*URL http www elsevier nl locate entcs volume html pages*

*Combining Local and Global Model Checking Armin Biere a b c Edmund M Clarke b c Yunshan Zhu b c*

*a Institut fu r Logik Komplexitat und Deduktionssysteme ILKD University of Karlsruhe Postfach Karlsruhe Germany*

*b Computer Science Department Carnegie Mellon University*

*Forbes Avenue Pittsburgh PA U S A*

*c Verysys Design Automation Inc Lawrence Place Fremont CA*

*Abstract*

*The veri cation process of reactive systems in local model checking and in explicit state model checking is on the y Therefore only those states of a system have to be traversed that are necessary to prove a property In addition if the property does not hold than often only a small subset of the state space has to be traversed to produce a counterexample Global model checking and in particular symbolic model checking can utilize compact representations of the state space e g BDDs to handle much larger designs than what is possible with local and explicit model checking We present a new model checking algorithm for LTL that combines both approaches In essence it is a generalization of the tableau construction of that enables the use of BDDs but still is on the y*

*Introduction*

*Model Checking is a powerful technique for the veri cation of reactive* systems In particular with the invention of symbolic model checking very large systems with more than states could be veri ed However it is often observed that explicit state model checkers outperform symbolic model checkers especially in the application domain of asynchronous systems and communication protocols We believe that. the main reasons are the

*following First symbolic model checkers traditionally use binary decision*

*diagrams BDDs as an underlying data structure BDDs trade space for* time and often their sheer size explodes Second depth rst search DFS is used in explicit state model checking while symbolic model checking usually traverses the state space in breadth rst search BFS DFS helps to reduce the space requirements and is able to nd counterexamples much faster Finally global model checking traverses the state space backwards and can in general not avoid to visit non reachable states without a prior reachability analysis

*c Published by Elsevier Science B V Open access under* [*CC BY-NC-ND license*](http://creativecommons.org/licenses/by-nc-nd/3.0/)*.*

*In a solution to the rst problem and partially to the second problem was presented by replacing BDDs by SAT propositional satis ability check ing procedures In this paper we propose a solution to the second and third* problem of symbolic model checking Our main contribution is a new model checking algorithm that generalizes the tableau construction of local model checking for LTL and enables the use of BDDs It is based on a mixed DFS and BFS strategy and traverses the state space in a forward oriented manner Our research is motivated by the success of forward model checking Forward model checking is a variant of symbolic model checking in which only forward image computations are used Thus it mimics the on the y nature of explicit and local model checking in visiting only reachable states Note that presented a technique for the combination of the BFS used in BDD based approaches with the DFS of explicit state model checkers It was shown that especially this feature enables forward model checking to nd counterexamples much faster However only a restricted class of properties

*i e path expressions can be handled by the algorithms of*

*Henzinger et al in partially lled this gap by proving that all proper ties speci ed by Bu chi Automata or equivalently all regular properties can* be processed by forward model checking In particular they de ne a forward oriented version of the modal calculus called post and translate the model checking problem of a regular property into a post model checking problem Because LTL linear temporal logic properties can be formulated as regular properties their result subsumes that all LTL properties can be checked by forward model checking

*The fact that LTL can be checked by forward model checking can also* be derived by applying the techniques of in the special case of FairCTL properties to the tableau construction of However this construction and also do not allow the mixture of DFS and BFS as in the layered approach of In addition DFS was identi ed as one major reason for explicit state model checking to outperform symbolic model checking on certain examples The contribution of our paper is the following First we present a new

*model checking algorithm that operates directly on LTL formulae For ex*

*ample requires two translations from LTL to Bu chi Automata and then* to post A similar argument applies to Second it connects the local model checking paradigm of with symbolic model checking in a natural way thus combining BDD based with on the y model checking Finally our approach shows that the idea of mixing DFS with BFS can be lifted from path expressions to LTL

*Our procedure is correct and complete for all of LTL If we consider ex istential model checking problems with no eventualities then the size of the* generated tableaux is linear in the number of states Checking eventualities may result in an tableau with exponential size in the number of states We are currently working on an extension that remains complete for all of LTL and produces tableaux with size linear in the number of states

*Our paper is organized as follows In the next section our notation is* introduced Section presents our new tableau construction The following section considers an essential optimization followed by a discussion of the complexity and the comparison with related work Finally we address open issues

*Preliminaries*

*A Kripke structure is a tuple K with a nite set of states*

*the set of initial states the transition relation between* states and p A the labeling of the states with atomic propositions As temporal operators we consider the next time operator X the nally operator F the globally operator G the until operator U and its dual the release operator R We use the standard semantics of CTL as in We further assume the formulae to be in negation normal form as in Thus negations only occur in front of atomic propositions This restriction does not lead to an exponential blow up because we included the R operator that ful lls the property f U g f R g

*Tableau Construction*

*In this section we present a new model checking algorithm for solving exis tential LTL model checking problems In particular given a Kripke structure* K and an LTL formula f the algorithm determines whether j Ef where S j Ef i there exists a path with S and j f A proce dure for generating counterexamples in case j Ef does not hold is also included

*The algorithm is based on a tableau construction Each tableau node is a* sequent that contains a set of states S and an LTL formula f written S E f The rules for the construction of the tableau are very similar to those in which is the dual construction of for LTL with an existential path quanti er

*The main di erence to is also the main idea of our paper We use* sets of states instead of single states as one part of the sequent With this modi cation we are able to represent set of states symbolically and use e cient BDD algorithms

*For the rest of the paper let S be a set of states and E E V i be*

*a conjunctively decomposed ELTL formula We also use the notation E f with the semantics E V i f Further for S p A we de ne*

*Sp fs S j p s g S p fs S j p s g Img S ft j s S s t g*

*Given an initial set of states S e g and an ELTL formula f we construct*

*RU*

*S E f U g*

*S E f g*

*R*

*S E g S E f Xf U g*

*S E f g*

*RR*

*S E f R g*

*R*

*S E f g*

*S E f g S E g Xf R g*

*S E Ff*

*RF*

*S E f S E XFf*

*S E X X n*

*RX*

*Img S E n*

*S E f S E g*

*S E Gf*

*RG*

*S E f XGf*

*S E p*

*RA*

*Sp E*

*Rsplit*

*S E*

*S E S E*

*S S S RA*

*S E p Sp E*

*Fig Tableau rules*

*a tableau by repeatedly applying the rules of Figure starting with the root* S E f

*We continue the application of the rules until no new sequents can be* added In the resulting graph which we call a tableau every sequent occurs only once Note that a tableau may be cyclic and in general is not uniquely de ned

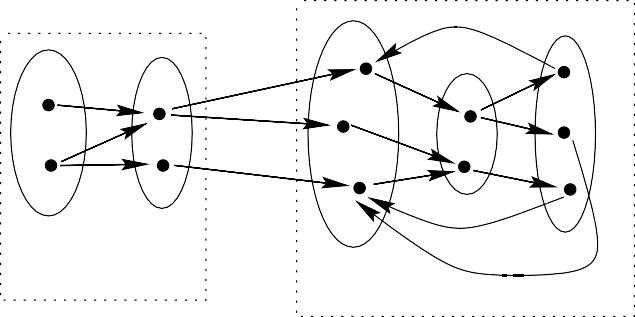
*Following we rst de ne a successful path in the tableau A nite path* through the tableau that ends with a sequent S E is called successful i S and An in nite path X is called successful i for every Fg X i and every f U g X i there exists a j i with g X j

*A tableau is called successful if it contains a successful path From this* successful path we can construct a witness for the existential model checking problem associated with the root sequent of the tableau

*The following theorem shows that no matter in which order we apply* the tableau rules the resulting tableau is successful i the root sequent is satis able We call a sequent S E f satis able i S j Ef

*Theorem Let K be a Kripke structure Ef an ELTL formula and T a tableau with root E f Then j Ef i T is successful*

*The proof consists of the combination of the following Lemma with the* correctness and completeness results of We call a path x of sequents singleton path i every sequent in x contains only a singleton set of states



1

2

3

A

B

4

5

6

Y

Z

*Fig Example for witness resp counterexample generation*

*Further let X S E f S E f be a nite or in nite path then a singleton path x fsg f fsg f matches X i si Si and if*

*X i is the result of applying RX to X i i e X i Img X i then*

*si si R*

*Lemma Let X be a successful path for the root sequent S E f Then there exists s S and a successful singleton path x for the root sequent fsg E f that matches X*

*The Lemma is proven by constructing a matching singleton path from a* successful path What follows is a sketch of this algorithm for an in nite path

*X Y Z A sequent is called an X sequent i the RX rule is applicable* to i e all formulae on the right hand side of are pre xed with the next time operator X For the purpose of constructing a singleton path only the X sequents of X are considered We pick an arbitrary state s out of the rst X sequent in Z Note that s is also contained in X j with X j an X sequent and j jY j jZj

*Now we traverse the X sequents of Z until the last X sequent of Z is* reached During this traversal we choose an arbitrary successor state from the following X sequent We can not choose a successor state in the immediate successor sequent since this successor state might be eliminated by the ap plication of the RA rule before the next X sequent is reached When the last X sequent in Z is reached then we check if the state chosen initially can be reached in one step from the current state If this is the case then we found a singleton cycle and continue to search a pre x singleton path for this cycle in Y

*Otherwise we repeat the traversal of Z starting from an arbitrary image* state of the last state that is contained in the rst X sequent of Z until such a cycle is found Because and thus the number of di erent sequents is

*nite the algorithm has to terminate The resulting singleton path obviously* matches the original path and is successful if the original path was successful Consider the example of Figure where each ellipsis depicts an X sequent

*The arrows between the single states are transitions of the Kripke struc ture We start with transition to and pick as successor of The next*

*transition from to brings us back to the rst X sequent of Z but no cycle* can be closed yet We continue with and and nally reach again From there we nd a pre x A B that leads from the initial state A to the start of the cycle at The resulting singleton path is A B Note that this algorithm is actually used for the generation of a witness for the root formula

*or a counterexample for the negation of the root formula*

*The theorem follows by the observation that every successful singleton path* can be interpreted as a successful path in the sense of and vice versa This mapping has to take into account the split rule Rsplit but otherwise just maps a singleton set into the single state contained in the set Note that the tableaux for x and X in general are di erent

*For instance consider the Kripke structure K with two states and both* initial states and two transitions from state to state and from state to state Both states are labeled with p the only atomic proposition The tableau for checking EGp looks as follows

*f g E Gp*



0

1

*f g E p XGp*

*p p*

*f g E XGp*

*and the application of RX to the leaf sequent leads back to the root sequent The tableau represents one successful path that contains only one image calcu lation However both matching singleton paths need two image computations* before the loop can be closed

*f g E Gp*

*f g E p XGp f g E XGp*

*f g E Gp*

*f g E p XGp f g E XGp*

*f g E Gp*

*f g E p XGp f g E XGp*

*f g E Gp*

*f g E p XGp f g E XGp*

*Again the application of RX to the leaf nodes yields the root In general*

*matching singleton paths may require longer closing cycles than a matched* path

*Algorithm*

*A more detailed description of the tableau construction is presented in this* section The overall approach expands open branches in DFS manner and stops when a successful path has been generated In this case the formula can be ful lled If no successful path can be found and the tableau has been fully generated then the algorithm stops reporting that no witness has been found



1

2

3

*Fig Example Kripke structure*

*If a leaf of a tableau is expanded and a sequent is generated that already* occurred in the tableau then we found a successful path if the previous occur rence is on the path from the root to the expanded node and all eventualities on this path are ful lled If the new sequent occurs in the tableau but not on the path from the root to the expanded leaf the parent of the new se quent then we already have proven that the new sequent is unsatis able In the remaining case the new sequent occurs on the path from the root to the expanded node and at least one eventuality is not ful lled the strongly connected components of the tableau have to be considered as in

*During the construction we have to remember the sequents that already* occurred in the tableau This can be accomplished by a partial function mapping a sequent to a node To implement this we can sort the sequents in the tableau use a hash table or simply an array Hash tables work very well in practice

*Our intention of course is to represent set of states with BDDs We* associate with each formula E the list of sequents in the tableau that contain E To check if a sequent already occurred we just go through the list of corresponding formulae and check whether the BDDs representing the sets of states are the same We can also combine several nodes on unsuccessful branches with the same formula by computing the disjunction of the BDDs But keeping the BDDs separate results in a partitioning of the search space and hopefully results in small BDDs Note that the same approach works for the optimization discussed in section with the only modi cation that we check for non empty intersection instead of checking for equality

*Heuristics*

*The rule Rsplit is not really necessary but it helps to reduce the search space i e the size of the generated tableau For instance consider the construction of* a tableau for the formula EFp This formula is the negation of a simple safety property In this case a good heuristics is to build the tableau by expanding the left successor of the rule RF rst Only if the left branch does not yield a successful path then the right successor is tried If during this process a sequent S E Ff is found and a sequent S E Ff occurs on the path from the root to and S S then we can remove the set S from S by applying Rsplit with S S and S S S The left successor immediately leads to an unsuccessful in nite path and we can continue with the right successor

*f g E Fp RF*

*f g E p*

*fg E p*

*f g E XFp*

*f g E Fp*

*Rsplit*

*f g E Fp*

*f g E Fp*

*RF*

*f g E p*

*fg E p*

*f g E XFp*

*f g E Fp*

*Rsplit*

*f g E Fp*

*RF*

*f g E Fp*

*f g E p*

*fg E p*

*f g E XFp*

*f g E Fp*

*Rsplit*

*f g E Fp f g E Fp*

*Fig Example for the usage of the split rule Rsplit*

*Applying this heuristics essentially computes the set of reachable states in a* BFS manner while checking on the y for states violating the safety property An example of this technique is shown in gure using the Kripke structure of gure

*Another heuristic is to avoid splitting the tableau as long as possible This* is one of the heuristics proposed in for the construction of small tableau as an intermediate step of translating LTL into the modal calculus with the algorithm of In general these heuristics are also applicable in our approach

*Optimization*

*The number of di erent left hand sides of sequents is exponential in j j the* number of states of the Kripke structure If we only consider LTL properties that do not contain eventualities then we can apply an optimization that reduces the maximal number of di erent left hand sides occurring in the tableau to j j This reduction can be achieved by modifying the tableau construction in such a way that all sequents with the same formula contain mutually exclusive set of states

*The tableau is built with DFS The construction is stopped immediately* if a successful path has been found Otherwise the still open branches are expanded If there are no more open branches the construction terminates with failure

*Assume that the result of applying a rule is a new sequent S E f and there is another sequent S E f with the same formula already in* the tableau First if is not on the path from the root to this is a cross edge in terms of DFS then we already have proven that all states s S

*can not ful ll s j Ef This allows us to remove all states in the intersection* S S and we use S S E f instead of as new tableau node

*Second let be a predecessor of Then we have to check if there is a* self loop of a state in the intersection S S along the segment If this is the case a successful path has been found since by our restriction the path does not contain any eventuality and we can terminate the search immediately Otherwise we can remove the intersection as in the previous case

*To check for a successful path as in the last case is similar to the gen eration of witnesses of Section We start with the intersection S S at*

*and compute all images along the segment restricting the image set to the* set of states occurring in the sequents along the segment If we reach and the set of states has become empty then no loop is possible This conclu sion remains correct even if the path contains eventualities Otherwise we repeat the calculation with the intersection of the calculated set with S S restricting the images to previously calculated images If we reach a x point a set that yields the same result after one iteration then a successful path exists A witness resp counterexample can be extracted with the algorithm of Lemma

*If the optimization is applied without the restriction i e the root formula* contains eventualities then our optimized procedure becomes incomplete but the size of the tableau is linear in j j Incompleteness means that a witness for an existential model checking problem found by the optimized procedure is indeed a witness However if the procedure can not nd a successful path applying the optimization then the root sequent might still be satis able

*Complexity and Related Work*

*In this section we discuss the complexity of our algorithm Then we compare* our approach with other local and global techniques for LTL model checking The size of a tableau with root E f not using the optimization of

*the last section is in O exp j j exp jf j The time taken is polynomial in*

*the size of the tableau Thus the time complexity is roughly the same as the* space complexity

*The optimization of the last section generates a tableau with the property* that sequents with the same formula have mutually exclusive sets of states Because there are no more than j j sets of states that are mutually exclusive any formula occurs in at most j j sequents Therefore the size of the resulting tableau is linear in the number of states and exponential in the size of the formula Consequently our algorithm is polynomial in the number of states with a small degree polynomial and exponential in the size of the formula However to achieve this complexity we have to restrict the class of properties or give up completeness

*This result almost matches the worst case complexity of explicit state* model checking algorithms for LTL which are linear in the number

*of states and exponential in the size of the formula However with our ap proach we are able to use e cient data structures to represent set of states* symbolically and thus can hope to achieve exponentially smaller tableaux and exponentially smaller running times for certain examples

*The method of translates an LTL formula into a tableau similar to the* tableaux in our approach In the nodes contain only formulae and no states The tableau can be exponential in the size of the LTL formula The second step is a translation of the generated tableau into a calculus formula that is again exponential in the size of the tableau Additionally the alternation depth of the calculus formula can not be restricted With this results in a model checking algorithm with time and space complexity that is double exponential in the size of the formula and single exponential in the size of the model K

*In an ELTL formula is translated to a Bu chi automata with the method* of This leads to an exponential blow up in the worst case But see for an argument why this explosion might not happen in practice which also ap plies to our approach The resulting Bu chi automata is translated to post a forward version of the standard modal calculus for which similar complexity results for model checking as in can be derived This translation pro duces a calculus formula of alternation depth which results in practically the same complexity as our algorithm

*The LTL model checking algorithm of is also forward oriented A* forward state space traversal potentially avoids searching trough non reachable states as it is usually the case with simple backward approaches However it is not clear how DFS can be incorporated into symbolic calculus model checking

*The method of translates an LTL model checking problem into a FairCTL* model checking problem With the result of this leads to a model checking algorithm that is linear in the size of the model and exponential in the size of the formula Again these complexity results are only valid for explicit state model checking If is not combined with then it also shares the following disadvantage with the LTL model checking algorithm of The algorithm is based on BFS and it is not clear how to combine it DFS

*The work by Iwashita does not handle full LTL and no complexity* analysis is given But if we restrict our algorithm to the path expressions of

*then our algorithm subsumes the algorithms of even for the* layered approach of the combination of DFS and BFS

*Conclusion*

*Although our technique clearly extends the work of and bridges the gap* between local and global model checking we still need to show that it works in practice In addition a formalization of the optimization in Section is necessary We are also working on a complete tableau construction for

*eventualities with linear tableau size in the number of states Finally we* want to investigate heuristics for applying the split rule The approximation techniques of are a good starting point

*References*

*G Bhat R Cleaveland and O Grumberg E cient on the y model checking for CTL In LICS IEEE Computer Society*

*A Biere A Cimatti Edmund M Clarke and Y Zhu Symbolic model checking without BDDs In TACAS LNCS Springer*

*R E Bryant Graph based algorithms for boolean function manipulation IEEE Transactions on Computers*

*J R Burch E M Clarke and K L McMillan Symbolic model checking states and beyond Information and Computation*

*E Clarke O Grumberg and K Hamaguchi Another look at LTL model checking In CAV LNCS Springer*

*E M Clarke and E A Emerson Design and synthesis of synchronization skeletons using branching time temporal logic In Logic of Programs Workshop LNCS Springer*

*R Cleaveland Tableau based model checking in the propositional mu calculus Acta Informatica*

*M Dam CTL and ECTL as fragments of the modal mu calculus Theoretical Computer Science*

*D L Dill The Mur veri cation system In CAV LNCS Springer*

*Y Dong X Du Y S Ramakrishna C T Ramkrishnan I V Ramakrishnan S A Smolka O Sokolsky E W Starck and D S Warren Fighting livelock in the i protocol A comparative study of veri cation tools In TACAS LNCS Springer*

*E A Emerson and J Y Halpern Sometimes and Not Never revisited on branching time versus linear time temporal logic Journal of the Association for Computing Machinery*

*E A Emerson and C L Lei Modalities for model checking Branching time strikes back Science of Computer Programming*

*R Gerth D Peled M Y Vardi and P Wolper Simple on the y automatic veri cation of linear temporal logic In Proceedings th Workshop on Protocol Speci cation Testing and Veri cation North Holland*

*Thomas A Henzinger Orna Kupferman and Shaz Qadeer From Pre historic to Post modern symbolic model checking In CAV LNCS Springer*

*G J Holzmann The model checker SPIN IEEE Trans on Software Engineering*

*H Iwashita and T Nakata CTL model checking based on forward state traversal In ICCAD ACM*

*H Iwashita T Nakata and F Hirose Forward model checking techniques oriented to buggy design In ICCAD ACM*

*A Kick Generierung von Gegenbeispielen und Zeugen bei der Modellpru fung PhD thesis Fakult at fu r Informatik Universitat Karlsruhe*

*D Kozen Results on the propositional calculus Theoretical Computer Science*

*O Lichtenstein and A Pnueli Checking that nite state concurrent programs satisfy their linear speci cation In Symposium on Principles of Programming Languages New York ACM*

*D E Long A Browne E M Clarke S Jha and W R Marrero An improved algorithm for the evaluation of xpoint expressions In CAV LNCS Springer*

*K L McMillan Symbolic Model Checking Kluwer*

*J P Quielle and J Sifakis Speci cation and veri cation of concurrent systems in CESAR In Proc th Int Symp in Programming*

*K Ravi K L McMillan T R Shiple and F Somenzi Approximation and decomposition of binary decision diagrams In DAC ACM*

*K Ravi and F Somenzi High density reachability analysis In ICCAD ACM*

*F Re el Modellpru fung von Unterlogiken von CTL Masterthesis Fakult at fu r Informatik Universitat Karlsruhe*

*M Y Vardi and P Wolper Reasoning about in nite computations Information and Computation*