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

*Concurrent Symbolic Veri cation of Liveness* Properties for Interleaved Models

*Felice Balarin*

*Cadence Berkeley Laboratories Berkeley CA USA*

*Abstract*

*Interleaved models of computations limit the number of system components that can change states simultaneously This interleaving constraint often decreases e ciency of symbolic veri cation methods It was shown previously that the constraint can be possibly partially removed while still preserving safety properties of systems We propose two extensions of this approach to liveness properties The rst one does not require changes to existing veri cation algorithms while the second which is usually more e cient does Our experiments indicate that both approaches can drastically reduce veri cation time*

*Introduction*

*Finite state models can be divided into two classes interleaved and concur rent In interleaved models sometimes also called asynchronous or disjunc tive a transition of the system corresponds to a transition of a single or a* small number of related components On the other hand in concurrent models sometimes also called synchronous or conjunctive or simultaneous a transition of the system corresponds to transitions of all or an arbitrary number of components

*The two classes di er in approaches to their veri cation Formal ver i cation tools based on interleaved models typically employ explicit state* space search while those based on concurrent models often use symbolic search While not fundamental this distinction has some practical justi

*cation On one side interleaved models are better for explicit search because*

*they require enumerating enabled transitions while concurrent models require* enumerating all subsets of enabled transitions On the other side the inter leaving constraint i e the requirement that only a single component executes at a time may cause arti cial correlation between components which may decrease the e ciency of the symbolic search

*Recently there have been several attempts to improve symbolic search for* interleaved models Most of these improvements are either based on extend ing partial order methods to symbolic search or on changing the search

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

*order to avoid the problem of arti cial correlation In we have analyzed* a di erent approach where a system in an interleaved model is analyzed as if its components were composed in a concurrent model We have shown that safety properties of systems can be veri ed even if the interleaving constraint is sometimes only partially removed We have also experimentally con rmed that removing the interleaving constraint increases the e ciency of the sym bolic search

*In this paper we extend the approach of to liveness properties We show* that it is possible to partially remove the interleaving constraint while check ing liveness properties with a standard veri cation algorithm We demonstrate that this partial removal can signi cantly improve the veri cation e ciency We also show that further improvements are possible if the veri cation algo rithm is modi ed so that the interleaving constraint is removed only at certain steps of the algorithm

*In the rest of this paper we rst review relevant existing concepts and* results in Section We propose two approach to veri cation of liveness prop erties in Section Some experimental results are presented in Section and

*nal comments are given in Section*

*Preliminaries*

*Notation*

*Formal veri cation algorithms require manipulating sets of states and relations* among states In this paper sets and relations are represented with character istic predicates over state variables We use bold face capitals e g I R T to denote predicates and lower case bold face letters to denote variables Also we use fSg to denote the set that some predicate S characterizes i e

*def*

*fSg fs j s satis es Sg*

*Model*

*Next we present an extension to in nite executions of the interleaved model of* computation proposed in In this formalism a system is a nite collection of processes Each process Pi is given by

*the present state variable pi and the next state variable ni both ranging over some nite set of local states we use p n to denote the vector containing all present next state variables and refer to their valuations as global states*

*the initial states predicate Ii in variable pi*

*the transition relation predicate Ti in variables p and ni*

*Note that Ti can depend on all present state variables in p i e the pro cesses communicate by observing states of other processes Consequently*

*elements of fTig are pairs p n where p is a global state itself a vector with* one component for each process and n is a local state of Pi

*An execution of the system is any in nite sequence s s of global* states such that s fI g for all processes P and for every j there exists a process Pi such that all components of sj and sj except the i th are the same and sj sj i fTig

*i i* *i*

*Intuitively an execution of the system proceeds in a series of steps such* that in every step a single process makes a transition If sj and sj di er in the i th component then Pi is the process that makes the transition in the j th step and the change in the corresponding local state must be consistent with its transition relation

*With every system we associate set F of fairness constraints Each fairness* constraint is a pair of predicates over p An execution s s is said to be fair if for every U V F the set of states appearing in the execution in nitely often is either contained in fUg or intersects fVg

*For example if fUg contains all states in which some service is not re quested and fVg contains states in which that service is granted the con straint U V would require that the service is granted in nitely often if it is* requested in nitely often

*Given a system consisting of processes Pi we say that a composition of* Pi s is any process that has vector p of the present state variables of Pi s as its present state variable vector n as its next state variable and the initial states predicate Ii In particular we de ne the interleaved composition Pk

*i*

*to be the composition with the transition relation predicate*

*Tk*

*i*

*Ti pj nj j i*

*and the concurrent composition P to be the composition with the transition relation predicate*

*T*

*i*

*Ti pi ni*

*Intuitively while Tk requires that at every step exactly one process exe cutes T allows any number of processes including zero to execute in any* step It is not hard to see that a collection of processes has the same executions as their interleaved composition

*Veri cation of liveness properties*

*First we need some de nitions Given some predicates S in variable p and T* in variables p and n we use N ext S T to denote the predicate in p that

*We use sj i to denote the component of global state sj corresponding to process Pi i e we assume that sj sj sj sj n where n is the number of processes in the system*

*characterizes the set of states reachable in one step by T from S Formally fN ext S T g fs j q fSg q s fTgg*

*Similarly we use Reach S T to denote the predicate that characterizes the*

*set of states reachable by T from S in any number of steps Formally fReach S T g is the smallest set satisfying*

*fReach S T g fSg fN ext Reach S T T g*

*Computing these predicates is a standard and crucial part of symbolic ap proaches to formal veri cation*

*Now we are ready to introduce a veri cation algorithm Veri cation of* many liveness properties can be reduced to checking for the existence of a fair execution Probably the simplest algorithm for this check is as follows

*Algorithm FAIR*

*F Reach Vi Ii Tk*

*repeat*

*F F N ext F Tk for each U V F*

*F F U Reach V F Tk rof*

*until convergence*

*The set fFg is an increasingly better over approximation of the set of fair* states We say that a state is fair if it appears in nitely often in some fair execution In step we remove from fFg all the states that cannot be reached from any other state in fFg Since every fair state must belong to a cycle of fair states no fair states are eliminated in this step In step we remove from fFg states which are neither in fUg nor can be reached from any fair state in fVg It is not hard to check that no such state is fair At the end fFg is empty if and only if the set of fair states is

*Concurrent approach to veri cation of safety properties*

*Verifying safety properties of systems usually reduces to reachability analysis It was observed in that it is often cheaper to compute Reach S T than* Reach S Tk However only

*Reach S Tk Reach S T*

*holds in general so replacing Tk with T may lead to incorrect veri cation* results In we have shown that Reach S Tk is equal to Reach S T if T is serializable i e if for every transition s q appearing in some execution of P there exists a sequence of states s sN such that

*s s sN q N jfPi j si qigj*

*there exists an ordering Pi PiN of processes in fPi j si qig such that for every j N all components of sj and sj except ij th are equal*

*and sj sj ij fTij g*

*We say that s sN is a serialization of s q Note that if some state* *v appears in a serialization of s q then v must be between s and q i e either vi si or vi qi must hold for every i*

*Even if T is not serializable it is always possible to modify it yielding*

*T such that T*

*is serializable and fT g fT g fT*

*g For such a modi*

*k*

*cation Reach S T Reach S T holds Therefore without a loss of*

*k*

*generality in the rest of this paper we will assume that T is serializable*

# *Concurrent Approach to Veri cation of Liveness Prop erties*

*In this section we propose two approaches to veri cation of liveness properties* for interleaved models Both approaches are based on relaxing the interleav ing constraint The rst approach called external does not require modifying existing veri cation algorithms but it may require that the interleaving con straint is only partially relaxed The second approach called internal is a modi cation of algorithm FAIR In the proposed modi cation the interleaving constraint is fully removed but only in some steps of FAIR In general the internal approach is more e cient but the external approach may be appro priate if an existing veri cation tool is available but impossible or hard to modify

*External Approach*

*Ideally P has fair executions if and only if Pk does It is not hard to see that* every fair execution of Pk is also a fair execution P For the desired relation to hold we need to show that for every fair execution of P there exists a related fair execution of Pk Unfortunately this is not generally true even if T is serializable

*Following the approach taken in for the nite case from a given fair* execution r of P we may try to construct a fair execution r of Pk by

*i removing from r all states sj such that sj sj i fTig for any process* Pi Such cases are possible because T allows that in a given step none of the processes make a transition Obviously sj sj must hold in this case so these transitions are called arti cial self loops

*ii replacing every remaining transition with its serialization This approach may not work for two reasons*

*after the rst step there may be only nitely many states left recall that*

*we require all executions to be in nite*

*expansion in the second step may cause the execution to violate some fair ness constraints*

*P* 1 *Pi i N* 1 *PN*



1

true

0



*p*

*pi*

. . . . . .

true

*pi*

0

*pN*

0

*Fig A system with N processes*

*Let us now examine the problem of arti cial self loops more closely Con sider the system shown in Figure A typical process Pi starts in the initial* state moves to if its left neighbor is in state and then moves back to if its right neighbor is in state Boundary processes P and PN are similar except that their state changes are unconditional if their neighbors do not exist Assume that the system in Figure has a single fairness constraint with U f alse and

*fVg f g*

*In other words an execution is fair if it visits in nitely often the state in which* all even numbered processes are in state and all odd numbered processes are in state Such a state is reachable in Pk However if N is even then that state is a deadlock state odd numbered processes cannot move from because their right neighbors are not in and even numbered processes cannot move from because their left neighbors are not in Therefore Pk has no fair executions On the other hand P does have a fair execution It consists of a path to the state in fVg and then looping in it forever

*To solve this problem we need to eliminate all arti cial self loops from* fT g This can be accomplished by replacing T with

*T Ti i*

*The modi ed T requires that at least one process makes a transition in any*

*given step In the special case when Pi s do not have any self loops which* occurs often for interleaved models is equivalent to

*T pi ni*

*i*

*Even though and are equivalent assuming Pi s have no self loops computing typically requires simpler intermediate results*

*Let us now focus on the problem of fairness constraint violation Again consider the example in Figure but this time assume that the single fairness* constraint has V f alse and

*N*

*fUg f s sn j X si is even g*

*i*

*In other words an execution is fair if the number of processes in state is odd* only nitely many times Since in every step of Pk exactly one process moves

*from or to state it is obvious that Pk has no fair executions However P does have fair executions even if T is modi ed to eliminate arti cial* self loops For example an in nite loop consisting of states and

*is fair This discrepancy occurs because states occuring in nitely* often in the execution of P are all contained in fUg but in every serialization of that execution some state outside of fUg appears in nitely often

*We can solve this problem by eliminating from fT g all transitions s q such that for some fairness constraint U V both s and q are in fUg but* every serialization of s q visits some state not in fUg This can be accom plished by replacing T with

*T*

*U V F*

*U Un q B Uq*

*where Un and Uq denote the predicates obtained from U by replacing every*

*occurrence of p with n and q respectively and fBg contains all states between*

*p and n More precisely*

*B qi pi qi ni i*

*Note that eliminates more transitions than it is necessary It eliminates* all transitions s q such that for some fairness constraint U V both s and q are in fUg but not all the states that could appear in a serialization of

*s q are in fUg Eliminating more transitions may diminish the advantages* of the concurrent approach Still we have chosen this approximation of the desired set of transitions because computing better approximations appears to be signi cantly more expensive Also note that even though eliminates more transitions than it is necessary it is still correct i e it never eliminates any transition from fTkg

*An important special case arises when all the predicates in fairness con straints depend on at most one variable It is not hard to check that in that* case leaves T unchanged This special case is quite common fairness constraints are typically just a union of component constraints which depend only on local states

*If T is modi ed as in and then P has fair executions if and only* if Pk does In most symbolic formal veri cation systems and can be implemented as additional monitor processes and do not require changing the internals of the program

*Internal approach*

*In this section we propose to remove the interleaving constraint in some steps* of algorithm FAIR Steps and of FAIR require reachability computation Since Reach S T Reach S Tk for any S assuming T is serializable we can safely replace Tk with T in steps and of FAIR We expect this modi cation to improve e ciency for the reasons which are explained in details

*Table*

*Experimental results*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *N* | *FAIR*  *size time* | | *external*  *size time* | | *internal*  *size time* | |
|  |  | *s* |  | *s* |  | *s* |
|  |  | *s* |  | *s* |  | *s* |
|  |  | *s* |  | *s* |  | *s* |
|  |  | *s* |  | *s* |  | *s* |
|  |  | *s* |  | *s* |  | *s* |
|  |  |  |  | *s* |  | *s* |
|  |  |  |  | *s* |  | *s* |

*in In Section we will show that these expectations are experimentally* con rmed

*Replacing Tk with T in step of algorithm FAIR may lead to incorrect* veri cation results Moreover it is not likely that such a modi cation would result in any improvement To reduce the number of iterations it is useful that as many states as possible are eliminated from fFg in step Since fT g fT g using T instead of T in step may only result in fewer states being eliminated and increasing number of iterations

*k k*

*Finally let us note that even though we consider only algorithm FAIR* here a similar analysis can be applied to other algorithms for checking for the existence of a fair execution

# *Experiments*

*We have tested the proposed approaches on the example from Figure with* a single fairness constraint de ned by and The results are summa rized in Table The rst column shows the number of processes Next two columns correspond to algorithm FAIR followed by two columns correspond ing to the external approach described in Section and nally two columns corresponding to the internal approach described in Section For each approach we give the run time and the size in terms of the number of BDD nodes of the relevant transition relation Tk for FAIR T for the internal approach and T modi ed according to and for the external approach All experiments were performed using CUDD package by Somenzi

*The rst thing to notice in Table is that T is smaller than Tk How ever if T is modi ed as required by the external approach then it becomes* larger than Tk Overall these di erences in sizes are relatively small and all three transition relations grow only linearly with the number of processes

*The di erence in run times is much more dramatic Both the internal and the external approach outperform FAIR by several orders of magnitude and the di erence seems to be growing with the number of processes Table also shows that the internal approach signi cantly outperforms the external approach but the precise relation is hard to determine because for all but largest experiments the run times of the internal approach are too small to measure precisely*

*Conclusions and Future Work*

*We have shown that relaxing the interleaving constraint may improve e ciency of symbolic veri cation of liveness properties We have proposed two approaches to this end One appears to be more e cient but the other has an advantage that it can be used with existing veri cation systems without any internal modi cations*

*Future work should include more experiments to better understand when the proposed approaches are bene cial and what are relative merits of each Another interesting problem is extending these approaches to model checking for branching time temporal logics*

*References*

*Rajeev Alur R K Brayton T A Henzinger S Qadeer et al Partial order reduction in symbolic state space exploration In O Grumberg editor Proceedings of Computer Aided Veri cation th International Conference CAV Haifa Israel June Springer LNCS vol*

*Felice Balarin Correctness of the concurrent approach to symbolic veri cation of interleaved models In Alan Hu and Moshe Vardi editors Proceedings of Computer Aided Veri cation th International Conference CAV Vancouver BC Canada June July Springer*

*R K Brayton A Sangiovanni Vincentelli G D Hachtel F Somenzi A Aziz S T Cheng S Edwards S Khatri Y Kukimoto S Qadeer R K Ranjan T R Shiple G Swamy T Villa A Pardo and S Sarwary VIS A system for veri cation and synthesis In Rajeev Alur and Thomas A Henzinger editors Proceedings of Computer Aided Veri cation th International Conference CAV Rutgers NJ July Springer LNCS vol*

*J R Burch Edmund M Clarke David E Long Ken L McMillan and David L Dill Symbolic model checking for sequential circuit veri cation IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*

*April*

*D L Dill A J Drexler A J Hu and C H Yang Protocol Veri cation as a Hardware Design Aid In Proceedings of ICCD pages October*

*E Allan Emerson and C L Lei E cient model checking in fragments of the propositional mu calculus extended abstract In Proceedings Symposium on Logic in Computer Science pages Cambridge Massachusetts June IEEE Computer Society*

*Z Har El and R P Kurshan Software for analysis of coordination In Proceedings of the International Conference on System Science pages*

*Ramin Hojati and Robert K Brayton An environment for formal veri cation based on symbolic computation Formal Methods in System Design An International Journal March*

*Gerard J Holzmann Design and validation of computer protocols Englewood Cli s N J Prentice Hall*

*R P Kurshan V Levin M Minea D Peled et al Verifying hardware in its software context In Digest of Technical Papers of the IEEE International Conference on CAD pages November*

*Zohar Manna and Anir Pnueli Veri cation of concurrent programs The temporal framework In R Boyer and J Moore editors Correctness Problem in Computer Science pages Academic Press*

*Kenneth L McMillan Symbolic Model Checking Kluwer Academic Publishers*

*F Somenzi CUDD CU Decision Diagram Package June User s Manual*