À propos de la vérification de modèles en logique modale K

Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, Valentin Montmirail

CRIL-CNRS UMR 8188, F62300 Lens, France

JIAF'2016 - 17 juin 2016

Table of Contents

Introduction

- Key role in the success of SAT solvers for classical propositional logic : Their practical evaluation;
- We believe that such success can be repeated for many other systems;
- We are interested in making it happen for satisfiability in modal logic K;

However...

Introduction

- There is a need for a common input and output format;
- It is important to check the answers returned by a solver. (Solvers could have bugs)
- This task can be easy (SAT [11]) or still a challenge (QBF [9])

Introduction

- There is a need for a common input and output format;
- It is important to check the answers returned by a solver. (Solvers could have bugs)
- This task can be easy (SAT [11]) or still a challenge (QBF [9])
- K-SAT is PSPACE-Complete [5, 8].
- Models may be exponentially larger than the formula ($\leq 2^n$)

Introduction

Our work is in line with the one being done for QBF;
We aim at verifying, when possible, the answers of K-modal SAT solvers, and building a library of benchmarks with verified answers;

2 problems:

- Producing a certificate on the solver side;
- Checking it using an independent tool.

Both problems are addressed here in this presentation.

Premilinaries : Modal Logic K

Modal Logic K $=$ Propositional Logic plus 2 operators: $\Box$ and $\Diamond$;

- $\Box\varphi$ (box phi) means φ is necessarily true;
- $\Diamond\varphi$ (diamond phi) means φ is possibly true;

Premilinaries : Kripke Model

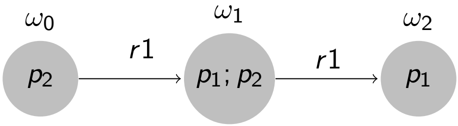
Let a non-empty countable set of variables P be given.

Definition

A Kripke model is a triplet $\mathrm{M} = \langle W, R, \mathcal{I} \rangle$, where:

- W : a non-empty set;
- $R \subseteq W \times W$;
- $\mathcal{I} : P \rightarrow 2^W$;

Example:



Let $M = \langle W, R, \mathcal{I} \rangle$, where:

- $W = \{\omega_0, \omega_1, \omega_2\}$
- $R = \{(\omega_0, \omega_1), (\omega_1, \omega_2)\}$
- $\mathcal{I} = \{(p_1, \{\omega_1, \omega_2\}), (p_2, \{\omega_0, \omega_1\})\}$

Definition

A pointed Kripke model is a Kripke Model M plus $\omega_0 \in W$.

Premilinaries : Satisfaction Relation

Definition

The satisfaction relation $\models$ between formulae and models is recursively defined as follows:

- $M, \omega \models p$ iff $\omega \in \mathcal{I}(p)$
- $M, \omega \models \Box\varphi$ iff for all γ if $(\omega, \gamma) \in R$ then $M, \gamma \models \varphi$
- $M, \omega \models \Diamond\varphi$ iff there exists γ s.t. $(\omega, \gamma) \in R$ and $M, \gamma \models \varphi$

Definition

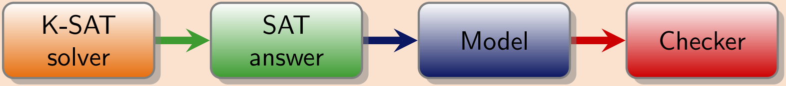
A formula φ is satisfiable in K if and only if there exists a model $\langle M, \omega_0 \rangle$ that satisfies φ.

Definition

Let a formula φ in the language of K be given. The K-satisfiability problem (K-SAT) is the problem of answering yes or no to the question "Is φ satisfiable?".

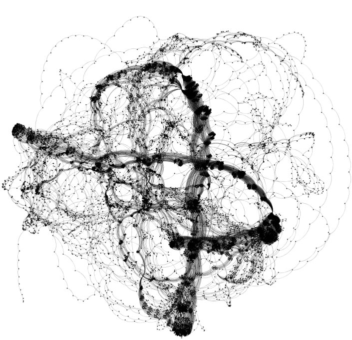
Checker

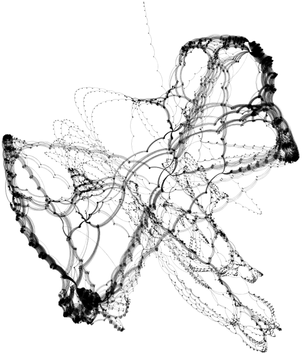
Ideally when the solution is SAT:



- Not the case in the solvers we used → Modifications.

Examples of Kripke models





Checker: Useful information

- The algorithm we used is based on the definition of the satisfiability relation.
- There are optimisations to make it efficient enough to check models in reasonable time (less than 300s).
- The code is written in C++. MacOSX, GNU/Linux
- Link: http://www.cril.fr/~montmirail/mdk-verifier.

To be used, this checker needs particular I/O...

I/O Format : InToHyLo

- There is no unified input format for K-SAT solvers;
- We analyse few: {ALC, LWB, InToHyLo, KRSS, TBOX}
- We select the more practical : InToHyLo [6]

I/O Format : InToHyLo

- There is no unified input format for K-SAT solvers;
- We analyse few: {ALC, LWB, InToHyLo, KRSS, TBOX}
- We select the more practical : InToHyLo [6]

Example

The formula $((p \rightarrow \Diamond^1 q) \wedge \Box^2 q)$
In InToHyLo as `((p1 -> <r1>p2) & [r2]p2 )`

The multiple modalities is not supported yet by our checker.
But the Input format is ready for it.

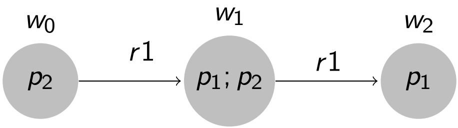
I/O Format : Flat Kripke Model

Check solvers answers $\rightarrow$ Output format to represent Kripke models.

We tried to keep the spirit of the DIMACS CNF format.

I/O Format : FKM : Example

Example:



2 3 1 2	#Vars #Worlds #Rels #Edges	
-1 2 0	$\neg p_1; p_2$	true in w_0
1 2 0	$p_1; p_2$	true in w_1
1 -2 0	$p_1; \neg p_2$	true in w_2
r1 w0 w1		
r1 w1 w2		

Solvers

- There are several solvers for K.
- We took only {InKreSAT, *SAT, Km2SAT, Spartacus} (All but one from [7]).
- Missing FaCT++ [12] because Spartacus outperform it ([4])

Solvers : Presentation

All modified to output a Kripke model in FKM.

Solver Name	Tableaux/SAT	SAT solver	Modifications
*SAT [3]	SAT	SATO 3.2.0 [13]	ALC $\rightarrow$ InToHyLo
Spartacus [4]	Tableaux		
InKreSAT [7]	Tableaux/SAT	MiniSAT 2.2.0 [1]	
Km2SAT [10]	SAT	MiniSAT 2.2.0 [1]	LWB $\rightarrow$ InToHyLo

Evaluation : Settings

The solvers ran on a cluster of:

- Identical computers
- 2 processors Intel XEON E5-2643 - 4 cores - 3.3 GHz
- CentOS 6.0
- 32 Go of memory

Each solver was given:

- 4 cores for its execution.
- Timeout : 900s
- Memory limit: 15500 MB.

Results

In the following tables:

- **Highest number of problems solved**;
- (How many times the solver is the fastest);
- [How many candidates for verification].

Results : 3CNF

d	#	**Km2SAT**	***SAT**	**InKreSAT**	**Spartacus**	**Verified SAT**
2	45	**45**	33	**45**	42	26 / 45 (57.78%)
4	45	9/MO	20	12	**45**	**45 / 45** (100.0%)
6	45	0/MO	7	8	**45**	**45 / 45** (100.0%)
total	135	54 (0)	60 (2)	65 (13)	**132** (120)	116 / 135 (85.92%)

Table: 3CNF_K Results (d = modal depth)

For each depth d, the problems consists of 9 formulae with the number of clauses = {30, 60, 90, 120, 150}, respectively.

Results : MQBF

n,*a*	#	**Km2SAT**	***SAT**	**InKreSAT**	**Spartacus**	**Verified SAT**
4,4	40	MO	**40** [**18**]	**40** [**18**]	**40** [**18**]	11 / 18 (61.11%)
4,6	40	MO	**40** [**20**]	39 [19]	**40** [**20**]	9 / 20 (45.00%)
8,4	40	MO	**40** [**31**]	29 [22]	37 [28]	6 / 31 (19.35%)
8,6	40	MO	31 [30]	16 [16]	**34** [31]	0 / 33 (00.00%)
16,4	40	MO	26 [25]	17 [16]	**29** [**28**]	0 / 28 (00.00%)
16,6	40	MO	25 [25]	16 [16]	**29** [**29**]	0 / 29 (00.00%)
total	240	MO	202 (41)	157 (1)	**209** (173)	26 / 159 (16.35%)

Table: qbfMS

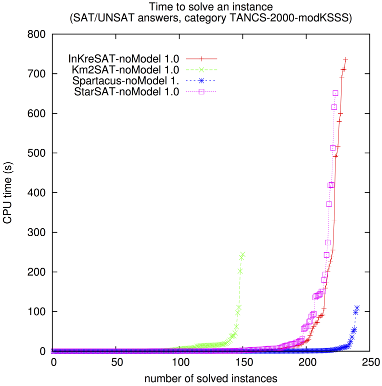
- *n*: Number of variable in the original QBF formula;
- *a*: Alternation depth of the QBF formula;

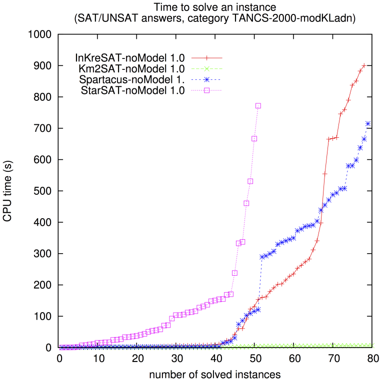
Results : TANCS-2000

n,*a*	#	Km2SAT	*SAT	InKreSAT	Spartacus	Verified SAT
4,4	40	**40** [**17**]	**40** [**17**]	**40** [**17**]	**40** [**17**]	**17 / 17** (100%)
4,6	40	**40** [**25**]	**40** [**25**]	**40** [**25**]	**40** [**25**]	23 / 25 (92.00%)
8,4	40	**40** [**26**]	**40** [**26**]	**40** [**26**]	**40** [**26**]	**26 / 26** (100%)
8,6	40	14/MO [14]	38 [35]	37 [34]	**40** [**37**]	18 / 37 (48.64%)
16,4	40	8/MO [8]	33 [33]	36 [36]	**40** [**40**]	21 / 40 (52.50%)
16,6	40	8/MO [8]	32 [32]	38 [38]	**40** [**40**]	15 / 40 (37.50%)
total	240	150 (2)	223 (13)	231 (1)	**240** (226)	120/ 185 (64.86%)
4,4	40	**40** [**19**]	**40** [**19**]	**40** [**19**]	**40** [**19**]	**19 / 19** (100%)
4,6	40	**40** [**24**]	11 [6]	38 [22]	39 [23]	**24 / 24** (100%)
total	80	**80** (80)	51 (0)	78 (0)	79 (0)	**43 / 43** (100%)

Table: Upper: modKSSS — Lower: modKLadn

Results : TANCS-2000





Results : Tableaux'98

name	#	**Km2SAT**	***SAT**	**InKreSAT**	**Spartacus**	**Verified SAT**
branch_n	17	5/MO	**12**	11	9	5 / 12 (41.66%)
dump_n	21	**21**	**21**	**21**	**21**	**21 / 21** (100.0%)
grz_n	21	**21**	**21**	**21**	**21**	**21 / 21** (100.0%)
d4_n	21	6/MO	**21**	**21**	**21**	19 / 21 (90.47%)
lin_n	21	**21**	**21**	**21**	**21**	**21 / 21** (100.0%)
path_n	21	8/MO	**21**	20	**21**	**21 / 21** (100.0%)
t4p_n	21	6/MO	**21**	**21**	**21**	**21 / 21** (100.0%)
ph_n	21	**21**	*17*	**21**	**21**	**21 / 21** (100.0%)
poly_n	21	**21**	**21**	**21**	**21**	**21 / 21** (100.0%)
branch_p	21	5/MO	**21**	20	13	
dump_p	21	20/MO	**21**	**21**	**21**	
grz_p	21	**21**	**21**	**21**	**21**	
d4_p	21	11/MO	**21**	**21**	**21**	
lin_p	21	**21**	**21**	**21**	**21**	
path_p	21	9/MO	**21**	17	**21**	
ph_p	21	**10**	**10**	**10**	9	
poly_p	21	**21**	**21**	**21**	**21**	
t4p_p	21	11/MO	**21**	**21**	**21**	
total	374	259 (2)	**354** (281)	351 (6)	346 (53)	171 / 184 (92.93%)

Table: Tableaux'98 Benchmarks for K

Results

- Check globally 67% of all the satisfiable benchmarks.
- Two solvers are the main providers.

	#SAT	#Verified	Uniquely Verified
Km2SAT	330	168	39/476
***SAT**	584	6	0/476
InKreSAT	573	26	0/476
Spartacus	696	443	352/476
Global	708	476	391/476

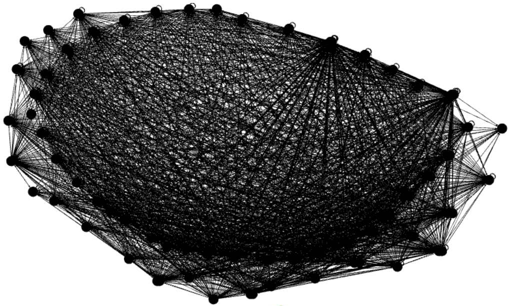
Results: Challenging models

	Provider	Instance	#Worlds	#Edges	#Vars	#Modal Operator	#Boolean Operator	Modal Depth	Verification time
Biggest	Km2SAT	modKSSS-C20-V8-D6.7	10,618,391	10,618,390	46	1,075	1,268	56	23.68 seconds
Smallest	*SAT	modKSSS-C10-V16-D4.4	81	2,792	80	161,495	167,598	799	>300 seconds
	Km2SAT	modKSSS-C10-V16-D4.4	2,972	2,971	80	161,495	167,598	799	1.46 seconds

Results: Challenging models

	Provider	Instance	#Worlds	#Edges	#Vars	#Modal Operator	#Boolean Operator	Modal Depth	Verification time
Biggest	Km2SAT	modKSSS-C20-V8-D6.7	10,618,391	10,618,390	46	1,075	1,268	56	23.68 seconds
Smallest	*SAT	modKSSS-C10-V16-D4.4	81	2,792	80	161,495	167,598	799	>300 seconds
	Km2SAT	modKSSS-C10-V16-D4.4	2,972	2,971	80	161,495	167,598	799	1.46 seconds

*SAT:



Conclusion

- We standardize the I/O by designing a new output format, the Flat Kripke Model (FKM) : $\checkmark$;
- We modified several state-of-the-art solvers for modal logic K to provide such models: $\checkmark$;
- We have been able to verify 67% of all the satisfiable benchmarks: $\checkmark$;

Perspectives

Verifying UNSAT answers

A next logical step would be to study the feasibility to validate UNSAT answers.
In the spirit of what is already done for SAT, based on the previous work on UNSAT proofs in the modal logic K [2].

À propos de la vérification de modèles en logique modale K

Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, Valentin Montmirail

CRIL-CNRS UMR 8188, F62300 Lens, France

JIAF'2016 - 17 juin 2016

Bibliography I

N. Eén and N. Sörensson.
An Extensible SAT-solver.
In *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Selected Revised Papers*, pages 502–518, 2003.

Patrice Enjalbert and Luis Fariñas del Cerro.
Modal Resolution in Clausal Form.
Theor. Comput. Sci., 65(1):1–33, 1989.

Bibliography II

E. Giunchiglia and A. Tacchella.
System description: *SAT: A platform for the development of modal decision procedures.
In *Automated Deduction - CADE-17 Proceedings*, pages 291–296, 2000.

D. Götzmann, M. Kaminski, and G. Smolka.
Spartacus: A Tableau Prover for Hybrid Logic.
ENTCS, 262:127–139, 2010.

J. Y. Halpern and Y. Moses.
A Guide to Completeness and Complexity for Modal Logics of Knowledge and Belief.
Artificial Intelligence, 54(2):319–379, 1992.

Bibliography III

G. Hoffmann.
Tâches de raisonnement en logiques hybrides.
PhD thesis, Citeseer, 2010.

M. Kaminski and T. Tebbi.
InKreSAT: Modal Reasoning via Incremental Reduction to SAT.
In *Automated Deduction - CADE-24 Proceedings*, pages 436–442, 2013.

Richard E. Ladner.
The computational complexity of provability in systems of modal propositional logic.
SIAM Journal of Computing, 1977.

Bibliography IV

M. Narizzano, C. Peschiera, L. Pulina, and A. Tacchella.
Evaluating and certifying QBFs: A comparison of state-of-the-art tools.
AI Communications, 22(4):191–210, 2009.

R. Sebastiani and M. Vescovi.
Automated reasoning in modal and description logics via SAT encoding: the case study of k(m)/alc-satisfiability.
J. Artif. Intell. Res. (JAIR), 35:343–389, 2009.

L Simon, D Berre, and E A. Hirsch.
The SAT2002 competition.
Annals of Mathematics and Artificial Intelligence, 43(1):307–342, 2004.

Bibliography V

D. Tsarkov and I. Horrocks.
FACT++ Description Logic Reasoner: System Description.
In *IJCAR 2006 Proceedings*, pages 292–297, 2006.

H. Zhang.
SATO: An Efficient Propositional Prover.
In *Automated Deduction - CADE-14 Proceedings*, pages 272–275, 1997.