

# Fixpoint Characterisations of LTL for Bounded Model Checking

Daniel Sheridan

Artificial Intelligence Group,  
Department of Computer Science,  
University of York, England. YO10 5DD  
djs@cs.york.ac.uk

## Introduction

Model checking is the process of verifying the behaviour of a dynamic system, where a *model* of the implementation of a system is compared with a mathematical *specification* of an intended behaviour of the system. A typical application is hardware verification: a model checking tool may be able to find bugs in a hardware design long before the product enters construction and testing.

Bounded model checking (BMC) (Biere et al., 1999) was proposed as a solution to some of the problems of conventional BDD-based symbolic model checking such as space explosion by introducing a temporal bound. The problem is then encoded as a Boolean formula: a conjunction of state transition functions and verification constraints which can be solved with a Boolean satisfiability (SAT) solver such as Chaff (Moskewicz et al., 2001). Infinite temporal operators can be handled by including a check for loops in the state transitions.

Although BMC has found acceptance, it can produce an exponential number of clauses in the specification depth. The present work is concerned with reducing this size explosion, so making BMC more accessible.

## Model Checking

A model checking problem is a pair  $\langle M, f \rangle$  of a model and a temporal logic specification.

A model  $M$  is defined as a Kripke structure  $\langle S, R, L, I \rangle$  where  $S$  is a set of states;  $R : S \rightarrow S$  is the transition relation;  $L : S \rightarrow \mathcal{P}(AP)$  is the labelling function, marking each state with the set of atomic propositions ( $AP$ ) that hold in that state; and  $I$  is the set of initial states, which may be equal to  $S$ . A path  $\pi \in M$  is a sequence of states  $s_0, s_1, \dots \in M$  such that  $\forall i. (s_i, s_{i+1}) \in R$ . We write  $\pi(i)$  to refer to the  $i$ th state in the path, and say that a path  $\pi$  is a  $k$ -loop if the  $\pi(k) = \pi(l)$ , for some  $l, 0 \leq l < k$ .

The *model checking problem for LTL* is to verify that for an LTL formula  $f$ , for all paths  $\pi_i \in M$  such that  $\pi_i(0) \in I$ ,  $(M, \pi_i) \models f$ .

## Bounded Model Checking

The bounded semantics of LTL depends on whether a path is a loop; for example the *global* operator  $\mathbf{G}$  cannot hold

on a bounded, non-looping path, because the behaviour after the bound is unknown.

The bounded model checking encoding represents a single bounded path  $\pi_{bmc}$ , and checks that it violates the bounded semantics of the specification  $f$  either with or without loops. That is, that  $(M, \pi_{bmc}) \not\models_k f$ .

## Fixpoint Characterisations for LTL

Emerson and Clarke (1980) give fixpoint characterisations for CTL; these are easily adapted for LTL. They are given for *unbounded* temporal logic, however the fixpoint characterisations are preserved for most of bounded LTL since we have bounded semantics for  $\mathbf{X}$ . We note that the characterisation of  $\mathbf{G}$  is valid if and only if the path is a  $k$ -loop. This constraint may be captured with a new temporal operator;  $\mathbf{X}_1 f$  holds when  $f$  holds in the next state, and the current path is a loop.

Each bounded LTL formula  $f$  is identified with a set of paths in which it holds. This allows us to give the following characterisations as the least and greatest fixpoints of functions on sets of paths:

$$\mathbf{F} f_1 = \mu Z. f_1 \vee \mathbf{X} Z$$

$$\mathbf{G} f_1 = \nu Z. f_1 \wedge \mathbf{X}_1 Z$$

$$[f_1 \mathbf{U} f_2] = \mu Z. f_2 \vee (f_1 \wedge \mathbf{X} Z)$$

$$[f_1 \mathbf{R} f_2] = \nu Z. f_2 \wedge (f_1 \vee \mathbf{X} Z)$$

## The Separated Normal Form

Fisher (1991) defined a normal form for temporal logic based on the Separation Theorem (Gabbay, 1989) and gave a series of transformations for reaching it. We have adapted the Separated Normal Form (SNF) for bounded LTL.

The general form of SNF is  $\mathbf{G} (\bigwedge_i (P_i \Rightarrow F_i))$  where  $P_i$  are (strict) past time formulae and  $F_i$  are (non-strict) future time formulae. For the bounded semantics, we introduce a new path operator to capture the semantics of *all reachable states*. The expression  $\mathbf{G}_k f$  holds iff  $f$  holds in all states before the bound. Since LTL (and CTL) have no explicit past-time operators, Bolotov and Fisher (Bolotov and Fisher, 1997) introduce the **start** operator, which holds only at the beginning of time.

The rules  $P_i \Rightarrow F_i$  are of the following form:

$$\begin{aligned} \text{start} &\Rightarrow \bigvee_j l_j && \text{An initial rule} \\ \bigwedge_i l_i &\Rightarrow \mathbf{X}_1 \bigvee_j l_j && \text{A global } \mathbf{X}_1\text{-rule} \\ \bigwedge_i l_i &\Rightarrow \mathbf{X} \bigvee_j l_j && \text{A global } \mathbf{X}\text{-rule} \\ \bigwedge_i l_i &\Rightarrow \mathbf{F} \bigvee_j l_j && \text{A global } \mathbf{F}\text{-rule} \end{aligned}$$

where  $l_i$  and  $l_j$  are literals. By applying transformations (adapted from Bolotov and Fisher (1997)), any temporal logic formula may be converted to SNF, hence to a formula of depth 2 with only the  $\mathbf{X}$  and  $\mathbf{F}$  temporal operators. This means that the encoding to SAT can be optimised for the limited number of cases, yielding a smaller and better performing encoding.

### The Fixpoint Normal Form

SNF treats  $\mathbf{F}$  as a fundamental operation. Since we do not have the same restrictions as the original application domain of SNF, we introduce a transformation which uses the fixpoint characterisation of  $\mathbf{F}$  to reduce the set of possible rules even further.

To make the satisfiability meaningful in the context of this transformation when using bounded temporal logic, we introduce a new past-time operator **bound**, which holds only at the temporal bound set on the system. This is used as a statement that the event we are looking for does not occur in the otherwise unconstrained future.

This additional transformation may occasionally be undesirable: it generates a larger number of short clauses, where a direct encoding of the  $\mathbf{F}$  operator may generate a single long clause if its argument is a literal.

### Results

Figure 1 shows the number of decisions made by zChaff for a simple shift register, comparing the new encodings with the original encoding from Biere et al. (1999). The depth of the specifications increases from top to bottom.

There is a general improvement in most cases, and as the problem size increases, the advantage of the new encodings also increases. The performance increases with nesting depth for most of the specifications, and the results for Fixpoint are consistently better than the original encoding, and are more uniform; it also outperforms Snf in almost all cases.

### Conclusion

We have demonstrated that performing SNF style transformations on the specification of a BMC problem can result in an improvement in performance in the SAT checker; extending the SNF transformations with a transformation

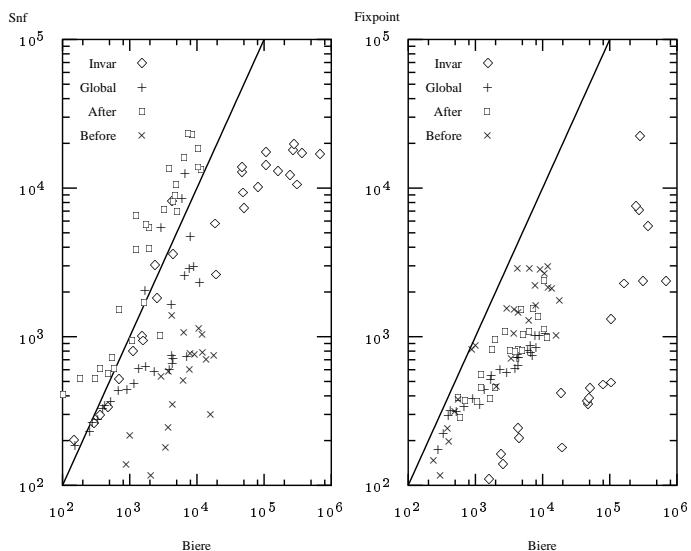


Figure 1: zChaff decisions

for the  $\mathbf{F}$  operator results in further advantages over SNF in most cases.

### Acknowledgements

This work could not have occurred without the initial suggestion from Michael Fisher at AISB 2001.

### References

- Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In W.R. Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems. 5th International Conference, TACAS'99*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer-Verlag Inc., July 1999.
- Alexander Bolotov and Michael Fisher. A resolution method for CTL branching-time temporal logic. In *Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning (TIME)*. IEEE Press, 1997.
- E. Allen Emerson and Edmund M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In Jan van Leeuwen J. W. de Bakker, editor, *Automata, Languages and Programming, 7th Colloquium*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181. Springer-Verlag Inc, 1980. ISBN 3-540-10003-2.
- Michael Fisher. A resolution method for temporal logic. In *Proceedings of Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, August 1991.
- Dov Gabbay. The declarative past and imperative future. In H. Barringer, editor, *Proceedings of the Colloquium on Temporal Logic and Specifications*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer-Verlag, 1989.
- M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *39th Design Automation Conference*, Las Vegas, June 2001.