HAL
open science

PRONOM: proof-search and countermodel generation for non-normal modal logics

Tiziano Dalmonte, Sara Negri, Nicola Olivetti, Gian Luca Pozzato

To cite this version:

Tiziano Dalmonte, Sara Negri, Nicola Olivetti, Gian Luca Pozzato. PRONOM: proof-search and countermodel generation for non-normal modal logics. AIIA 2019: 18th International Conference of the Italian Association for Artificial Intelligence, Nov 2019, Rende, Italy. hal-02439537

HAL Id: hal-02439537

https://hal.science/hal-02439537v1

Submitted on 14 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PRONOM: proof-search and countermodel generation for non-normal modal logics*

Tiziano Dalmonte[1][0000−0002−7153−0506], Sara Negri[2][0000−0003−3958−6312], Nicola Olivetti[1][0000−0001−6254−3754], and Gian Luca Pozzato[3][0000−0002−3952−4624]

[1] Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France - {tiziano.dalmonte,nicola.olivetti}@lis-lab.fr

[2] Department of Philosophy, University of Helsinki, Finland - sara.negri@helsinki.fi

[3] Dipartimento di Informatica, Universitá degli Studi di Torino, Turin, Italy - gianluca.pozzato@unito.it

Abstract. We present PRONOM, a theorem prover and countermodel generator for non-normal modal logics. PRONOM implements some labelled sequent calculi recently introduced for the basic system **E** and its extensions with axioms M, N, and C based on bi-neighbourhood semantics. PRONOM is inspired by the methodology of leanT^{A}P and is implemented in Prolog. When a modal formula is valid, then PRONOM computes a proof (a closed tree) in the labelled calculi having that formula as a root in the labelled calculi, otherwise PRONOM is able to extract a model falsifying it from an open, saturated branch. The paper shows some experimental results, witnessing that the performances of PRONOM are promising.

Keywords: Non-normal modal logics · Labelled sequent calculi · Theorem proving

1 Introduction

Non-Normal Modal Logics (NNML for short) have been studied since the seminal works by C.I. Lewis, Scott, Lemmon, and Chellas (for an introduction see [3]) in the 1960s. They are a generalization of ordinary modal logics that do not satisfy some axioms or rules of minimal normal modal logic **K**. They have gained interest in several areas such as epistemic and deontic reasoning, reasoning about games, and reasoning about "truth in most of the cases". In epistemic reasoning, where $\Box A$ is read as "the agent knows/believes A", it was early observed [21] that NNML offers a partial solution to the problem of omniscience: a non-omniscient agent would not necessarily be able to conclude that she knows (or believes) B from that fact that she knows both A and $A \to B$, that is $\Box B$ does not follows from $\Box A$ and $\Box A \to \Box B$. This corresponds to rejecting the K-axiom, or even more strongly, the rule of monotonicity (RM) $A \to B$ implies $\Box A \to \Box B$ and possibly also the rule of necessitation (if B is valid then also $\Box B$ is valid) as it corresponds to the assumption that the agent knows every logical validity.

* Supported by the ANR project TICAMORE ANR-16-CE91-0002-01, the Academy of Finland project 1308664 and INdAM project GNCS 2019 "METALLIC #2".

In deontic logic, where $\Box A$ is interpreted as "it is obligatory that A", NNML may offer a way-out to some well-known paradoxes caused by standard (normal) deontic logic. The simplest example is Ross' paradox [20]: let M denotes "the letter is mailed" and B "the letter is burnt", obviously $M \to (M \vee B)$, but from $\Box M$, i.e. the obligation of send the letter, it seems odd to conclude $\Box(M \vee B)$, that is the obligation to send the letter or to burn it. Again, in this case the "culprit" is the (RM) rule mentioned above. A similar analysis underlies the *gentle-murder* paradox. Moreover normal deontic logic does not allow one to represent conflicting obligations: for instance let A be "you go to the faculty meeting", it may hold both $\Box A$ and $\Box\neg A$ (the former because you are a member of the academic staff, the latter because you have a more important thing to do), without wanting $\Box\bot$, that by (RM) would trivialize obligations. Here the critical point is axiom C which allows one to conclude $\Box(A \wedge \neg A)$ from $\Box A$ and $\Box\neg A$. Moreover, also $\Box\top$ (whence the necessitation rule) has been rejected by some authors, on the base that a logical truth cannot be the object of an obligation.

A non-normal interpretation of modal operators has been considered in logics of Ability (see [18] and references therein) where the formula $\Diamond A$ is interpreted as "the agent has the ability of doing something which makes A true"; let R denote "Ann draws a red card" and B "Ann draws a black card", clearly $\Diamond(R \vee B)$ holds as Ann can choose a card from a normal deck of cards that will be either red or black, but unless she has a "magical" ability, she cannot ensure that she will pick a red card or a black one, thus it is reasonable (or at least consistent) to assume both $\neg\Diamond R$ and $\neg\Diamond B$. But this shows that the logic of ability does not satisfy the C axiom (in the dual form): $\Diamond(A \vee B) \to \Diamond A \vee \Diamond B$. NNML have also some interest in the area of game logic, more precisely it turns out that Monotonic logic extended with axiom $\Diamond\top$ is a particular case of *coalition logics*, see [19].

Finally, $\Box A$ can be interpred as "A is true in almost all cases" [2], with this interpretation axiom C clearly fails, as the fact that A and B are independently true in "almost all cases" does not entail that $A \wedge B$ will also be such; a similar situation arises with a probabilistic reading of $\Box A$ as "A is true with high probability" [18].

Non-normal modal logics enjoy a simple semantic characterization in terms of Neighbourhood models: these are possible world models where each world is equipped with a set of neighbourhoods, each one being itself a set of worlds; the basic stipulation is that a modal formula $\Box A$ is true at a world w if the set of worlds which make A true belongs to the neighbourhoods of w. A family of logics is obtained by imposing further closure conditions on the set of neighbourhoods.

In this paper we describe PRONOM (theorem PROver for NOnnormal Modal logics) a Prolog theorem prover for the classical cube of non-normal modal logic[4]. Not many theorem provers for NNML have been developed so far. Here is a brief account: in [8] optimal decision procedures are presented for the whole cube of NNML; these procedures reduce a validity/satisfiability checking in NNML to a set of SAT problems and then call an efficient SAT solver. For this reason they probably outperform any (implementation of) specific calculi for these log-

[4] A complete description of the whole cube of NNML will be provided in Section 2.

ics, but they do not provide explicitly "proofs", nor countermodels. A theorem prover for logic **EM** based on a tableaux calculus (very similar to the one in [10]) is presented in [9]: the system handles more complex Coalition Logic and Alternating Time Temporal logic, and it is implemented in ELAN, an environment for rewriting systems. Finally [11] presents a Prolog implementation of a non-normal modal logic containing both the $[\forall\forall]$ and the $[\exists\forall]$ modality; the fragment with just $[\exists\forall]$ coincides with the logic **EM**, which is covered also by our theorem prover.

The prover PRONOM implements the labelled sequent calculi presented in [4]. These calculi are based on *bi-neighbourhood* semantics, a variant of the neighbourhood semantics recalled above: in a bi-neighbourhood model each world has associated a set of *pairs* of neighbourhoods, the idea being that the two components of a pair provide independently a positive and negative support for a modal formula. The bi-neighbourhood semantics is particularly significant for logics without monotonicity and maybe of interest in itself. However the main reason to consider it, rather than the standard one, is that it is easier to generate countermodels in the bi-neighbourhood semantics than standard neighbourhood models. On the other hand, it is shown in [4] that the two semantics are equivalent, and more precisely standard neighbourhood models and bi-neighbourhood models can be constructively transformed into each other. The calculi are modular and make use of labels to represent both worlds and neighbourhoods in the syntax. They have invertible rules and provide a decision procedure for the respective logic. Because of the invertibility of the rules, a finite countermodel in the bi-neighbourhood semantics (whence in the standard one) can be directly extracted from a failed derivation. The Prolog implementation closely corresponds to the calculi: each rule is encoded by a Prolog clause of a `prove` predicate. This correspondence ensures in principle both the soundness and completeness of the theorem prover. Termination of proof search is obtained by controlling the non-redundant application of the relevant rules. PRONOM provides both proof search and countermodel generation: it searches for a derivation of an input formula, but in case of failure, it generates a countermodel (in the bi-neighbourhood semantics) of the formula.

As far as we know, PRONOM is the *first* theorem prover that provides both proof search and countermodel generation for the *whole* cube of non-normal modal logics. Although there are no benchmarks, its performance seems promising. The program PRONOM, as well as all the Prolog source files, including those used for the performance evaluation, are available for free usage and download at http://193.51.60.97:8000/pronom/.

2 Non-normal modal logics, neighbourhood semantics and labelled calculi

In this section, we present the classical cube of NNMLs, both axiomatically and semantically. The latter is defined in terms of bi-neighbourhood models [4] and it is equivalent to the standard neighbourhood semantics.

Let Atm be a countable set of propositional variables. The language $\mathcal{L}$ contains formulas given by the following grammar: $A ::= p \mid \bot \mid \top \mid A \vee A \mid A \wedge A \mid A \to A \mid \Box A$, where $p \in \mathsf{Atm}$. The minimal logic **E** in the language $\mathcal{L}$ is defined by adding to classical propositional logic the rule of inference

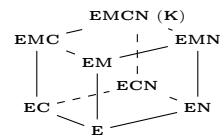
$$\text{RE}\ \frac{A \to B \qquad B \to A}{\Box A \to \Box B}\ ,$$

and can be extended further by choosing any combination of axioms M, C, and N below on the left, thus producing eight distinct logics (see the *classical cube*, below on the right).

M ▶ $\Box(A \wedge B) \to \Box A$

C ▶ $\Box A \wedge \Box B \to \Box(A \wedge B)$

N ▶ $\Box\top$



We recall that axioms M and N are respectively equivalent to the rules RM ($A \to B$ / $\Box A \to \Box B$) and RN (A / $\Box A$), and that axiom K ($\Box(A \to B) \to \Box A \to \Box B$) is derivable from M and C. As a consequence, we have that the top system **EMCN** is equivalent to the weakest normal modal logic **K**.

We consider here a variant of the standard neighbourhood semantics for NNMLs, called bi-neighbourhood semantics [4].

Definition 1. *A* bi-neighbourhood model *is a tuple $\mathcal{M} = \langle \mathcal{W}, \mathcal{N}_b, \mathcal{V} \rangle$, where $\mathcal{W}$ is a non-empty set of worlds (states), $\mathcal{V}$ is a valuation function, and $\mathcal{N}_b$ is a bi-neighbourhood function $\mathcal{W} \longrightarrow \mathcal{P}(\mathcal{P}(\mathcal{W}) \times \mathcal{P}(\mathcal{W}))$, where $\mathcal{P}$ denotes the power set. We say that $\mathcal{M}$ is a* M-model *if $(\alpha, \beta) \in \mathcal{N}_b(w)$ implies $\beta = \emptyset$, it is a* N-model *if for all $w \in \mathcal{W}$ there is $\alpha \subseteq \mathcal{W}$ such that $(\alpha, \emptyset) \in \mathcal{N}_b(w)$, and it is a* C-model *if $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \in \mathcal{N}_b(w)$ implies $(\alpha_1 \cap \alpha_2, \beta_1 \cup \beta_2) \in \mathcal{N}_b(w)$. The forcing relation for boxed formulas is as follows:*

$$w \Vdash \Box A \quad \textit{iff} \quad \textit{there is } (\alpha, \beta) \in \mathcal{N}_b(w) \textit{ s.t. } \alpha \subseteq [A] \textit{ and } \beta \subseteq [\neg A],$$

where $[A]$ is, as usual, the truth set of A in $\mathcal{W}$ obtained by the valuation $\mathcal{V}$.

In [4] it is shown that the bi-neighbourhood semantics characterises the whole cube of NNMLs, in the sense that:

Theorem 1. *A formula A is a theorem of* **E** *iff it is valid in all bi-neighbourhood models. The correspondence carries over to the extensions: A is a theorem of* **E**+(M/C/N) *iff it is valid respectively in all bi-neighbourhood M/N/C-models (including any combination of axioms/corresponding model conditions).*

It is instructive to recall also the standard neighbourhood semantics and see how the two semantics are related. A standard *neighbourhood model* has the form $\mathcal{M} = \langle \mathcal{W}, \mathcal{N}_s, \mathcal{V} \rangle$, where $\mathcal{W}$, $\mathcal{V}$ are as before, and $\mathcal{N}_s$ has type $\mathcal{W} \longrightarrow \mathcal{P}(\mathcal{P}(\mathcal{W}))$. The forcing relation for boxed formulas is: $w \Vdash \Box A$ iff $[A] \in \mathcal{N}_s(w)$. In addition we may consider the following conditions: a model $\mathcal{M}$ is *supplemented* if $\alpha \in$

$\mathcal{N}_s(w)$ and $\alpha \subseteq \beta$ implies $\beta \in \mathcal{N}_s(w)$, it *contains the unit* if $\mathcal{W} \in \mathcal{N}_s(w)$ for all $w \in \mathcal{W}$, and it is *closed under intersection* if $\alpha, \beta \in \mathcal{N}_s(w)$ implies $\alpha \cap \beta \in \mathcal{N}_s(w)$.

It is easy to see that every standard model gives rise to a bi-neighbourhood model, by taking for each neighbourhood $\alpha \in \mathcal{N}_s(x)$, the pair $(\alpha, \mathcal{W} \setminus \alpha)$. Moreover if the model is supplemented, contains the unit, or is closed under intersection the corresponding bi-neighbourhood model is a M/N/C model respectively.

On the other hand every bi-neighbourhood model can be transformed into a standard model [4]: given a bi-neighbourhood model $\mathcal{M} = \langle \mathcal{W}, \mathcal{N}_b, \mathcal{V} \rangle$ we can define the standard neighbourhood model $\mathcal{M}' = \langle \mathcal{W}, \mathcal{N}_s, \mathcal{V} \rangle$ by taking for all $w \in \mathcal{W}$, $\mathcal{N}_s(w) = \{\gamma \subseteq \mathcal{P}(W) \mid$ there is $(\alpha, \beta) \in \mathcal{N}_b(w)$ s.t. $\alpha \subseteq \gamma$ and $\beta \subseteq \mathcal{W} \setminus \gamma\}$. It can be proved that the two models are equivalent and that the transformation preserves additional properties (supplementation etc.) whenever the bi-neighbourhood model is a M/N/C model. For logics without monotonicity the above transformation can be optimized in order to obtain a model whose size is polynomially bounded by the size of the original one [4].

We turn now to present the labelled calculi for NNMLs based on the bi-neighbourhood semantics. The language $\mathcal{L}_{\mathsf{L}S}$ of labelled calculi extends $\mathcal{L}$ with a set $WL = \{x, y, z, ...\}$ of *world labels*, and a set $NL = \{a, b, c, ...\}$ of *neighbourhood labels*. We define *positive neighbourhood terms*, written $[a_1, ..., a_n]$, as finite multisets[5] of neighbourhood labels, with the unary multiset $[a]$ representing an atomic term. Moreover, if t is a positive term, then $\overline{t}$ is a negative term. Negative terms $\overline{t}$ cannot be proper subterms, in particular cannot be negated. The term τ and its negative counterpart $\overline{\tau}$ are neighbourhood constants.

Intuitively, positive (resp. negative) terms represent the intersection (resp. the union) of their constituents, whereas t and $\overline{t}$ are the two members of a pair of neighbourhoods in bi-neighbourhood models.

The formulas of $\mathcal{L}_{\mathsf{L}S}$ are of the following kinds:

$$\phi ::= x : A \mid t \Vdash^{\forall} A \mid t \Vdash^{\exists} A \mid x \in t \mid t \in \mathcal{N}(x).$$

Sequents are pairs $\Gamma \Rightarrow \Delta$ of multisets of formulas of $\mathcal{L}_{\mathsf{L}S}$. The fully modular calculi LSE^* are defined by the rules in Fig. 1.

The above version of the calculi are sound and complete for sequents that may appear in a backward proof search having at the root a single formula (these sequents belong to the class of regular sequents, see [4] for details). It is easy to see that in case of monotonic logics (i.e. logics containing M) the rule $R\Box$ can be simplified by eliminating the second premise. The reason is that an application of the rule $L \Vdash^{\exists}$ to the term $\overline{t}$ will introduce an element $y \in \overline{t}$ in the antecedent, so that the sequent immediately succeeds by rule M. So we can replace the rule $R\Box$ with the following:

$$\frac{t \in \mathcal{N}(x), \Gamma \Rightarrow \Delta, x : \Box A, t \Vdash^{\forall} A}{t \in \mathcal{N}(x), \Gamma \Rightarrow \Delta, x : \Box A} \; R\Box_M$$

Moreover the rule M can also be deleted as it is not applicable anymore.

[5] As a difference with [4] here terms are multisets rather than sets. This is ininfluent for the properties of the calculi.

Initial sequents: $x : p, \Gamma \Rightarrow \Delta, x : p \qquad x : \bot, \Gamma \Rightarrow \Delta \qquad \Gamma \Rightarrow \Delta, x : \top$

Propositional rules: As for **G3K** [12].

$$\frac{x \in t, x : A, t \Vdash^{\forall} A, \Gamma \Rightarrow \Delta}{x \in t, t \Vdash^{\forall} A, \Gamma \Rightarrow \Delta}\ L \Vdash^{\forall} \qquad \frac{x \in t, \Gamma \Rightarrow \Delta, x : A}{\Gamma \Rightarrow \Delta, t \Vdash^{\forall} A}\ R \Vdash^{\forall}$$

$$\frac{x \in t, x : A, \Gamma \Rightarrow \Delta}{t \Vdash^{\exists} A, \Gamma \Rightarrow \Delta}\ L \Vdash^{\exists} \qquad \frac{x \in t, \Gamma \Rightarrow \Delta, x : A, t \Vdash^{\exists} A}{x \in t, \Gamma \Rightarrow \Delta, t \Vdash^{\exists} A}\ R \Vdash^{\exists}$$

$$\frac{[a] \in \mathcal{N}(x), [a] \Vdash^{\forall} A, \Gamma \Rightarrow \Delta, \overline{[a]} \Vdash^{\exists} A}{x : \Box A, \Gamma \Rightarrow \Delta}\ L\Box$$

$$\frac{t \in \mathcal{N}(x), \Gamma \Rightarrow \Delta, x : \Box A, t \Vdash^{\forall} A \qquad t \in \mathcal{N}(x), \bar{t} \Vdash^{\exists} A, \Gamma \Rightarrow \Delta, x : \Box A}{t \in \mathcal{N}(x), \Gamma \Rightarrow \Delta, x : \Box A}\ R\Box$$

$$\frac{}{t \in \mathcal{N}(x), y \in \bar{t}, \Gamma \Rightarrow \Delta}\ M \qquad \frac{\tau \in \mathcal{N}(x), \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}\ N\tau \qquad \frac{}{x \in \overline{\tau}, \Gamma \Rightarrow \Delta}\ N\overline{\tau}$$

$$\frac{[a_1, ..., a_n] \in \mathcal{N}(x), [a_1] \in \mathcal{N}(x), ..., [a_n] \in \mathcal{N}(x), \Gamma \Rightarrow \Delta}{[a_1] \in \mathcal{N}(x), ..., [a_n] \in \mathcal{N}(x), \Gamma \Rightarrow \Delta}\ C$$

$$\frac{x \in [a_1], ..., x \in [a_n], \Gamma \Rightarrow \Delta}{x \in [a_1, ..., a_n], \Gamma \Rightarrow \Delta}\ dec \qquad \frac{x \in \overline{[a_1]}, \Gamma \Rightarrow \Delta \quad ... \quad x \in \overline{[a_n]}, \Gamma \Rightarrow \Delta}{x \in \overline{[a_1, ..., a_n]}, \Gamma \Rightarrow \Delta}\ \overline{dec}$$

Application conditions:
x is fresh in $R \Vdash^{\forall}$ and $L \Vdash^{\exists}$, a is fresh in $L\Box$, and x occurs in the conclusion of $N\tau$.

Fig. 1: The rules of LSE^*.

3 Design of PRONOM

In this section we present a Prolog implementation of the labelled calculi recalled in Section 2. The program, called PRONOM, is inspired by the "lean" methodology of $\mathsf{lean}T^AP$, even if it does not follow its style in a rigorous manner. The program comprises a set of clauses, each one of them implementing a sequent rule or an axiom of LSE and its extensions. The proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism, following the line of the theorem provers for modal and conditional logics in [7, 17, 16, 1, 15, 14, 13] and for preferential reasoning [5] in [6]. In the case of **EM** we consider both the modular version like in Fig. 1, and the optimised version in which the rule $R\Box$ is replaced by $R\Box_M$.

PRONOM represents a sequent with Prolog lists `Spheres`, `Gamma` and `Delta`. Lists `Gamma` and `Delta` represent the left-hand side and the right-hand side of the sequent, respectively. Elements of `Gamma` and `Delta` are labelled formulas, implemented by Prolog lists with two, three or four elements, as follows:

- standard formulas are pairs `[x,f]`, where `x` is a label and `f` is a formula;

- formulas of the form either $x \in t$ or $x \in \bar{t}$ are triples `[x,0,t]` (`[x,1,t]`, respectively), where `x` is a label and `t` represents term t; the inner value, either 0 or 1, is used to distinguish between positive and negative terms, t and $\bar{t}$, respectively;
- formulas of the form $t \models^{\exists} A$, or $t \models^{\forall} A$, or $\bar{t} \models^{\exists} A$, or $\bar{t} \models^{\forall} A$ are represented by quadruples `[exists,t,0,a]`, `[forall,t,0,a]`, `[exists,t,1,a]`, `[forall,t,1,a]`, respectively.

The list `Spheres` contains pairs of the form `[x,Items]`, where `Items` is the list of terms belonging to $N(x)$. Symbols $\top$ and $\bot$ are represented by constants `true` and `false`, respectively, whereas connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\Box$ are represented by `-`, `^`, `?`, `->`, and `box`. Propositional variables are represented by Prolog atoms. As an example, the Prolog lists

```
[ [x,[t]] ]
[ [y,1,t], [y,a], [forall,t,0,a^b] ]
[ [exists,t,1,a^b], [x,box(a)] ]
```

are used to represent the sequent $t \in N(x), y \in \bar{t}, y : A, t \models^{\forall} A \wedge B \Rightarrow \bar{t} \models^{\exists} A \wedge B, x : \Box A$.

Given a non-normal modal formula F represented by the Prolog term `f`, PRONOM executes the main predicate of the prover, called `prove`[6], whose only two clauses implement the functioning of PRONOM: the first clause checks whether F is valid and, in case of a failure, the second one computes a model falsifying F. In detail, the predicate `prove` first checks whether the formula is valid by executing the predicate:

```
terminating_proof_search(Spheres,Gamma,Delta,ProofTree,RBox,RExist,LAll).
```

This predicate succeeds if and only if the sequent represented by the lists `Spheres`, `Gamma` and `Delta` is derivable. When it succeeds, the output term `ProofTree` matches with a representation of the derivation found by the prover. Further arguments `RBox`,`RExist`, and `LAll` are used in order to control the application of rules $R\Box$, $R \Vdash^{\exists}$, and $L \Vdash^{\forall}$, for obtaining a terminating proof search. More in detail, let us consider the rule $R\Box$, which is applied (backward) to a sequent of the form $t \in N(x), \Gamma \Rightarrow \Delta, x : \Box A$: both the principal formulas $t \in N(x)$ and $x : \Box A$ are copied into the premises, then we need to prevent further applications of the same rule in a backward proof search. In order to control the application of this rule, the list `RBox` contains triples of the form `[x,a,t]` in order to keep trace of the fact that, in the current branch of the tree, the rule $R\Box$ has been already applied to $x : \Box A$ by using $t \in N(x)$. Therefore, the application of the rule is restricted by instantiating a Prolog variable `T` such that `[X,A,T]` does not belong to `RBox`. Similarly for `RExist` and `LAll`.

As an example, in order to prove that the sequent $x : \Box(A \wedge (B \vee C)) \Rightarrow x : \Box((A \wedge B) \vee (A \wedge C))$ is valid in **E**, one queries PRONOM with the goal:

[6] The user can run PRONOM without using the interface of the web application. To this aim, he just need to invoke the goal `prove(f)`.

```
terminating_proof_search([x, [  ]], [[x, (box (a ^ (b ? c)))]],
  [[x, (box ((a ^ b) ? (a ^ c)))]], ProofTree, [ ], [ ], [ ]).
```

Each clause of `terminating_proof_search` implements an axiom or rule of the sequent calculi LSE and extensions. To search for a derivation of a sequent $\Gamma \Rightarrow \Delta$, PRONOM proceeds as follows. First of all, if $\Gamma \Rightarrow \Delta$ is an instance of an axiom, the goal will succeed immediately by using the following clause:

```
terminating_proof_search(Spheres,Gamma,Delta,tree(axiom),_,_,_):-
    member([X,A],Gamma),
    member([X,A],Delta),!.
```

The modular, unoptimised, version for logic **EM** has also the following clause:

```
terminating_proof_search(Spheres,Gamma,Delta,tree(m),_,_,_):-
   member([_,List],Spheres),
   member(T,List),
   member([_,1,T],Gamma),!.
```

If $\Gamma \Rightarrow \Delta$ is not an instance of the axioms, then the first applicable rule will be chosen, e.g. if `Spheres` contains an element `[X, List]`, such that `List` contains `T`, representing that $t \in N(x)$, and `Delta` contains a formula `[X,box A]`, representing that $x : \Box A$ belongs to the right hand side of the sequent, then the clause implementing the $R\Box$ rule will be chosen, and PRONOM will be recursively invoked on the premises of such a rule. PRONOM proceeds in a similar way for the other rules. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible. As an example, the clause implementing $R\Box$ is as follows:

```
1.  terminating_proof_search(Spheres,Gamma,Delta,
                 tree(rbox,LeftTree,RightTree),RBox,RExist,LAll):-
2.      member([X,box A],Delta),
3.      member([X,SpOfX],Spheres),
4.      member(T,SpOfX),
5.      \+member([X,A,T],RBox),
6.      !,
7.      terminating_proof_search(Spheres,Gamma,[[forall,T,0,A]|Delta],
                                 LeftTree,[[X,A,T]|RBox],RExist,LAll),
8.      terminating_proof_search(Spheres,[[exists,T,1,A]|Gamma],Delta,
                                 RightTree,[[X,A,T]|RBox],RExist,LAll).
```

Line 5 implements the restriction on the application of the rule described above, and used in order to ensure a terminating proof search: given an instantiation of the Prolog variable `T`, the rule is applied only in case it has not been already applied by using the same `T` for the formula $x : \Box A$ in the current branch, namely `[X,A,T]` does not belong to `RBox`. Since the rule is invertible, Prolog cut `!` is used in line 6 to eventually block backtracking.

When the predicate `terminating_proof_search` fails, then the initial formula is not valid, and PRONOM extracts a model falsifying such a formula from an open saturated branch. This is computed by executing the predicate:

```
build_saturate_branch(Spheres,Gamma,Delta,Model,RBox,RExist,LAll).
```

This predicate has the same arguments of `terminating_proof_search`, with the exception of the fourth one: here the variable `Model` matches a description of an open, saturated branch obtained by applying the rules of the calculi to the initial formula. Since the very objective of this predicate is to build an open, saturated branch in the sequent calculus, its clauses are essentially the same as the ones for the predicate `terminating_proof_search`, however rules introducing a branch in a backward proof search are implemented by *pairs* of (disjoint) clauses, each one representing an attempt to build an open saturated branch. As an example, the following clauses implement the saturation in presence of a boxed formula $x : \Box A$ in the right hand side of a sequent:

```
1.   build_saturate_branch(Spheres,Gamma,Delta,Model,RBox,RExist,LAll):-
2.       member([X,box A],Delta),
3.       member([X,SpOfX],Spheres),
4.       member(T,SpOfX),
5.       \+member([X,A,T],RBox),
6.       build_saturate_branch(Spheres,Gamma,[[forall,T,0,A]|Delta],Model,
                                  [[X,A,T]|RBox],RExist,LAll).
7.   build_saturate_branch(Spheres,Gamma,Delta,Model,RBox,RExist,LAll):-
8.       member([X,box A],Delta),
9.       member([X,SpOfX],Spheres),
10.      member(T,SpOfX),
11.      \+member([X,A,T],RBox),
12.      build_saturate_branch(Spheres,[[exists,T,1,A]|Gamma],Delta,Model,
                                 [[X,A,T]|RBox],RExist,LAll).
```

PRONOM will first try to build a countermodel by considering the left premise of the rule $R\Box$, corresponding to recursively invoking the predicate `build_saturate_branch` on the premise introducing $t \models^{\forall} A$ in the right hand side of the sequent in line 6. In case of a failure, the saturation process is completed by considering the right premise of $R\Box$ introducing $\bar{t} \models^{\exists} A$ by the recursive call of line 12.

Clauses implementing axioms for the predicate `terminating_proof_search` are replaced by the last clause, checking whether the current sequent represents an open and saturated branch:

```
build_saturate_branch(Spheres,Gamma,Delta,model(Spheres,Gamma,Delta),_,_,_):-
    \+instanceOfAnAxiom(Spheres,Gamma,Delta).
instanceOfAnAxiom(_,Gamma,Delta):-member([X,A],Gamma),member([X,A],Delta),!.
```

Since this is the very last clause of the predicate `build_saturate_branch`, it is considered by PRONOM only if no other clause/rule is applicable, therefore the branch is saturated. The auxiliary predicate `instanceOfAnAxiom` checks whether the branch is open by proving that it is not an instance of the axioms. The third argument matches a term `model` representing the countermodel extracted from the lists `Spheres, Gamma`, and `Delta`.

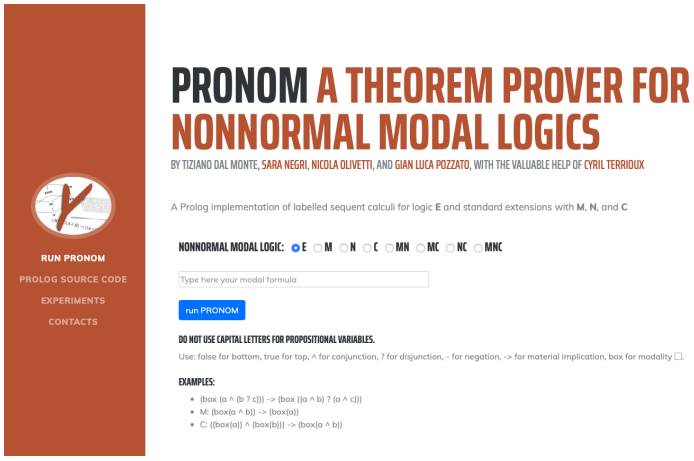


Fig. 2: Home page of PRONOM. When the user wants to check whether a formula F is valid, then (i) he selects the non-normal modal logic to use, (ii) he types F in the form and (iii) clicks the button in order to execute the calculi.

The implementation of the calculi for extensions of **E** is very similar: given the modularity of the calculi LSE*, the systems implementing the extensions are easily obtained by adding clauses for both the predicates `terminating_proof_search` and `build_saturate_branch` corresponding to the rules specifically tailored for the extensions under consideration. The only exception is logic **EM**, for which we give also an optimised version containing the rule $R\Box_M$ instead of $R\Box$. For the extensions of **EM** we only propose the version with $R\Box_M$ in place of $R\Box$.

PRONOM can be used by means of a simple web interface, implemented in php and allowing the user to check whether a non-normal modal formula is valid by using a computer or a mobile device. The web interface also allows the user to choose the modal system to adopt, namely **E** or one of the extensions mentioned in Section 2. When a formula is valid, PRONOM builds a pdf file showing a derivation in the sequent calculus LSE (or one of its extensions) as well as the LaTeX source file. Otherwise, a countermodel falsifying the initial formula is displayed. Prolog source codes and experimental results are also available. Some pictures of PRONOM are shown in Figures 2, 3 and 4.

4 Performance of PRONOM

The performance of PRONOM seems to be promising. We have tested it by running SWI-Prolog, version 7.6.4, on an Apple MacBook Pro, 2.7 GHz Intel Core i7, 8GB RAM machine. We have performed two kinds of experiments: 1. we have tested PRONOM over sets of valid formulas in the basic system **E** as well as in each considered extension; 2. we have tested PRONOM on randomly generated formulas, fixing different time limits, numbers of propositional variables, and levels of nesting of connectives.

Concerning 1, we have considered around hundred valid formulas obtained by generalizing schemas of valid formulas by varying some crucial parameters,

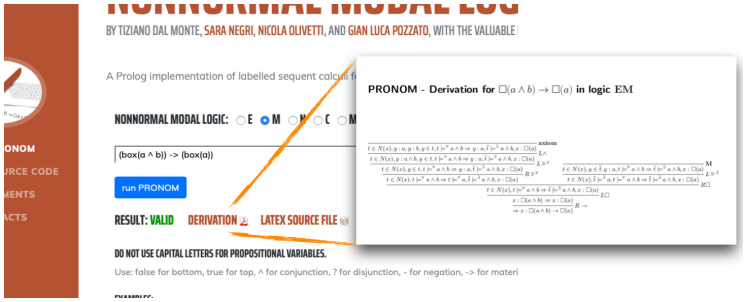


Fig. 3: When a formula is valid, PRONOM computes a pdf file as well as a LaTeX source file of a derivation.

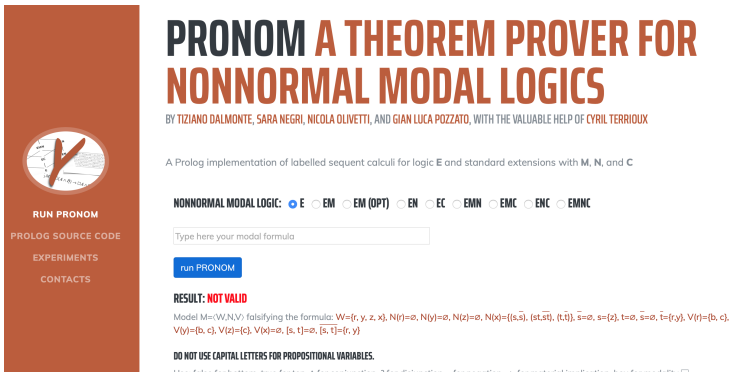


Fig. 4: When a formula is not valid, PRONOM computes and displays a counter model falsifying it.

like the modal degree of a formula (the level of nesting of the $\Box$ connective). For instance, we have considered the following schemas (valid in all systems):

$$(\Box(\Box(A_1 \wedge (B_1 \vee C_1)) \wedge \ldots \wedge \Box(A_n \wedge (B_n \vee C_n)))) \to (\Box(\Box((A_1 \wedge B_1) \vee (A_1 \wedge C_1)) \wedge \ldots \wedge \Box((A_n \wedge B_n) \vee (A_n \wedge C_n)))$$

$$(\Box^n C_1 \wedge \ldots \wedge \Box^n C_j \wedge \Box^n A) \to (\Box^n A \vee \Box^n D_1 \vee \ldots \vee \Box^n D_k)$$

We have obtained encouraging results: Table 1 reports results for **E**, from which it can be observed that PRONOM is able to answer in less than one second on more than the 75% of the tests, also on valid formulas with high modal degrees.

0.1 ms	1ms	100ms	1s	5s
51	32	26	22	18

Table 1: Number of timeouts of PRONOM over 91 valid formulas in **E**.

Concerning 2, we have tested PRONOM for **E** over 1000 random formulas, obtaining the experimental results of Table 2. It is worth observing that, even in case formulas are generated from 7 different atomic variables and with a high

level of nesting (10), PRONOM is able to answer in more than 80% of the cases within a time limit of 10 ms.

Number of variables/Depth	***1ms***	***10ms***	***1s***	***10s***
3 variables - depth = 5	3 %	2 %	0	0
3 variables - depth = 7	29 %	16 %	14 %	12 %
7 variables - depth = 10	27 %	19 %	14 %	9 %

Table 2: Percentage of timeouts in 1000 random tests (system **E**).

The random generation often leads to not valid formulas; as a consequence, this kind of tests has been useful also in order to evaluate the performance of PRONOM in computing countermodels: indeed, we have considered the number of timeouts in the execution of the top-level predicate `prove` described in the previous section, including the extraction of a countermodel in case of a failure in the proof search. Again, the experimental results seems to be adequate, and the time required for the generation of a counter model of a not valid formula is negligible with respect to the time needed to perform the whole computation.

We have repeated the above experiments also for all the extensions of **E** considered by PRONOM: to save space, we have omitted the results that can be all found at http://193.51.60.97:8000/pronom/#experiments.

It is worth noticing that the above experimental results refer to the Prolog component of PRONOM only, thus they do not take into account the effort of the graphical interface in computing the pdf file of the derivation. Since the web application often requires a long time to answer, we are currently working on improving the performances of PRONOM in such a way that the interface will first provide an answer about the validity of the formula, whereas the generation of the LaTeX/pdf file will be performed only if this option is explicitly selected by the user by clicking a suitable button. Moreover, we are planning to perform more accurate tests following the approach of [8], where randomly generated formulas can be obtained by selecting different degrees of probability about their validity.

5 Conclusions

We have described a Prolog Theorem prover for non-normal modal logics. As far as we know ours is the first program that provides both proof-search and countermodel generation for the whole cube of NNML. It implements directly, concisely, and modularly the labelled sequent calculi presented in [4]. The system provides both proof-search and countermodel construction: given a formula to check, the system ouputs either a derivation or a countermodel of the formula, the latter in the bi-neighbourhood semantics, a variant of the standard neighbourhood semantics. Although the implementation does not make use of any optimization or any sophisticated data structure, its performances are encouraging. In future research we intend to study some improvements like the use of free variables for term instatiation and other optimisations. We also intend to implement an automated and *efficient* transformation of the bi-neighbourhood countermodels into standard neighbourhood models, as shown in [4].

References

1. Alenda, R., Olivetti, N., Pozzato, G.L.: CSL-lean: A theorem-prover for the logic of comparative concept similarity. Electronic Notes in Theoretical Computer Science (ENTCS) **262**, 3 – 16 (2010)
2. Askounis, D., Koutras, C.D., Zikos, Y.: Knowledge means 'all', belief means 'most'. Journal of Applied Non-Classical Logics **26**(3), 173–192 (2016)
3. Chellas, B.F.: Modal Logic. Cambridge University Press (1980)
4. Dalmonte, T., Olivetti, N., Negri, S.: Non-normal modal logics: Bi-neighbourhood semantics and its labelled calculi. In: Bezhanishvili, G., D'Agostino, G., Metcalfe, G., Studer, T. (eds.) Advances in Modal Logic 12, proceedings of the 12th conference on Advances in Modal Logic, held in Bern, Switzerland, August 27-31, 2018. pp. 159–178. College Publications (2018)
5. Giordano, L., Gliozzi, V., Olivetti, N., Pozzato, G.L.: Analytic tableaux for KLM preferential and cumulative logics. In: Sutcliffe, G., Voronkov, A. (eds.) Logic for Programming, Artificial Intelligence, and Reasoning, 12th International Conference, LPAR 2005, Montego Bay, Jamaica, December 2-6, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3835, pp. 666–681. Springer (2005)
6. Giordano, L., Gliozzi, V., Pozzato, G.L.: KLMLean 2.0: A theorem prover for KLM logics of nonmonotonic reasoning. In: Olivetti, N. (ed.) Automated Reasoning with Analytic Tableaux and Related Methods - Proceedings of TABLEAUX 2007 (16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods). Lecture Notes in Artificial Intelligence LNAI, vol. 4548, pp. 238 – 244. Springer, Aix en Provence, France (July 2007)
7. Girlando, M., Lellmann, B., Olivetti, N., Pozzato, G.L., Vitalis, Q.: VINTE: An implementation of internal calculi for lewis logics of counterfactual reasoning. In: Nalon, C., Schmidt, R. (eds.) Proceedings of TABLEAUX 2017 - 26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. Lecture Notes in Artificial Intelligence LNAI, vol. 10501, pp. 149 – 159. Springer, Brasilia, Brazil (September 2017)
8. Giunchiglia, E., Tacchella, A., Giunchiglia, F.: Sat-based decision procedures for classical modal logics. Journal of Automated Reasoning **28**(2), 143–171 (2002)
9. Hansen, H.: Tableau games for coalition logic and alternating-time temporal logic–theory and implementation. Master's thesis, University of Amsterdam (2004)
10. Lavendhomme, R., Lucas, T.: Sequent calculi and decision procedures for weak modal systems. Studia Logica **65**, 121–145 (2000)
11. Lellmann, B.: Countermodels for non-normal modal logics via nested sequents. In: Bezhanishvili, N., Venema, Y. (eds.) SYSMICS2019 - Booklet of Abstracts, pp. 107–110. Institute for Logic, Language and Computation University of Amsterdam (2019)
12. Negri, S.: Proof theory for non-normal modal logics: The neighbourhood formalism and basic results. IfCoLog J. Log. Appl **4**(4), 1241–1286 (2017)
13. Olivetti, N., Pozzato, G.L.: CondLean: A theorem prover for conditional logics. In: Mayer, M.C., Pirri, F. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods - Proceedings of TABLEAUX 2003 (12th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods). Lecture Notes in Artificial Intelligence LNAI, vol. 2796, pp. 264 – 270. Springer, Roma, Italy (September 2003)
14. Olivetti, N., Pozzato, G.L.: CondLean 3.0: Improving Condlean for stronger conditional logics. In: Beckert, B. (ed.) Automated Reasoning with Analytic Tableaux

and Related Methods - Proceedings of TABLEAUX 2005 (14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods). Lecture Notes in Artificial Intelligence LNAI, vol. 3702, pp. 328 – 332. Springer, Koblenz, Germany (September 2005)

15. Olivetti, N., Pozzato, G.L.: Theorem proving for conditional logics: CondLean and GoalDuck. Journal of Applied Non-Classical Logics **18**, 427 – 473 (2008)
16. Olivetti, N., Pozzato, G.L.: NESCOND: an implementation of nested sequent calculi for conditional logics. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Proceedings of IJCAR 2014 (7th International Joint Conference on Automated Reasoning). Lecture Notes in Artificial Intelligence LNAI, vol. 8562, pp. 511 – 518. Springer, Vienna (Austria) (July 2014)
17. Olivetti, N., Pozzato, G.L.: Nested sequent calculi and theorem proving for normal conditional logics: The theorem prover NESCOND. Intelligenza Artificiale **9**, 109 – 125 (2015)
18. Pacuit, E.: Neighborhood semantics for modal logic. Springer (2017)
19. Pauly, M.: A modal logic for coalitional power in games. Journal of Logic and Computation **12**(1), 149–166 (2002)
20. Ross, A.: Imperatives and logic. Theoria **7**, 53–71 (1941)
21. Vardi, M.Y.: On epistemic logic and logical omniscience. In: Theoretical aspects of reasoning about knowledge. pp. 293–305. Elsevier (1986)