ, an agent may need to show that ψ is the case both under supposition that ϕ is false, and under the supposition that ϕ is true. In this example, the supposition that ϕ is true or that ϕ is false does not count as information *actually possessed* by the agent, but is only *virtual*, i.e. instrumental for the derivation of ψ.

A logic $\vdash_0$ which does not manipulate any virtual information at all has been presented proof-theoretically in a system which is a mixture of classic natural deduction and a tableau method. A (unsigned) presentation of this system is provided in Figure 1, see D'Agostino, Gabbay, et al. (2024) for more details, and for a corresponding information-based semantics. This calculus consists of classical introduction and elimination rules for connectives (*intelim rules* for short). Intelim rules induce in a natural way the deducibility relation $\vdash_0$.

Completeness with respect to classical propositional logic is obtained by adding to $\vdash_0$ the rule of bivalence (RB), shown in Figure 2.

The rule of bivalence is taken to be the "difficult" part of an inference, both from a cognitive and computational point of view. Indeed, from the cognitive point of view, RB requires "guessing" the truth value of a formula ϕ, beyond the information that is actually given. Computationally, it is only the unbounded use of RB which makes classical logic intractable.

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

$$\frac{a \quad b}{a \wedge b}\,\mathrm{I}\wedge \qquad \frac{\neg a}{\neg(a \wedge b)}\,\mathrm{I}\neg\wedge_1 \qquad \frac{\neg b}{\neg(a \wedge b)}\,\mathrm{I}\neg\wedge_2$$

$$\frac{\neg a \quad \neg b}{\neg(a \vee b)}\,\mathrm{I}\neg\vee \qquad \frac{a}{a \vee b}\,\mathrm{I}\vee_1 \qquad \frac{b}{a \vee b}\,\mathrm{I}\vee_2$$

$$\frac{a}{\neg\neg a}\,\mathrm{I}\neg\neg \qquad \frac{a \quad \neg a}{\bot}\,\mathrm{I}\bot$$

$$\frac{a \vee b \quad \neg a}{b}\,\mathrm{E}\vee_1 \qquad \frac{a \vee b \quad \neg b}{a}\,\mathrm{E}\vee_2 \qquad \frac{\neg(a \vee b)}{\neg a}\,\mathrm{E}\neg\vee_1 \qquad \frac{\neg(a \vee b)}{\neg b}\,\mathrm{E}\neg\vee_2$$

$$\frac{\neg(a \wedge b) \quad a}{\neg b}\,\mathrm{E}\neg\wedge_1 \qquad \frac{\neg(a \wedge b) \quad b}{\neg a}\,\mathrm{E}\neg\wedge_2 \qquad \frac{a \wedge b}{a}\,\mathrm{E}\wedge_1 \qquad \frac{a \wedge b}{b}\,\mathrm{E}\wedge_2$$

$$\frac{\neg\neg a}{a}\,\mathrm{E}\neg\neg \qquad \frac{\bot}{a}\,\mathrm{E}\bot$$

Figure 1: Introduction and Elimination rules for Boolean operators ¬, ∨, ∧. To make it easier to refer to the rules in the remainder of the paper, each rule has been given a unique label shown on the right side of each rule.

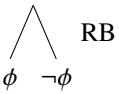


Figure 2: The rule of bivalence *(RB) for short), also known as the* cut rule*, where ϕ is some propositional formula.*

The hierarchy of DBBLs $\vdash_k$ is then defined by the integer *depth* parameter k, standing for the maximum number k of nested applications of RB allowed. As the depth $k \to \infty$ we obtain in the limit classical propositional logic.

Let us provide a few more formal notions, for introducing $\vdash_k$. We denote by $\mathcal{L}$ our language, consisting of the set of propositional variables $\mathrm{Var}_\mathcal{L}$ and the connectives $\neg, \vee, \wedge$ and by $\mathrm{Fm}_\mathcal{L}$ the set of formulas obtained by closing off recursively as usual the variables in $\mathrm{Var}_\mathcal{L}$ under the connectives. For any subset $T \subseteq \mathrm{Fm}_\mathcal{L}$, we denote by $\mathrm{Var}_\mathcal{L}(T)$ the subset of $\mathrm{Var}_\mathcal{L}$ of propositional variables occurring in T.

We then present the definition of $\vdash_k$, slightly simplifying from D'Agostino, Gabbay, et al. (2024) as follows.

Definition 1. *Let $\mathcal{T} \cup \{\phi\} \subseteq \mathrm{Fm}_\mathcal{L}$. We say that $\mathcal{T} \vdash_{k+1} \phi$ for $k \geq 0$ if and only if there is $a \in \mathrm{Var}_\mathcal{L}(\mathcal{T} \cup \{\phi\})$ such that $\mathcal{T}, a \vdash_k \phi$ and $\mathcal{T}, \neg a \vdash_k \phi$.*

Note that RB is only implicit in the recursive definition above. Nevertheless, we find it useful for readability to include RB in proof trees as an explicit rule of $\vdash_k$, despite the abuse of notation.

An important result of DBBLs is that for any fixed $k \geq 0$, deducibility and refutability of formulas in $\vdash_k$ is polynomial, hence tractable.

As a running example of the application of DBBLs to cognitive agents, consider the following variation on a well-known logical puzzle:

Example 1 (Children)**.** *Let Anne, Martha, and Jane be children. We know that Anne is blonde, while Jane is not. Additionally, Anne looks at Martha, and Martha looks at Jane (see Figure 3 for a diagrammatic representation).*

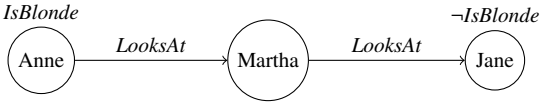


Figure 3: Anne looks at Martha, who in turn looks at Jane. Is any blonde girl looking at a non-blonde girl?

The puzzle involves answering the question:

Is it the case that a blonde child is looking at a non-blonde child? (1)

This puzzle was informally and empirically examined in Stanovich (1999: Who Is Rational? Studies of Individual Differences in Reasoning*, Lawrence Erlbaum). Participants could choose one of the following possible answers:*

(A) Yes

(B) No

(C) Cannot be determined

It has been observed that approximately 80% *of subjects selected option (C) "Cannot be determined." However, the correct answer is actually (A) "Yes." This can be illustrated by examining two scenarios based on whether Martha is blonde or not. In the first scenario, if Martha is blonde, then she is looking at Jane, who is not blonde. In the second scenario, if Martha is not blonde, then Anne, who is blonde, is looking at Martha. In both cases, the required relationship is maintained. In a first approximation, a first-order axiomatization of this scenario is*

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

Figure 4: DBBL Intelim derivation for the children example.

$\mathcal{T}$ = {*IsBlonde*(*Anne*), ¬*IsBlonde*(*Jane*), *LooksAt*(*Anne*, *Martha*), *LooksAt*(*Martha*, *Jane*)}.

Note that 0-depth logic does not allow to infer such conclusion. The empirical results thus suggest that people tend to behave as 0*-depth reasoning agents. A* 1*-depth reasoning agent, however, would be able to correctly apply RB on Martha's hair color and solve the puzzle. This is shown in the derivation in Figure 4, where we decorate the initial assumptions with* $*$.

The example provides motivation for adopting DBBLs to model similar situations that require reasoning about hypothetical scenarios.

In the remainder of this paper, we demonstrate that DBBLs are

inadequate if individuals are not permitted to choose the incorrect option (C). We argue that to better understand their behavior in this alternative scenario, it is necessary to incorporate some elements of non-monotonicity into DBBLs. This approach is especially relevant in decision-making contexts where individuals face time constraints and encounter counterintuitive or cognitively challenging problems.

Non-monotonic Reasoning and Logic Programming

Logic programming implements non-monotonicity largely based on a non-classical principle known as *Negation as Failure* (*NF* for short). This principle states that if a certain fact cannot be proven from a given theory, then its negation may be derived. To exemplify its relevance for commonsense reasoning, if a train from Lecce to Verona is not listed in a station's timetable, we typically – and reasonably – conclude that such a train does not exist. Later, if we receive new information that such a train does, in fact, exist – say, we access an updated timetable – we must retract our previous conclusion. Inferences based on NF are, therefore, tentative and subject to change when new information is made available, that is, non-monotonic.

Logic programs consists of *facts* of the form

$$\texttt{f.}$$

and *rules* of the form

$$\texttt{h :- b}_1,\ldots,\texttt{b}_n,\texttt{not b}_{n+1},\ldots,\texttt{not b}_{n+m}. \tag{2}$$

to be read as "h if b_1 and b_2 and...b_n and not b_{n+1} and... not b_{n+m}". At first glance, it may thus seem that $\texttt{f}, \texttt{h}, \texttt{b}_i$, $\texttt{not b}_i$ may be read

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

as literals, with the `:-` sign serving as an "implication" ← connective and the comma representing a "conjunction" connective.

There are however some significant differences between logic programming and the corresponding classical logic notions.

First, rather than a classical negation, `not p` is to be interpreted as a Negation as Failure, i.e. lack of information concerning `p`. Second, *modus tollens* does not apply to the `:-` operator. Specifically, consider a rule `p :- q`. While the truth of `p` can be inferred when `q` is true, the falsity of `q` cannot be inferred from the falsity of `p`.

A consolidated approach to interpret these constructs is due to Vladimir Lifschitz and consists in computing the *stable models* or *answer sets* of a given logic program. Informally, an answer set is a minimal and consistent set of literals that represents the information implicitly provided by the rules and facts of the program. Here we illustrate the concept only with examples, referring the reader to e.g. Lifschitz (2019: *Answer Set Programming*, Springer Cham) for a formal description. Consider first the following program.

```
q :- not p.                                    (3)
```

This program has a unique answer set, i.e. {`q`}. This is because the program does not provide any information about `p` and states that, by default, if the agent does not hold positive information about `p` then it may infer that `q` is the case. This latter is the minimal information inferrable from the program and is thus its unique answer set.

Assume now the program (3) is extended with the fact `p`. The rule of the program will now cease to fire and the unique answer set of the resulting program will be this time {`p`}, retracting the previ-

ously endorsed information q. This shows how answer sets capture the non-monotonic nature of negation as failure in logic programming.

Note that, in the light of the mechanism of negation as failure, answer sets need not be unique. To briefly illustrate the concept, consider now the following program :

```
p :- not q.
q :- not p.
```

This is well-known for having two answer sets, i.e. :

{p} and {q}.

Again, {p} is an answer set since, if we were to fire the first rule of the calculus, we would obtain p, due to the lack of information about q. This would then inhibit the application of the second rule where the negation of p is present in the premise. A similar reasoning leads to obtaining the answer set {q}. Both answer sets provide self-consistent minimal models, satisfying the original rules without contradiction. Note that, if we were to add both literals p and q to the program, none of the rules would fire and the only answer set would turn out to be {p, q}.

This latter is an instance of a more general fact concerning answer sets: if a program contains as facts all the literals in its language, i.e. the program represents an agent holding *complete information*, then non-monotonicity ceases to play any role in inference. Formally, in these scenarios either the program has no answer set, or there is a unique answer set amounting to all the facts of the program.

Finally, it is important to highlight that while our discussion of logic programming primarily focuses on propositional programs,

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

the stable set semantics can also accommodate first-order theories through a process known as *grounding*. This process involves substituting (or *grounding*) variables with constants within the logic program. This allows us, in the following, to use examples in a first-order formalism, but apply on them exclusively propositional reasoning. This convention is particularly useful also for allowing us to focus only on DBBLs, which are propositional logics, rather than their first-order counterparts, which are considerably more intricate, see e.g. D'Agostino, Larese, and Modgil (2021: "Towards depth-bounded natural deduction for classical first-order logic", *IfCoLoG Journal of Logics and their Applications*, pp. 423–451).

A hierarchy of non-monotonic bounded agents

In this section, we present the main proposal of this paper: the combination of non-monotonic reasoning and bounded reasoning.

This merging is both technically and conceptually challenging. Broadly speaking, the key conceptual issue at hand here is determining whether the non-monotonicity observed in logic programming is: (i) a suboptimal feature of inference that ideal agents should avoid whenever possible, or (ii) a neutral or even beneficial characteristic that cognitively capable agents may choose to utilize. Recalling that non-monotonic reasoning is based on supra-classical inferences, or "jumps to conclusions" and that we related RB with cognitive resources, we may thus either: (i) view RB as inhibiting supra-classical inferences, meaning that higher-depth agents are "less" non-monotonic than their lower-depth counterparts, or (ii) view RB as being neutral towards such inferences or even enhancing them, allowing thus for "more" non-monotonicity, on the basis of virtual information.

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

Let us note that in work D'Asaro, Baldi, and Primiero (2021: "Introducing k-lingo: a k-depth Bounded Version of ASP System Clingo", *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, DOI: 10.24963/kr.2021/65, pp. 661–665) we followed option (ii), albeit implicitly, taking non-monotonicity and depth-boundedness as two orthogonal phenomena. There we preliminarily developed a hierarchy of bounded non-monotonic systems that approximate standard logic programming in the limit. In other words, we developed *bounded non-monotonic logics* approximating a *unbounded non-monotonic logic*.

Instead, the proposal we put forward here joins non-monotonicity and depth-boundedness, under the perspective that more powerful agents will also be less non-monotonic. We introduce thus *bounded non-monotonic logics* that approximate *classical logic*.

The idea underlying this proposal is that at least some form of non-monotonicity is due to the limitations in cognitive power of agents on the one hand and the real-world pressure of providing answers or making decisions on the other.

A thorough analysis of the actual and virtual information available to an agent should then result in an inhibition of her jumps to conclusions, characterizing non-monotonic reasoning. Let us consider a variant of Example 1 as an illustration of this phenomenon.

Example 2 (Children, continued)**.** *Recall from Example 1 that Anne, Martha, and Jane are children, where Anne is blonde, Jane is not, and Anne looks at Martha, who in turn looks at Jane.*

In this example, we pose a new question: how would people respond if presented with only two options (A) "Yes" and (B) "No," when the commonly chosen but incorrect option (C) "Cannot be determined" is omitted? We hypothesize that, under strict time

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

constraints, most individuals would select (B) "No." This hypothesis is based on 1) the riddle does not explicitly present a blonde child looking at a non-blonde child and 2) the agents lack the means to infer such relation from the available data. Thus, they may infer that no such relationship exists. We tested this hypothesis in a small-scale experiment. Participants were asked to respond to Question (1) *within a* 1*-minute time limit, choosing between options (A) "Yes" and (B) "No." A total of* 63 *participants took part in the study. Of these,* 23 *participants (*36.5%*) selected option (A), while* 40 *participants (*63.5%*) selected option (B). The observed preference for option (B) was statistically significant, differing from random choice (*$\chi^2 = 4.0635$*,* $p = 0.04382$*), with a statistical power greater than* 80% *for a medium effect size (*0.5*).*

Our preliminary experimental evidence suggests that bounded agents typically choose the erroneous answer (B). On the other hand, classical agents, i.e., agents correctly capable of reasoning by cases via RB, thus going for the classically correct answer (A). We formalize this behavior in Example 2, assuming that 0-depth agents perform non-monotonic inferences, while 1-depth agents inhibit such inferences through reasoning by cases.

Before proceeding with our proposal, let us recall that rules of logic programs are only deceptively similar to implication, since they fail to satisfy modus tollens.

On the other hand, reading implication $\phi \to \psi$ as $\neg\phi \vee \psi$, it can be easily shown that modus tollens hold even for the logic $\vdash_0$.

For these reasons, following, e.g., Miller (2022: "A Survey of the Proof-Theoretic Foundations of Logic Programming", *Theory and Practice of Logic Programming*, DOI: 10.1017/S1471068421000533, pp. 859–904), we opt for encoding the clauses of logic programs directly as *rules* to be added to the proof

system in 1. Consider a program containing a rule

```
q :- p.
```
(4)

Rather than the formula $\neg p \vee q$, a theory including this rule will add to a logical system, such as the one for $\vdash_0$ in Figure 1, the rule

$$\frac{p}{q}. \tag{5}$$

Similarly, a rule

```
q :- not p.
```
(6)

will be rendered as

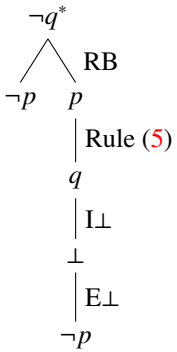
$$\frac{\neg p}{q}$$

Note that we are here conflating the standard negation $\neg$ with the negation as failure `not` from logic programming, thus implicitly assuming a closed worlds assumption for the formulas in the premises. More sophisticated logic programming formalisms allow for a fine-grained control of negation, including two distinct operators, the non-monotonic `not` interpreted as negation as failure, but also a classical negation $\neg$.

Contrary to the usual logical rules, p and q are here taken to be concrete atoms, and may not be substituted with arbitrary formulas. Henceforth, by a *theory* $\mathcal{T}$ we will mean a set of formulas, together with additional rules of the form (5) above.

An analog of modus tollens does not hold for $\vdash_0$ extended with rules of the form above, but only for higher DBBLs. Indeed, assume one has a theory containing both the literal $\neg q$ and the rule (5) , i.e., the theory $\mathcal{T} = \left\{\frac{p}{q}, \neg q\right\}$. In $\vdash_0$, it is not possible to infer $\neg p$ using rule (5) and the fact $\neg q$. On the other hand, this is again possible in 1-depth logic, as follows:

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1



In other words, a 0-depth agent can only apply the rule (5) forward. Nevertheless, a 1-depth agent, conjecturing about the information about the antecedent, may simulate modus tollens. This is coherent with the failure of modus tollens in logic programming on the one hand, and the empirical results attributing more cognitive weight to modus tollens over rules such as modus ponens, on the other, see e.g. Stenning and Lambalgen (2008: *Human Reasoning and Cognitive Science*, MIT Press). We thus believe that our formalism can shed light on the conceptual debate concerning the controversial status of modus tollens in non-monotonic reasoning.

Henceforth, we denote by $ST(\mathcal{T})$ the set of answer sets of a theory $\mathcal{T}$, which may possibly include rules corresponding to rules of logic programs.

We are now ready to introduce our depth-bounded non-monotonic consequence relations, denoted by $\mid\!\sim_k$ for $k = 0, 1, \ldots$, in analogy to the Depth-Bounded Boolean consequence relations $\vdash_k$. Unlike their DBBLs counterpart, $\mid\!\sim_k$ exhibits non-monotonicity, and furthermore, in line with (i), such non-monotonicity will reduce as k increases.

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

Let us start from defining the $\mid\sim_0$ consequence relation.

Definition 2. *We write $\mathcal{T} \mid\sim_0 \phi$ iff either $ST(\mathcal{T}) = \emptyset$ and $\mathcal{T} \vdash_0 \phi$, or $ST(\mathcal{T}) \neq \emptyset$ and $S \vdash_0 \phi$ for each $S \in ST(\mathcal{T})$.*

This means that $\mid\sim_0$ permits to entail from a theory only those formulas that are 0-depth consequences of *all* its answer sets, if there are any, and otherwise it reduces to $\vdash_0$.

Note that while $\vdash_0$ is weaker than classical logic, $\mid\sim_0$ is *incomparable* with classical logic. Indeed, as for $\vdash_0$ we have $\emptyset \not\mid\sim_0 p \vee \neg p$. On the other hand, given the non-monotonicity of the answer sets, one may infer from a theory also more than what is classically inferable. Consider indeed the theory $\mathcal{T}$ encoding Equation (3).

$$\mathcal{T} = \left\{\frac{\neg p}{q}\right\} \tag{7}$$

We have that $ST(\mathcal{T}) = \{\{q\}\}$, hence there is a unique answer set for $\mathcal{T}$. One has thus $\mathcal{T} \mid\sim_0 q$, since $q \vdash_0 q$, while $\mathcal{T} \nvdash q$ and in particular also $\mathcal{T} \not\vdash_0 q$.

Before proceeding to the introduction of the hierarchy, let us denote by $Dec(\mathcal{T})$ the subset of the variables in $\mathrm{Var}_{\mathcal{L}}$ that are *decided* by $\mathcal{T}$ at depth 0, i.e. the variables a such that either $\mathcal{T} \vdash_0 a$ or $\mathcal{T} \vdash_0 \neg a$.

Definition 3. *We write $\mathcal{T} \mid\sim_{k+1} \phi$ for $k \geq 0$ iff there exists $a \in \mathrm{Var}_{\mathcal{L}}(\mathcal{T} \cup \{\phi\}) \setminus Dec(\mathcal{T})$, such that $\mathcal{T}, a \mid\sim_k \phi$ and $\mathcal{T}, \neg a \mid\sim_k \phi$.*

A few words on the restriction on the virtual information a occurring in the application of RB in the definition above are in place. First, DBBLs typically allow for a restriction on the set of virtual information. Here we have opted for a rather standard restriction (see Definition 1), namely the variables occurring in the premises

$\mathcal{T}$ and the formula ϕ that we want to prove. On the other hand, we are also imposing the additional requirement that the virtual information is not already decided by $\mathcal{T}$. Note that, if this were not the case, RB would be useless. Unpacking the recursive definition above, this means that each application of RB at depth k requires the use of a literal *distinct* from the ones used at depths j for $j < k$.

While this additional constraint is harmless for DBBLs, it makes a difference in our non-monotonic hierarchy, as we will see in the next examples.

First, let us recall our former example (7). We have showed that $\mathcal{T} \vdash_0 q$. To show that $\mathcal{T} \vdash_1 q$, we need to apply RB either on q or p i.e. show that either both $\mathcal{T}, q \vdash_0 q$ and $\mathcal{T}, \neg q \vdash_0 q$ or both $\mathcal{T}, \neg p \vdash_0 q$ and $\mathcal{T}, p \vdash_0 q$. Let us start from applying RB on q. Clearly $\mathcal{T}, q \vdash_0 q$ since $\mathcal{T}, q \vdash_0 q$. On the other hand, $ST(\mathcal{T} \cup \{\neg q\}) = \emptyset$, hence $\mathcal{T}, \neg q \not\vdash_0 q$, since $\mathcal{T}, \neg q \nvdash_0 q$. Let us consider now an application of RB on p. On the one hand $\mathcal{T}, \neg p \vdash_0 q$, since $\mathcal{T}, \neg p \vdash_0 q$, but on the other hand we have $\mathcal{T}, p \not\vdash_0 q$. Indeed $ST(\mathcal{T} \cup \{p\}) = \{\{p\}\}$ and $p \nvdash_0 q$. In summary, we have obtained that $\mathcal{T} \vdash_0 q$, while $T \not\vdash_1 q$. Thus, in contrast with DBBLs, where $\vdash_0 \subseteq \vdash_1$, here we have $\vdash_0 \not\subseteq \vdash_1$.

As a final note, recall from the children Example 2 that people seem to show a preference for option (B) "No." We now briefly show that this behavior can be modeled in our framework.

Example 3 (Children, continued (ii))**.** *Consider the theory* $\mathcal{T} = \{IsBlonde(Anne), \neg IsBlonde(Jane), LooksAt(Anne, Martha), LooksAt(Martha, Jane)\}$ *from our running Example 1, and extend it to* $\mathcal{T}'$*, by adding the rule*

$$\frac{\neg Answer(A)}{Answer(B)}$$

encoding a closed world *assumption, and the rule*

$$\frac{LooksAt(x, y) \wedge IsBlonde(x) \wedge \neg IsBlonde(y)}{Answer(A)}$$

This rule includes in its premises variables that can be eliminated through grounding, considering that our language is finite. At depth 0*, a non-monotonic DBBL agent that does not make assumptions about whether Martha is blonde or not would fail to obtain answer A, and thus non-monotonically infer that B is the correct answer. In other words, we have* $ST(\mathcal{T}') = \{\{\mathcal{T} \cup \{Answer(B)\}\}\}$ *and thus* $\mathcal{T}' \mid\!\sim_0 Answer(B)$*.*

This is consistent with the preliminary experimental results in Example 1. Note that an agent able to apply RB on IsBlonde(Martha) would be able to correctly infer that, in fact, answer (A) "Yes" is the correct answer to the puzzle, as its reasoning would reduce to the classical derivation in Figure 4.

The examples illustrate the feature distinguishing our hierarchy from other works integrating DBBLs and non-monotonic reasoning, such as D'Asaro, Baldi, and Primiero (2021: pp. 661–665) and Chapter 3 of D'Agostino, Gabbay, et al. (2024). In those works we have, for instance, that a 0-depth non-monotonic consequence relation is included in the 1-depth one, while in our case the consequence relations $\{\mid\!\sim_k\}_{k\in\mathbb{N}}$ are all incomparable. In particular, when focusing on those formulas that are not classically derivable, as k increases the consequence relations allow us to infer *less*.

This is a crucial point for proving that the $\{\mid\!\sim_k\}_{k\in\mathbb{N}}$ hierarchy approximate classical logic. By "approximate" we mean, more precisely what follows.

Theorem 1. *Let* $\mathcal{T}$ *be a theory consisting of classical formulas*

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

and rules of the form (5)*. Then,*

- *If* $\mathcal{T} \vdash \phi$ *then there exists a* $k \in \mathbb{N}$ *such that* $T \mid\sim_k \phi$.
- *If* $\mathcal{T} \nvdash \phi$ *then there exists a* $k \in \mathbb{N}$ *such that* $\mathcal{T} \not\mid\sim_k \phi$.

Proof. The first claim is immediate. Indeed, if $\mathcal{T} \vdash \phi$, by the approximation result in D'Agostino, Gabbay, et al. (2024), there exists a $k \in \mathbb{N}$ such that $T \vdash_k \phi$. But this fact, by the definition of $\mid\sim_k$, implies $\mathcal{T} \mid\sim_k \phi$.

Assume now that $\mathcal{T} \nvdash \phi$, and by contradiction, that for each $j \in \mathbb{N}$, we have $\mathcal{T} \mid\sim_j \phi$. Now, as discussed above, each application of RB in Definition 3 requires the choice of a different piece of virtual information. Assume that $|\mathrm{Var}_{\mathcal{L}}(\mathcal{T} \cup \{\phi\})| = k$. Unpacking the recursive Definition 3, we get that $\mathcal{T} \mid\sim_k \phi$ if and only if for each maximal set of literals S built from $\mathrm{Var}_{\mathcal{L}}(\mathcal{T} \cup \{\phi\})$, we have $\mathcal{T}, S \mid\sim_0 \phi$. Now since S contains all the literals of the language, for what we discussed in the previous section, we have either $S\,T(\mathcal{T} \cup S) = \{S\}$ or $S\,T(\mathcal{T} \cup S) = \emptyset$. Hence in both cases $\mathcal{T}, S \mid\sim_0 \phi$ implies $\mathcal{T}, S \vdash_0 \phi$. We thus obtained that $\mathcal{T}, S \vdash_0 \phi$ for each maximal set of literals S from $\mathrm{Var}_{\mathcal{L}}(\mathcal{T} \cup \{\phi\})$. But this in turn means that $\mathcal{T} \vdash_k \phi$, hence $\mathcal{T} \vdash \phi$, contradicting our initial assumption.

□

To conclude, let us note a possible criticism against our approach. It may be taken as too strong in suggesting, at least implicitly, that ideal agents should avoid non-monotonicity in any situation, at any cost. We do not endorse this claim and believe that we can still make our hierarchy compatible with a weaker position regarding non-monotonicity. In particular, we believe that one can take non-monotonic agents using logics of lower depths of the

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1

hierarchy as rationally minimizing their "cost of reasoning", when facing decisions. We thus plan in future work to investigate how to embed our hierarchy in a wider decision-theoretic framework and how this can shed light also on different ways of combining bounded reasoning and non-monotonicity.

Acknowledgements

We thank the anonymous reviewers for suggestions that helped us improve on the final version of the paper.

Paolo Baldi acknowledges support from the project "Ethical Design for AI", part of the Spoke 6 of the PNRR project FAIR (PE00000013, CUP H97G22000210007), funded by the European Union-NextGenerationEU and the Italian Ministry of University and Research (MUR).

Paolo Baldi
Università degli Studi del Salento
Fabio D'Asaro
Università degli Studi di Verona

THE REASONER 19 (1) https://doi.org/10.54103/1757-0522/2025/1