Design and Results of TANCS-2000 Non-Classical (Modal) Systems Comparison*

Fabio Massacci[1] and Francesco M. Donini[2]

[1] Dip. di Ingegneria dell'Informazione — Università di Siena
Via Roma 56, 53100 Siena, Italy — massacci@dii.unisi.it
[2] Dip. di Elettrotecnica ed Elettronica — Politecnico di Bari
Via Re David 200, 70125 Bari, Italy — donini@poliba.it

Abstract. The aim of the TABLEAUX-2000 Non-Classical (Modal) System Comparisons (TANCS-2000) is to provide a set of benchmarks and a standardized methodology for the assessment and comparison of ATP systems in non-classical logics, as it is done for first-order logic with the CADE System Competition. We believe that TANCS can benefit the scientific community in two ways: by promoting the competition among ATP systems and thus yielding novel solutions, and by providing a scientific design for benchmarking non-classical ATP systems.
This paper reports the main ideas behind the design, the benchmarks, the organization, and the rating of the ATP systems of TANCS-2000.

1 Design and Organization of the Comparison

The first comparison has been held in 1998 [1], a second one has been held in 1999 [11] and this one continues the series with a focus on expressive modal and description logics (for an introduction to modal logics see [7], for description logics see [3]). The following Automated Theorem Proving (ATP) systems have been submitted this year: MSPASS (based on resolution), ⋆SAT, FaCT, DLP and RACE (based on various optimization of tableaux and Davis-Puntam procedures). Their descriptions can be found in these proceedings.

As in past years, they have been compared along two yardsticks: *effectiveness* and *usability*. Effectiveness can be measured on the basis of the type and number of problems solved, the average runtime for successful solutions, the scaling of the prover as problems gets bigger. Usability can be assessed on the basis of availability via web or other sources, portability to various platforms, need for additional software besides the prover itself, ease of installation and use (e.g., visual interfaces), possibility of customizing the search heuristics, etc.

For what regard effectiveness, a sensitive decision is the *choice of benchmark problems* which should offer the possibility to generate enough different samples

* More details are at http://www.dis.uniroma1.it/~tancs. We thank S. Demri for its valuable contribution, P. Patel-Schneider for beta-testing some benchmarks and all TANCS participants for their constructive suggestions. TANCS problem repository is hosted by the Dipartimento of Informatica e Sistemistica, Università di Roma "La Sapienza", ITALY. Work supported by MURST (project MOSES) and CNR.

so that "benchmark-tailored" techniques will not work, and which are either representative of the difficulties of the underlying satisfiability decision problem or representative of some real-world case.

Another key decision is the *rating of the systems* which cannot just rely on raw running times nor on internal aspects of the algorithm (e.g. Davis-Putnam calls) as we may end up with the impossibility of comparing in any fair way the performance of ATPs using different hardware, operating systems and calculi.

2 Benchmark Problems

Benchmarks are grouped into main divisions and then into categories, as in the CADE System Competition [17], according the complexity of the corresponding decision problem[1]: a modal PSPACE division, a multi-modal PSPACE one, a global PSPACE division, a modal EXPTIME division.

For each category within a division there are few *reference problems* which every entrant of the comparison has to try. Then a *C program* can generate all random instances of one's size and choice, as in '99 [11]. Besides parameters such as numbers of clauses and variables, the C program makes it possible to choose between a "plain" version of the benchmark and "modalized" one which hides away propositional reasoning. For details see [11].

The basic benchmark is Unbounded Modal QBF. It has been first proposed for TANCS in [11] and its intuition is to encode validity of Quantified Boolean Formulae (QBF) into satisfiability in modal logic K.

In practice we generate a QBF with c clauses, alternation depth equal to d, with at most v variables for alternation. Setting $d = 3$ and $v = 2$, we can generate a QBF like $\forall v_{32} v_{31}.\exists v_{22} v_{21}.\forall v_{12} v_{11}.\exists v_{02} v_{01}.cnf_{c\text{-clauses}}(v_{01} \dots v_{32})$. For each clause we randomly generate k different variables (default 4) and each is negated with probability 0.5. The first and the third variable (if it exists) are existentially quantified, whereas the second and fourth variable are universally quantified. This aims at eliminating trivially unsatisfiable formulae [2]. Other literals are either universal or existentially quantified variables with probability 0.5. The depth of each literal is randomly chosen from 1 to d.

The QBF formula can be translated into modal logic with different encodings:

1. An optimization of Ladner's original translation [10] that does not introduce new variables but still contains many formulae which guarantees that a tree-like model is constructed following the alternating quantifier prefix;
2. a further optimized translation in which the formulae corresponding to the alternation depth are somewhat compiled away;
3. a yet more optimized translation which is fairly close to Schmidt-Schauss and Smolka's reduction of QBF validity into $\mathcal{ALC}$ satisfiability [15].

[1] We recall that deciding modal logic satisfiability is PSPACE complete and EXPTIME-complete if one uses global axioms Fitting-style [4]. However, not necessarily every benchmark set is able to capture these complexity classes.

For every fixed valued of d we can capture hard problems at the $d+1$-th level of the polynomial hierarchy in the same way that is done for 3-SAT problems [16]: we fix the ratio clauses over variables to some value[2] where a phase transition SAT/UNSAT occurs and increase the number of variables. That's better than [6] as we move upward in the complexity chain, yet PSPACE can only be reached by an unbounded and always increasing value of d (but then we can set v=1).

Wrt TANCS past edition, this benchmark has been enhanced to accommodate more expressive modal constructs: benchmark problems generated from QBF have been also encoded using the *inverse* operator on modalities (aka converse in $\mathcal{ALCI}$) and *sequence, union and star* of Converse Propositional Dynamic Logic (or their description logic equivalent). The basic idea behind the encodings is to replace a single relation in modal logic K by a sequence of back-forth-back relations, in some clever way to avoid making most formulae unsatisfiable.

Finally, the benchmark Periodic Modal CNF can capture PSPACE. It encodes periodic satisfiability problems [13] with global axioms Fitting-style [4]. We refer to [11] for further details on this encoding.

3 Usability of Submitted Systems

Since TANCS deals primarily with ATP systems, we considered only usability issues [12] which could be reasonable for ATP. Before the systems were submitted, we envisaged three main characteristics in evaluating usability.

Additional software (beside standard compilers). Two systems needed only a standard C compiler (MSPASS and ⋆SAT), DLP needed a free software (New Jersey ML), and two systems (FaCT and RACE) needed proprietary software (Allegro Common Lisp, available together with patches through `www.franz.com`). Of the two systems requiring no additional software, MSPASS was the only system we correctly installed on a Linux-based PC and Solaris-based SUN, without any interaction with their creators, while ⋆SAT needed a special *make*. For MSPASS and FaCT we needed to contact the authors to obtain the translator from the ATP input syntax to the TPTP syntax (and also hack down the translator).

User interface. Every system provided only a bare command-line interface, with no help available but a "readme" file. This was not a problem, since TANCS deals with theorem provers, not complete systems. However, while a graphical interface would have been clearly a surprise, we believe that a simple textual menu interface might have been helpful to promote usage of these systems.

Proof management. By this, we mean the ability of a system to provide the (even bare) steps followed to reach the conclusion. No submitted system provided us by default the proof of the results it obtained. MSPASS can be set to provide some proof — either a refutation or a saturated set of clauses.

We also considered the level of expertise required to run the system. Of course, we did not expect naive users to be able to run the systems; however, it turned out that because of the characteristics of user interfaces, and additional

[2] The ratio may depend on the number of variables and may not just be a constant as in 3-SAT. See also [2].

Table 1. Benchmark Results on Unbounded Modal QBF

Absolute Running Time in 10msec (Geometric Mean)

	DLP		FaCT+		MSPASS		⋆SAT		RACE	
Benchmark	Time	%Tout	Time	%Tout	Time	%Tout	Time	%Tout	Time	%Tout
K4-C45-V8-D4	80	-	237	-	2453	-	8661	2%	6136	80%
K4-C45-V8-D5	781	-	5236	-	5187	-	38222	41%	9257	81%
K4-C55-V8-D6	554	-	2842	-	10612	-	56135	64%	9509	89%

Normalised Running Time (Geometric Mean)

	$\boxed{\text{MSPASS}}$			DLP			FaCT+			QBF solver		
Benchmark	Tot.	sat	unsat	Tot.	sat	unsat	Tot.	sat	unsat	Tot.	sat	unsat
K4-C45-V8-D4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
K4-C45-V8-D5	2.1	2.1	2.1	9.8	15.1	6.4	22.1	32.1	15.2	1.3	2.3	0.7
K4-C55-V8-D6	4.3	4.3	4.3	7.0	9.1	6.1	12.0	19.9	9.3	1.2	1.1	1.3

software required, no system could have been run by an expert of the problem itself — namely, either an expert of description logics and KRSS, or an expert in modal logics and the TPTP syntax. More expertise about the architecture of systems, operating systems compatibilities, and even shell languages was needed.

4 Effectiveness and Performance Analysis

The *normalized running time* is the yardstick used to compare provers. In a nutshell, for every prover we compute the geometric mean time on some reference problems and then normalise the run time of each comparison problem with respect to (i.e. divide by) this reference mean time. Then we obtain a relative ranking which makes it possible to abstract away, at least to a certain extent, machine- and run-dependent characteristics. Notice that the geometric mean time must be used, otherwise we may draw meaningless conclusions [5]. Ability of handling large instances and asymptotic behavior emerge more clearly [8, 9].

A compact report of the comparison on some Unbounded Modal QBF problems (using the encoding 3 and composed by approximately 50% of SAT and UNSAT instances) is presented in Table 1 (more details are in the web pages).

The first table reports the absolute running time as submitted by the entrants and the corresponding timeouts. Out of this table, we might be tempted to conclude that DLP is the fastest. But, as we said, this might depend on the compiler, the software etc. The second table shows the normalized time and a new picture: MSPASS is TANCS-00's leader on Unbounded Modal QBF, closely followed by DLP and FaCT+ (a version of FaCT with caching enabled). The last entry is a QBF-solver [14] which solved all problems within a second: modal and description logics ATP systems have still far to go.

DLP and FaCT time increases and then decreases as expected: they are tableau based and satisfiable instances are harder at even levels of $d + 1$ [2].

Extensibility. We want also to mention some other features of the systems, as dealing with more expressive logics may be worth their slow down. All systems can reason in the basic multi-modal logic K, aka the description logic $\mathcal{ALC}$. Except ⋆SAT, all systems can deal with transitive roles (full transitive closure of roles for DLP), i.e., multi-modal logic $\mathsf{S4}$, and global axioms (aka general TBoxes) and submitted results in the corresponding category. FaCT and MSPASS can deal with inverse modalities and both submitted results using inverse (MSPASS also reported about union and sequential composition).

For what regards problems not among TANCS's benchmarks (yet), FaCT and RACE can reason with qualified number restrictions, aka graded modalities. RACE can also handle ABoxes, which correspond to a restricted use of nominals in modal logics. Finally, we note that MSPASS could, in principle, deal with any feature that can be expressed within first-order logic; its running behavior, though, cannot be predicted for such extensions.

References

1. P. Balsinger and A. Heuerding. Comparison of theorem provers for modal logics. In *Proc. of TABLEAUX-98*, *LNAI* 1397, p. 25–27, Springer Verlag 1998.
2. M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified boolean formulae. In *Proc. of AAAI-98*, 1998.
3. F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In *Foundation of Knowledge Representation*, p. 191–236. CSLI-Publications, 1996.
4. M. Fitting. Basic modal logic. In *Handbook of Logic in AI and Logic Programming*, vol. 1, p. 365–448. Oxford Univ. Press, 1993.
5. P. Fleming and J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *CACM*, 29(3):218–221, 1986.
6. E. Giunchiglia, F. Giunchiglia, R. Sebastiani, and A. Tacchella. More evaluation of decision procedures for modal logics. In *Proc. of KR-98*, p. 626–635. 1998.
7. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *AIJ*, 54:319–379, 1992.
8. D. Johnson. A theoretician's guide to the experimental analysis of algorithms. Invited talk at AAAI-96. See `http://www.research.att.com/~dsj`, Aug. 1996.
9. D. Johnson and M. Trick, editors. *Cliques, Coloring, Satisfiability: the second DIMACS implementation challenge*, Am. Math. Soc., 1996.
10. R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM JoC*, 6(3):467–480, 1977.
11. F. Massacci. Design and Results of Tableaux-99 Non-Classical (Modal) System Competition. In *Proc. of TABLEAUX-99*, *LNAI*, Springer Verlag 1999.
12. D. McGuinness, and P. Patel-Schneider. Usability Issues in Description Logic Systems, *Proc. of DL-97*, pp 84-88, 1997.
13. J. Orlin. The complexity of dynamic languages and dynamic optimization problems. In *Proc. of STOC-81*, p. 218–227, 1981.
14. J. Rintanen. Improvements to the Evaluation of Quantified Boolean Formulae. in *Proc. of IJCAI-99*, p. 1192-1197, 1999.
15. M. Schmidt-Schauß and G. Smolka. Attributive Concept Descriptions with Complements. *AIJ*, 48(1):1–26, 1991.
16. B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *AIJ*, 81(1-2):17–29, 1996.
17. C. Suttner and G. Sutcliffe. The CADE-14 ATP system competition. *JAR*, 21(1):99–134, 1998.