DefTab: A Tableaux System for Sceptical Consequence in Default Modal Logics

Carlos Areces[1], Valentin Cassano[12], Raul Fervari[13], and Guillaume Hoffmann[13]

[1] CONICET and Universidad Nacional de Córdoba, Argentina
[2] Universidad Nacional de Río Cuarto, Argentina
[3] Guangdong Technion - Israel Institute of Technology, China

Abstract. We report on an implementation of a tableaux calculus for sceptical consequence in Default Logic built on Hybrid Modal Logic. In turn, our tool offers support for checking default consequence over formulas from Propositional Logic, Basic Modal Logic and Hybrid Logic. We develop a test suite for assessing the correctness, scalability, and efficiency of our system, and inform on the results. Interestingly, our method can be adapted to generate examples for other default provers.

1 Introduction

A tableau method [11] is a standard proof procedure based on 'refutations'. To prove that a certain fact is valid, the procedure begins with a syntactical expression intended to assert the negation of the given fact. Then, successive steps syntactically break down this assertion into cases. Finally, impossibility conditions dictate closing cases. A proof is obtained if all cases are closed. Tableaux are one of the most popular proof calculi for Modal Logics, as they are known to lead to efficient and modular implementations [9].

The tableaux method presented here, called *default tableaux*, operates in the way just described. The novelty is that this tableaux method captures sceptical consequence in Default Logic [17], one of the most prominent approaches for non-monotonic reasoning [1]. Two distinguishing characteristics of a default logic are *defaults* and *alternative extensions*. Briefly, defaults can be understood as defeasible rules of inference, whereas extensions can be understood as sets closed under the application of defaults. Alternative extensions originate from 'consistency checks' on the application of defaults. A formula is called a 'sceptical consequence' if it is a consequence from every alternative extension. Our tableaux method handles sceptical consequence for $\mathscr{D}$HL, a default logic built over Hybrid Logic (HL) [3,4], via *default tableaux*. Default tableaux are introduced as an extension of tableaux for HL. These tableaux build on results presented in [5,7].

Moreover, we report on DefTab, an implementation of the default tableaux mentioned above. DefTab was originally conceived for checking sceptical consequence in Default Intuitionistic Logic [7]. Here, we advance on a modular implementation of a default prover acting over different modal logics. The general implementation of the tool is based on the architecture of HTab [13], a tableaux

system for HL (see also [12]). Given the ability of handling formulas from HL, our prover also supports formulas from fragments of HL such as Classical Propositional Logic and Basic Modal Logic. Each fragment is in itself interesting.

We discuss the overall architecture of DefTab, the implementation of default tableaux algorithm, and optimization details. In addition, we present an empirical evaluation of the tool to assess its correctness and efficiency. To this end, we build a test suite for sceptical consequence in $\mathscr{D}$HL by using hGen [2], a random formula generator for HL and the mentioned fragments. We provide a systematic method to convert formulas generated by hGen into interesting test cases for $\mathscr{D}$HL. We posit other provers could benefit from our method in the future.

2 Basic Definitions

Hybrid Logic. The language of HL is defined on an enumerable set $\mathscr{P} = \{ p_i \mid 0 \leq i \}$ of *proposition symbols* and an enumerable set $\mathscr{N} = \{ n_i \mid 0 \leq i \}$ of *nominals*, and is determined by the following BNF:

$$\varphi ::= p_i \mid n_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box\varphi \mid @_{n_i}\varphi \mid \mathsf{A}\varphi.$$

Other Boolean connectives are defined as usual. The modal formula $\Diamond\varphi$ is an abbreviation for $\neg\Box\neg\varphi$, whereas $\mathsf{E}\varphi$ abbreviates $\neg\mathsf{A}\neg\varphi$. We will also refer to some fragments of HL: the Basic Hybrid Logic (HL^-) is obtained by removing the constructor $\mathsf{A}\varphi$ from the BNF above. The Basic Modal Logic (BML) is obtained by additionally removing n_i and $@_{n_i}\varphi$ from the BNF. Finally, the Classical Propositional Logic (CPL) is obtained by additionally removing $\Box\varphi$.

A hybrid Kripke model $\mathfrak{M}$ is a tuple $\langle W, R, V\rangle$ where: W is a non-empty set of elements called worlds; $R \subseteq W^2$ is the accessibility relation; and the valuation $V : \mathscr{P} \cup \mathscr{N} \mapsto 2^W$ is a function s.t. for all $n \in \mathscr{N}$, $|V(n)| = 1$.

The notion of satisfiability, written $\mathfrak{M}, w \models \varphi$, is defined inductively as follows, with the Boolean cases defined as usual:

$$\begin{array}{ll}
\mathfrak{M}, w \models p_i & \text{iff } w \in V(p_i) \\
\mathfrak{M}, w \models n_i & \text{iff } \{w\} = V(n_i) \\
\mathfrak{M}, w \models \Box\varphi & \text{iff for all } w' \in W,\ Rww' \text{ implies } \mathfrak{M}, w' \models \varphi \\
\mathfrak{M}, w \models \mathsf{A}\varphi & \text{iff for all } w' \in W,\ \mathfrak{M}, w' \models \varphi \\
\mathfrak{M}, w \models @_{n_i}\varphi & \text{iff } \mathfrak{M}, w' \models \varphi, \text{ where } \{w'\} = V(n_i).
\end{array}$$

We write $\mathfrak{M}, w \models \Phi$ to abbreviate: for all $\varphi \in \Phi$, $\mathfrak{M}, w \models \varphi$. We call φ a *(local) semantic consequence* ([3]) of Φ, notation $\Phi \vDash \varphi$, iff for every hybrid Kripke model $\mathfrak{M}$, and world w of $\mathfrak{M}$, if $\mathfrak{M}, w \models \Phi$, then $\mathfrak{M}, w \models \varphi$.

Normal Default Logic. The work on *Default Logic*, initiated in [17], comprises nowadays a wide range of non-monotonic formalisms built on an underlying (typically monotonic) logic. In what follows, we describe a default logic built on HL, and call this default logic Default Hybrid Logic ($\mathscr{D}$HL).

$\mathscr{D}$HL is characterized by *normal defaults* and *extensions*. A normal default is a pair (π, χ) of formulas of HL written as π/χ; where π is called the prerequisite of the default, and χ its consequent. A normal default can be understood as a

non-admissible rule of inference of HL which is only applied if its application does not yield a contradiction. Normal defaults are common in the literature, since interestingly most existing variants of Default Logic converge in the case of normal defaults (see, e.g., [1]). Extensions are defined with respect to default theories. A default theory is a pair $\Theta = \langle \Phi, \Delta \rangle$ where: Φ is a set of formulas of HL, also indicated by Φ_Θ; and Δ is a set of *normal defaults*, also indicated by Δ_Θ. An extension can be understood as a saturation of a set of facts via the application of defaults. The precise definition of an extension is given in Def. 4.

Definition 1. *Let $\delta = \pi/\chi$ be a default and Δ be a set of defaults; then: $\delta^\Pi = \pi$, $\delta^\mathrm{X} = \chi$; $\Delta^\Pi = \{ \delta^\Pi \mid \delta \in \Delta \}$, $\Delta^\mathrm{X} = \{ \delta^\mathrm{X} \mid \delta \in \Delta \}$ and $\Delta \cup \delta = \Delta \cup \{\delta\}$.*

Definition 2 (Detachment). *Let Θ be a default theory, and $\Delta \cup \delta \subseteq \Delta_\Theta$; we say that δ is* triggered *by Δ (in Θ) iff $(\Phi_\Theta \cup \Delta^\mathrm{X}) \models \delta^\Pi$. We say that δ is* blocked *by Δ iff $(\Phi_\Theta \cup (\Delta \cup \delta)^\mathrm{X}) \models \bot$. We say that δ is* detached *by Δ if δ is triggered, and not blocked, by Δ.*

If we think of a default π/χ as a rule which enables us to pass from π to χ, the notion of detachment in Def. 2 tells us under which conditions on π we can obtain χ. The definition of detachment is an intermediate step towards the definition of an extension via generating sets.

Definition 3 (Generating Set). *Let Θ be a default theory; we call $\Delta \subseteq \Delta_\Theta$ a* generating set *if there is a total-ordering $\lessdot$ on Δ_Θ s.t. $\Delta = \mathsf{D}^{\lessdot}_\Theta(n)$, where $n = |\Delta_\Theta|$, $\mathsf{D}^{\lessdot}_\Theta(0) = \emptyset$, and for all $0 < i < n$:*

$$\mathsf{D}^{\lessdot}_\Theta(i{+}1) = \begin{cases} \mathsf{D}^{\lessdot}_\Theta(i) \cup \delta & \text{if } \delta \in \Delta_\Theta \backslash \mathsf{D}^{\lessdot}_\Theta(i) \text{ is detached by } \mathsf{D}^{\lessdot}_\Theta(i), \text{ and} \\ & \quad \text{for all } \eta \neq \delta \in \Delta_\Theta \backslash \mathsf{D}^{\lessdot}_\Theta(i), \text{ if } \eta \text{ is detached by } \mathsf{D}^{\lessdot}_\Theta(i), \ \delta \lessdot \eta \\ \mathsf{D}^{\lessdot}_\Theta(i) & \text{otherwise.} \end{cases}$$

Definition 4 (Extension). *Let Θ be a default theory and $E = \Phi_\Theta \cup \Delta^\mathrm{X}$; the set E is an extension of Θ iff Δ is a generating subset of Δ_Θ. We use $\mathscr{E}(\Theta)$ to indicate the set of all extensions of Θ.*

As mentioned, intuitively, an extension is a set of formulas that is closed under detachment. We present the definition of default consequence in Def. 5.

Definition 5 (Default Consequence). *We say a formula φ is a* sceptical consequence *of a default theory Θ, notation $\Theta \mathrel{|\!\sim} \varphi$, iff for all $E \in \mathscr{E}(\Theta)$, $E \models \varphi$.*

The notion of default consequence in Def. 5 is referred to as sceptical in the literature on Default Logic. In Sec. 3 we present a syntactic characterization of sceptical consequence via a default tableaux proof calculus. This proof calculus is the focus of our system description. We illustrate our definitions in Ex. 1.

Example 1. We start by assuming that every world in the model has a successor, and that every world is either a *sink* world (nominal s) or 'sees' the sink world. These assumptions are expressed in a default theory as facts, i.e., by

$\Phi = \{\mathsf{A}\Diamond\top, \mathsf{A}(s \vee \Diamond s)\}$. Moreover, we have three defaults: $\delta_1 = \top/@_{n_2}\Diamond n_3$, $\delta_2 = \top/@_{n_3}\neg s$, and $\delta_3 = \top/@_{n_3}\Box n_3$. Thus, we have $\Delta = \{\delta_1, \delta_2, \delta_3\}$, and $\Theta = \langle \Phi, \Delta \rangle$. The default δ_1 expresses that n_2 must 'see' n_3. This default is detached by Φ. Then, we have the defaults δ_2, expressing that n_3 must not be the sink world, and δ_3, expressing that n_3 must only 'see' itself. Both of these defaults are individually detached by δ_1, but they block each other: δ_2 forces n_3 to have a successor different from itself to comply with the facts, while δ_3 forces n_3 to see only itself, i.e., it forces n_3 be the sink. This means that we have two generating sets, $\{\delta_1, \delta_2\}$ and $\{\delta_1, \delta_3\}$, thus there are two extensions: $E_1 = \Phi \cup \{@_{n_2}\Diamond n_3, @_{n_3}\neg s\}$ and $E_2 = \Phi \cup \{@_{n_2}\Diamond n_3, @_{n_3}\Box n_3\}$. In both cases, n_2 sees the sink in two steps, i.e., $\Theta \mathrel{|\!\sim} @_{n_2}\Diamond\Diamond s$.

3 Default Tableaux Proof Calculus

We present the default tableaux calculus for sceptical consequence in $\mathscr{D}\mathsf{HL}$ which is the focus of our system description. In what follows, we consider all the formulas from HL in *negation normal form*. The default tableaux calculus for sceptical consequence in $\mathscr{D}\mathsf{HL}$ constructs so-called *default tableaux*. A default tableau is a tree whose nodes are of three different kinds. We write nodes of the first kind as $@_i\varphi$, meaning that φ holds at world i. The second kind of nodes (which is a special case of the first kind) is written as $@_i\Diamond j$, meaning that world j is accessible from world i. Nodes of the third kind are indicated by defaults. This last kind of nodes marks the use of a default in a proof attempt. A default tableau for a formula φ from a default theory Θ, is a default tableau whose root is $@_0\neg\varphi$, and whose construction is carried out using the rules from Fig. 1.

$$\frac{@_i(\varphi \wedge \psi)}{@_i\varphi,\ @_i\psi}\ (\wedge) \qquad \frac{@_i\Diamond\varphi}{@_i\Diamond j,\ @_j\varphi}\ (\Diamond)^1 \qquad \frac{@_i@_a\varphi}{@_a\varphi}\ (@) \qquad \frac{@_i\mathsf{E}\varphi}{@_j\varphi}\ (\mathsf{E})^1$$

$$\frac{@_i(\varphi \vee \psi)}{@_i\varphi \mid @_i\psi}\ (\vee) \qquad \frac{@_i\Box\varphi,\ @_i\Diamond j}{@_j\varphi}\ (\Box) \qquad \frac{@_i\varphi,\ @_i j}{@_j\varphi}\ (nom)^2 \qquad \frac{@_i\mathsf{A}\varphi}{@_j\varphi}\ (\mathsf{A})^2$$

$$\frac{}{@_j j}\ (\mathrm{ref})^2 \qquad \frac{}{@_0\varphi}\ (\mathsf{F})^3 \qquad \frac{}{\begin{array}{ccccc} \delta_1 & & \delta_i & & \delta_n \\ @_0\delta_1^{\mathrm{X}} & \ldots & @_0\delta_i^{\mathrm{X}} & \ldots & @_0\delta_n^{\mathrm{X}} \end{array}}\ (\mathsf{D})^4$$

[1] The nominal j is new to the branch.
[2] The nominal j is already in the branch.
[3] For $\varphi \in \Phi_\Theta$.
[4] For $\{\ \delta_i \mid i \in [1, n]\ \} = \{\ \delta \in \Delta_\Theta \backslash \Delta_B \mid \delta$ is detached by $\Delta_B\ \}$, where Δ_B is the set of defaults in the branch.

Fig. 1. Tableau expansion rules for $\mathscr{D}\mathsf{HL}$.

The rule (F) enables us to incorporate formulas from Φ_Θ into a default tableau, while the rule (D) enables us to incorporate defaults from Δ_Θ. This

last rule corresponds to the concept of detachment in Def. 2. The notion of deducibility using default tableaux is made precise in Def. 7.

Definition 6 (Closure). *A branch of a default tableau is* closed (▲)*, if* $@_i\varphi$ *and* $@_i\neg\varphi$ *occur in the branch. A branch is* open (▼) *if it is not closed. A default tableau is* closed *if all of its branches are closed; otherwise it is* open*.*

Definition 7 (Default Deducibility). *We call any closed default tableau for* φ *from* Θ *a* sceptical proof *of* φ *from* Θ*, notation* $\Theta \mathrel{|\!\sim} \varphi$*.*

The expansion rules in Fig. 1 together with Def. 7 yield a sceptical proof calculus which is is *sound* and *complete* (see [7] for details of this claim).

Theorem 1 (Soundess and Completeness.). $\Theta \mathrel{|\!\sim} \varphi$ *iff* $\Theta \approx \varphi$*.*

In addition, notice that if we forbid the application of the rule (D), we obtain a notion of deducibility $\Phi_\Theta \vdash \varphi$ which yields a sound and complete proof calculus for HL, i.e., $\Phi_\Theta \vdash \varphi$ iff $\Phi_\Theta \models \varphi$ (see [16]). We use $\vdash$ to syntactically check the side condition of the rule (D), and decide whether it can be applied or not.

Definition 8 (Saturation). *A branch of a default tableau is* saturated*, notation* (⧫)*, if the application of any of the expansion rules in Fig. 1 is redundant.*

It can be proven that every branch of a default tableau can be extended to one that is saturated in a finite number of steps. Also, if a default tableau for φ from Θ has a branch that is open and saturated, then $\Theta \not\mathrel{|\!\sim} \varphi$. From these two facts, it follows that default tableaux decide sceptical consequence.

4 Implementation

DefTab is an implementation of the tableaux proof calculus for sceptical default consequence in Sec. 3. The architecture of DefTab is based on the hybrid logic prover HTab [13], and incorporates the specific features for implementing default reasoning. HTab implements a terminating tableaux algorithm for HL and comes ready with some optimizations such as semantic branching and backjumping. All these features, as well as others, are reported in detail in [13]. Given Θ and φ as input, DefTab builds proof attempts of $\Theta \mathrel{|\!\sim} \varphi$ by searching for Kripke models for φ, and subsequently restricting these models with the use of sentences from Φ_Θ and defaults from Δ_Θ. DefTab reports whether a default proof has been found or not. In the latter case, it exhibits an extension of Θ from which the φ does not follow; thus establishing that φ is not a default consequence of Θ. In what follows we discuss some implementation details, including some comments on optimizations. DefTab is available at `http://tinyurl.com/deftab0`.

Tableaux and Subtableaux. The tableaux algorithm of DefTab follows a standard strategy for proof search, and the novel part is the treatment of the rule (D). In such a case, it selects a default δ from the set Δ_Θ, and checks if δ is detached, according to Def. 2. This relies on subtableaux, that is, tableaux executions that are independent of the main default tableaux. These subtableaux are needed to check whether δ is detached in the branch; i.e., whether it is triggered (i.e., δ^Π is a consequence of the premises and the consequences already obtained in the branch), and not blocked (i.e., if δ^X adds an inconsistency into the branch). If δ is detached, then $@_0\delta^X$ is added to the branch, δ is marked as treated, and the algorithm continues with the expansion of the updated branch. Once no rule can be applied, the algorithm returns TRUE if and only if φ is a default consequence of Θ.

Subtableaux caching. One of the main optimizations provided in DefTab is *caching*, operating under the following premise. Subtableaux are executed to check which default rules are triggered or blocked in the context of a branch. Many of these checks are redundant, since the results of such subtableaux does not change change unless a default rule is applied to a branch. DefTab implements a simple caching system that stores subtableaux results in a dictionary. Each time a subtableaux is about to be executed, the set of initial formulas is checked against the cache. If there is a cache hit, the result is taken from the cache and a tableaux run is saved. Note that subtableaux do not involve the rule (D), that is, they are purely tableaux of the underlying logic.

Default rules data structures. At any given moment, DefTab maintains defaults in two lists: *available* and *triggered*. The available list contains the defaults of the input default theory. When the (D) rule is about to be applied, several steps are performed to handle default rules systematically. First, the *available* list is scanned, and each rule is checked to be triggered. Triggered rules are moved into the *triggered* list, and the rest is left into the *available* list. Note that non-triggered rules, may become triggered in the future after some default is added to the branch. The *triggered* list is also scanned, and each rule is checked to be blocked in the current branch. When a rule is blocked, it is deleted from the *triggered* list and will never come back again in the branch. Once this is done, DefTab uses that list to apply the rule (D). The tableaux branches as many times as there are rules in the (non-blocked) *triggered* list. For each new branch, the procedure removes the corresponding rule from the *triggered* list, and adds it and its consequent formula to the branch.

Backjumping. Backjumping [14] is a standard optimization for the HL calculus that greatly improves performance (see [13]). The overall idea is that, instead of performing a simple backtracking when a branch is found to be closed, backjumping calculates the lowest level to which the execution of the tableaux may directly come back when a clash is found. This requires all formulas in the tableaux to be annotated with a set of *dependencies*. A dependency is the level of a branching rule application. For the specific case of default tableaux, we take

special care of tracking dependencies of the formulas introduced by the application of rule (D). To do so, once a default π/χ is triggered, we bookkeep it in the *triggered* list along with the dependencies of the formulas that triggered it, according to Definition 2. Concretely, this is the union of the dependencies of all defaults Δ such that $\Phi_\ominus \cup \Delta^{\mathrm{X}} \models \pi$. When (D) rule is applied, the consequent of a default is added to the current branch with these dependencies, plus the dependency of the current tableaux level.

Usage. DefTab takes as input a file following the structure of the following simple example file `hybrid01.dt`.

```
facts:
N0: <> N1;
defaults:
(N0: <>N1) --> (N1:<>N0);

consequence:
N0:<><>N0;
```

– The keyword `facts` indicates the beginning of the set of formulas of the default theory.
– The keyword `defaults` indicates the beginning of the set of defaults. The syntax for a default π/χ is π `-->` χ.
– The keyword `consequence` indicates the formula to be proven.

DefTab is executed from the command line as:

```
$ ./deftab -f hybrid01.dt
--------------------------------
Indeed a sceptical consequence.
Elapsed time: 0.00 seconds
```

The output indicates that `N0:<><>N0` ($@_{n_0}\Diamond\Diamond n_0$) is a sceptical consequence of the default theory.

5 Testing Generation and Methodology

Hybrid and Default Formulas Generation. Another contribution of our work is to provide a systematic way of constructing test cases for $\mathscr{D}$HL provers. To our knowledge, there is no standard test set for automated reasoning with default logic, and less so for default reasoning based on HL.

We build test cases for $\mathscr{D}$HL using the random formula generator hGen [2]. hGen enables us to generate formulas in conjunctive normal form (CNF) from several fragments of HL, such as CPL, BML and HL$^-$. Moreover, hGen also allows us to specify the different parameters of a formula: number of clauses, size of clauses and modal depths of each subformula of a clause, probability of that an operator appears in the clause (e.g. modal, hybrid, universal), and the total number of propositional symbols and nominals.

We adapted hGen to generate normal default theories from random HL formulas. The transformation depends on the satisfiability status of the original HL formulas. The first case applies to satisfiable formulas of HL in CNF. Given $c_1 \ldots c_n$ the clauses of an HL formula, we put each one of them as the consequent

of a default $\top/c_i$, and put $\bot$ as the consequence to be proved. As the original set of clauses is satisfiable, and the consequence is never provable, all the defaults will be applied (as putting $\top$ as the prerequisite triggers every rule) in all possible permutations. This is an easy way to stress our tool.

The second case works with unsatisfiable formulas of HL in CNF. Here, we use an intentionally harder transformation. Given $c_1 \dots c_n$ the clauses of the HL formula, then for all $i<n$, we generate two rules: $\top/c_i \vee c_{i+1}$ and $c_i \vee c_{i+1}/c_i \wedge c_{i+1}$. Finally, we add c_n as consequence. In this case, not all defaults will be applied to a same branch, but a great amount of them. Moreover, the formula c_n may or may not be a sceptical consequence of the default theory; this is another difference with the case of satisfiable formulas. This case not only serves to test the scalability of our tool, but also its correctness.

Test Suite Structure. The Bash script `testsuite.sh` executes four steps: formula generation, renaming, benchmark, and consistency check.

The *formula generation* step uses hGen to generate random sets of formulas from CPL, BML, HL^- and HL, respectively. Initially, each set contains 1000 formulas. Then, the Hybrid Logic prover HTab ([13]) is run to classify each set of formulas into satisfiable (SAT) and unsatisfiable (UNSAT). This way, hGen generates the corresponding default theories, as described in the previous section. The *renaming* step is then performed to organize file names in each folder.

The *benchmark* step enables to specify a list of provers to be run. Currently, it is performed with DefTab with cache disabled (NC) and DefTab with cache enabled (C), but the script can be easily modified to run any new default prover. The provers are executed on all input files of each combination of 4 languages and 2 satisfiability values, and the results (execution time and answer) are stored in log files. The script reports how many formulas could be solved within 10 seconds, 30 seconds, and 60 seconds. This is done by running the provers with the highest timeout value; the other values are deduced from the prover's running time.

Finally, the *consistency check* step looks for inconsistent outputs between provers by comparing the log files generated in the previous step.

Although the preselected option is to run all these steps together, they can also be run separately. This enables to run the benchmark step on a known set of formulas, to reproduce results. Instructions on how to run the tests, the test script and the set of formulas used to generate the following results can be found at http://tinyurl.com/deftab0.

hGen parameters. For each language, we tuned hGen's parameters to get a good SAT/UNSAT balance of its output (ideally a 50/50 ratio). We also aimed at getting a balanced difficulty of the translated default theories. That is, the sets of default theories should be hard enough so that many of them make DefTab timeout and we may measure improvements in the future, but not too hard so we can already observe different results according to different timeout values. The parameters for each language are: for CPL, 33 clauses and 10 proposition symbols; for BML, 34 clauses, 10 proposition symbols, one relation and 2 nested modal operators as maximum; for HL^-, 15 clauses, 3 proposition symbols, 3

nominals, one relation and 6 nested modal and hybrid operators as maximum; and for HL, 13 clauses, 2 proposition symbols, 2 nominals, one relation and 6 nested modal, hybrid and universal operators as maximum. Moreover, each language has fine-tuned probabilities of the different logic connectives in order to meet the SAT/UNSAT and timeout balances that the following results show. All parameters can be found in the released test script.

Results. We report below a run of the benchmark script with 1000 formulas per language, performed with DefTab with cache disabled (NC) and DefTab with cache enabled (C). DefTab was compiled with GHC 8.10.7, and the tests were run on the following platform: Ubuntu 22.04 operating system, Linux 5.19 kernel, 12th Gen Intel i7-1260P CPU with 16 cores, 16GB of RAM and SSD storage.

Formulas \ Timeout	10 secs. (NC)	10 secs. (C)	30 secs. (NC)	30 secs. (C)	60 secs. (NC)	60 secs. (C)
CPL SAT (516)	122	135	133	144	138	146
CPL UNSAT (484)	255	324	309	364	336	384
BML SAT (462)	356	399	384	417	398	425
BML UNSAT (538)	154	193	252	324	295	367
HL^- SAT (534)	401	434	419	444	431	450
HL^- UNSAT (466)	142	153	150	170	158	183
HL SAT (480)	284	321	309	331	320	343
HL UNSAT (520)	145	161	161	183	169	193

Finally, the following table describes the outcome of checking sceptical consequence of those formulas that were originally unsatisfiable. We take therein all the tests cases that finished with timeout of 60 seconds, solved using caching. The column label by 'Consequence' indicates the number of formulas for which running DefTab returns it is indeed a sceptical consequence in the corresponding default theories; while 'Not Consequence' indicates the number of formulas for which DefTab returns they are not a sceptical consequence.

Formulas \ Results	Total	Consequence	Not Consequence
CPL UNSAT	384	24	360
BML UNSAT	367	322	45
HL^- UNSAT	183	111	72
HL UNSAT	193	100	93

These results are useful for checking consistency across the execution of different provers, or provers executed with different parameters, as we are currently doing with DefTab's cache option. Moreover, we would like to compare the obtained data with the results of running other provers for the different fragments that are supported by DefTab, to assess both soundness and the performance of our tool. This is part of our future work agenda.

6 Final Remarks

We reported on DefTab, a tableaux-based system to decide sceptical consequence in Default Logic over Hybrid Modal Logic. To the best of our knowledge, DefTab is the first prover combining Modal and Default Logic. This said, other provers do exist for Default Logic. For instance, DeReS is a default logic reasoner with an underlying propositional tableaux calculus [8]. This prover is designed to check default consequence treating reasoning in the underlying logic as a "black box". This contrasts with DefTab which extends tableaux reasoning in the underlying logic with the use of defaults. At present, DefTab only supports sceptical consequence checking, while DeReS also supports credulous consequence checking. We have not been able to find a working implementation of DeReS. However, many of the ideas presented in [8] can be explored in our setting, in particular, the kind of (graph-based) problems that are used to generate test cases.

Although not a default logic reasoner, in [15], a nonmonotonic reasoning plug-in for OWL ontologies is presented. DefTab could approach this tool by implementing multiple relations (roles) and role inclusions to its underlying modal language. In [10] a tool supporting default reasoning over knowledge bases is reported, this time not via a calculus implementation but via a translation into conjunctive query programs in a Description Logic reasoner. After adapting our calculus to handle Description Logic features, it would be interesting to use the above-mentioned tools to perform a comparison with DefTab, both for correctness and performance.

We provided a systematic way of testing our tool, by introducing a test suite generation method based on hGen [2] and HTab [13,12]. This idea can be easily adapted to any kind of default prover working over CPL, BML, HL^- and HL. We tested the performance of our tool using this test suite, and empirically showed that DefTab's *subtableaux caching* optimization positive impacts on performance.

For future work there are several other interesting lines of research. The treatment of defaults in the calculus can be seen as parametric on the underlying logic (modulo some basic properties, e.g., the possibility of using premises, see [6]). DefTab was originally designed to handle Default Logic over Intuitionistic Logic [7]. Herein, the tableaux-based procedure not only handles classical reasoning instead of intuitionistic reasoning, but also it is extended to support a family of Modal Logics (i.e., the fragments we described along the paper). Moreover, our approach allowed us to design test suites that can be used to test DefTab and other nonmonotonic provers. These ideas can be extended to better assess the behaviour of the tools. We believe that our implementation is a first step towards having a modular prover that can be generalized to a wider family of Default Logics.

Acknowledgments. We thank the reviewers for their valuable comments. Our work is partially supported by the projects ANPCyT-PICT-2020-3780, ANPCyT-PICT-2021-00400, CONICET PIP 11220200100812CO, the EU Grant Agreement 101008233 (MISSION), and by the Laboratoire International Associé SINFIN.

References

1. G. Antoniou and K. Wang. Default logic. In D. Gabbay and J. Woods, editors, *The Many Valued and Nonmonotonic Turn in Logic*, volume 8 of *Handbook of the History of Logic*, pages 517–555. North-Holland, 2007.
2. C. Areces and J. Heguiabehere. hgen: A random cnf formula generator for hybrid languages. In *Methods for Modalities 3 - M4M-3, Nancy, France*, Nancy, France, 2003.
3. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge U Press, 2001.
4. P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*. Elsevier, 2007.
5. V. Cassano, C. Areces, and P. Castro. Reasoning about prescription and description using prioritized default rules. In G. Barthe, G. Sutcliffe, and M. Veanes, editors, *22nd International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-22)*, volume 57 of *EPiC Series in Computing*, pages 196–213. EasyChair, 2018.
6. V. Cassano, R. Fervari, C. Areces, and P. Castro. Interpolation and beth definability in default logics. In F. Calimeri, N. Leone, and M. Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *LNCS*, pages 675–691. Springer, 2019.
7. V. Cassano, R. Fervari, G. Hoffmann, C. Areces, and P. Castro. A tableaux calculus for default intuitionistic logic. In P. Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *LNCS*, pages 161–177. Springer, 2019.
8. P. Cholewinski, V. Marek, and M. Truszczynski. Default reasoning system deres. In *5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 518–528. Morgan Kaufmann, 1996.
9. M. D'Agostino, D. M. Gabbay, R. Hahnle, and J. Posegga, editors. *Handbook of Tableau Methods*. Springer, 1999.
10. M. Dao-Tran, T. Eiter, and T. Krennwallner. Realizing default logic over description logic knowledge bases. In C. Sossai and G. Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 602–613, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
11. M. Fitting. Introduction. In D'Agostino et al. [9], pages 1–43.
12. G. Hoffmann. Lightweight hybrid tableaux. *Journal of Applied Logic*, 8(4):397–408, 2010.
13. G. Hoffmann and C. Areces. HTab: a terminating tableaux system for hybrid logic. In C. Areces and S. Demri, editors, *Proceedings of the 5th Workshop on Methods for Modalities, M4M 2007, Cachan, France, November 29-30, 2007*, volume 231 of *ENTCS*, pages 3–19. Elsevier, 2007.
14. U. Hustadt and R. Schmidt. Simplification and backjumping in modal tableau. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX'98)*, volume 1397 of *LNCS*, pages 187–201. Springer, 1998.
15. T. Meyer, K. Moodley, and U. Sattler. DIP: A defeasible-inference platform for OWL ontologies. CEUR Workshop Proceedings, 2014.
16. G. Priest. *An Introduction to Non-classical Logic: From If to Is*. Cambridge U Press, 2000.
17. R. Reiter. A logic for default reasoning. *AI*, 13(1-2):81–132, 1980.