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

*Model Checking Semi Continuous Time Models* Using BDDs

# *S ergio Campos a M arcio Teixeira a Marius Minea b* Andreas Kuehlmann c Edmund Clarke b

*a Univ Federal de Minas Gerais Dept de Ci encia da Computac*

*fscampos mtog dcc ufmg br*

*a o Brasil*

*b Carnegie Mellon University School of Computer Science USA fmarius emcg cs cmu edu*

*c IBM T J Watson Research Center USA kuehl watson ibm com*

*Abstract*

*The veri cation of timed systems is extremely important but also extremely di cult Several methods have been proposed to assist in this task including extensions to symbolic model checking One possible use of model checking to analyze timed systems is by modeling passage of time as the number of taken transitions and ap plying quantitative algorithms to determine the timing parameters of the system The advantage of this method is its simplicity and e ciency In this paper we extend this technique in two ways First we present new quantitative algorithms that are more e cient than their predecessors The new algorithms determine the number of occurrences of events in all paths between a set of starting states and a set of nal states We then use these algorithms to introduce a new model of time in which the passage of time is dissociated from the occurrence of events With this new model it is possible to verify systems that were previously thought to require dense time models We use the new method to verify two such examples previously analyzed by the HyTech tool a steam boiler example and a fuel injection controller*

*Introduction*

*Computers are frequently used in applications where failures can have severe consequences such as in the control of industrial machinery or transportation equipment In these applications the computer system must not only produce the correct result but must do so in timely fashion For example a command to apply the brakes of a car or to turn an airplane to a certain direction cannot be late otherwise an accident may occur Such failures cannot be tolerated making the correctness of these systems an extremely important issue*

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

*However veri cation of such systems is a very complex problem made* even harder by timing requirements Several methods have been proposed to accomplish this task One method that has obtained signi cant success is model checking In this technique the system being veri ed is modeled as a state transition graph and properties of the system are expressed as temporal logic formulas The veri cation procedure consists of a search on the state space of the graph to determine which states satisfy the properties

*Original model checkers were not designed to verify timing characteristics Several extensions have been proposed to express and verify such properties The rst and simplest is to associate each transition with the passage of one* time unit and to determine elapsed time by counting the number of transitions between events This technique assumes a discrete time model The main ad vantage is its simplicity and extremely e ciently implementation particularly in BDD based symbolic model checkers such as SMV or Verus

*Another approach is to use a continuous time model in which events can* happen at any moment in the dense time domain e g timed automata Since in this case the state space is inherently in nite model checking entails constructing a nite equivalent model the complexity of which can be quite high These models as well as the veri cation algorithms are considerably more complex than in the discrete time case Initial tools were unable to handle models with more than hundreds or thousands of states Current tools are signi cantly more e cient but verifying timed automata is still much more expensive than the veri cation of discrete time models

*However discrete time models have one major disadvantage over contin uous time models their limitation in expressing the semantics of event se quences that happen in short periods of time For example if the occurrence* of an event a triggers an alarm b and an immediate response c we can model these events as happening simultaneously or taking at least two time units to occur This may not correspond to reality however It may be the case that after event a has occurred but before alarm b another event d occurs that would change response c But if a b and c happen at the same time this possibility would not be present On the other hand if it takes time unit between a and b it would not be possible for d to occur between a and b For this reason discrete time models cannot be used in some applications where accuracy is essential

*The proposed method overcomes this problem by using zero length transi tions to model the occurrences of events without time passing The passage* of time then occurs in discrete steps using unit length transitions The advan tage of this new model is that it removes the limitation on event orderings for the discrete time model For example it is now possible to let events a b c and d described previously occur in time zero preserving their order and only let time elapse after all events have occurred We argue that this enables the veri cation of many systems that have been previously thought to require dense time models

*In order to determine the time between events in the semi continuous time* model we use quantitative timing analysis as described in Of partic ular interest are the condition counting algorithms that count the minimum and maximum number of occurrences of a speci c event in a given set of in tervals In this work these algorithms are used to count the minimum and maximum number of unit transitions on paths of interest computing the time elapsed between events We propose new condition counting algorithms that are signi cantly more e cient than the previous ones These algorithms allow veri cation to be done as e ciently as for the simple discrete time case They are similar to the xpoint computation used in model checking for untimed systems and as such can be implemented e ciently using BDDs

*To demonstrate the expressive power and e ciency of the method we have* veri ed two examples of systems in which high accuracy is necessary to achieve the correct results The rst is the steam boiler example described in This example while small demonstrates that the proposed model can be used to verify systems which are not usually considered in the realms of discrete time We have then veri ed an automotive engine controller developed for Magneti Marelli that has been previously veri ed by HyTech We have modeled the controller that identi es that the driver has released the accelerator and regulates the reduction of fuel injection This identi cation is a complex time critical function of the position of several sensors If the timing of the events that take place during its execution is wrong the algorithm may not converge and the controller can malfunction We have veri ed both examples using Verus demonstrating the e ectiveness of the proposed method

*Related Work*

*A precursor to the presented analysis method has been developed in the real time model checker Verus This tool implements quantitative timing analy sis algorithms that determine the timing characteristics of a system by count ing the time between events or the number of occurrences of events in given* intervals The method has been used to verify large and complex timed sys tems such as an aircraft controller a robotics controller and the PCI local bus However the condition counting algorithms used in that context require the augmentation of the state space with a additional integer time vari able which added a signi cant overhead to veri cation The new algorithms do not require this construct and are e ciently implemented using BDDs

*The occurrence of events without the passage of time has been discussed* in But that work does not consider a symbolic implementation using BDDs and is not as e cient It also does not use quantitative analysis algorithms and cannot generate the same type of information as the method proposed

*A signi cant body of research exists on continuous time models One of* the most widely used models are the timed automata which add real valued clock variables to represent time Clocks evolve at the same rate modeling

*passage of time and formulas can refer to the value of the clocks to express* timing properties Veri cation is then performed on a nite state quotient model such as the region graph or the zone automaton However the expressive accuracy comes with a signi cant increase in complexity and a signi cant e ort in the development of continuous time model checkers has been devoted to dealing with the state explosion problem

*The expressiveness and e ciency trade o s between discrete and continu ous time raise the question when a discrete time approximation is su cient to* model all continuous time behaviors of a system This problem is analyzed e g in This work introduces the notion of digitizability and proves that such a reduction is possible for timed transition systems for veri cation of properties such as time bounded invariance and time bounded response More recent work shows that a reduction to discrete time can be performed for acyclic combinational circuits but not for all cyclic ones These can only be reduced under the constraint that no strict inequality is used in their design

# *Condition Counting Algorithms*

*Our method relies on the ability to count some transitions on a path but not* necessarily all of them In order to accomplish this we use the algorithms described in this section The original algorithms used in our method to verify real time systems determine the length of a path leading from a set of starting states to a set of nal states But to verify semi continuous time models we also need to compute the minimum and maximum number of times a given condition holds on any path from start to nal In we have presented algorithms that compute this information However these algorithms required an augmentation of the state space with a counter to store intermediate results This made the algorithms very expensive in some cases The algorithms described in this section do not su er from this limitation

*We require that every state of the model has at least one outgoing tran sition We also assume that any path beginning in start reaches a state in*

*nal in a nite number of steps This is necessary so that the minimum and* maximum are well de ned It can be checked using the maximum algorithm described in We also consider only reachable states which can be achieved by intersecting start with the set of reachable states computed a priori

*Minimum Condition Counting*

*The minimum condition count algorithm computes the minimum number of* states satisfying a given condition cond over all paths that start in a state in start and end in a state in nal Any paths starting in start but which do not reach nal in a nite number of steps are excluded from this computation In particular if no path from start ever reaches nal the algorithm will return the special value NOPATH

*The algorithm looks for paths beginning in start that have an increasing* number of occurrences of cond Each iteration consists of two phases The rst is a forward traversal through states that do not satisfy cond This traversal is performed until all states not satisfying cond reachable from the current frontier are found If nal has not been reached yet the frontier is expanded by one step to states that satisfy cond and the condition counter is incremented The algorithm iterates until nal is found or all reachable states are visited The algorithm must di erentiate between states that do not satisfy cond and those that do and similarly between transitions leading to these states We use subscripts and respectively for the two types of states and transi tions For example start is the set of initial states that do not satisfy cond

*and start is the set of initial states that satisfy cond*

*start start cond start start cond*

*Furthermore if N s s is the transition relation we denote by T S and* T S the set of transitions from a state in S that lead to states not satisfying cond and to states satisfying cond respectively

*T S fs j s S N s s s condg T S fs j s S N s s s condg*

*The argument about the correctness of the algorithm follows from invari ants stating that R at the ith iteration contains the set of all states that can be* reached as endpoints of nite intervals starting in start have no state in nal

*except perhaps the last one and having i or less states satisfying condition*

*The proof can be found in the full version of the paper*

*Maximum Condition Counting*

*The maximum condition count algorithm computes the maximum number of* states satisfying a given condition cond over all paths that begin in a state in start and end in a state in nal without previously traversing a state in nal If there is a path beginning in start that goes through cond in nitely often without reaching nal the algorithm returns in nity The basic idea behind the algorithm is to nd paths with increasing condition count whose states are all within nal The condition count of the longest path satisfying this condition and starting in start is the desired maximum

*Similarly to the mincount algorithm we consider transitions into states* that satisfy cond and that do not satisfy cond separately This algorithm however performs a backward search and uses the reverse image of the tran sition relation In this case B S is the set of states satisfying neither cond nor nal that lead to a state in S in one step Similarly B S is the set of states satisfying cond but not f inal that lead to a state in S in one step Note that nal only appears implicitly in the algorithm in the de nitions of B and B

*B S fs j s S N s s s f inal s condg*

*B S fs j s S N s s s f inal s condg*

*proc mincount start cond f inal i R R start*

*do*

*do*

*if R f inal return i R R*

*R T R R while R R*

*R T R R*

*if i R R start i i*

*while R R return NOPATH*

*proc maxcount start cond f inal i R cond*

*do*

*R R do*

*R R*

*R R B R while R R*

*if R start return i R B R*

*i i while R R return*

*Fig Minimum and maximum condition count algorithms*

*Again we argue the correctness of the algorithm using an invariant similar* to the previous one It states that at the ith iteration R is the set of all states that are the start of a nite path which has no states in nal except possibly the last one and which has i states that belong to cond The proof can be found in the full version of the paper

*Semi Continuous Time*

*The basic idea of the proposed method is to allow zero length transitions* that model the occurrence of events without time passing thus making the occurrence of events independent of the passage of time To allow zero length transitions we have created a special variable t in the model of the system being veri ed Time passage is controlled by enabling unit length transitions only when t is true and enabling zero length transitions only when t is f alse Parallel composition of processes under the new model is de ned as fol

*lows Unit transitions have to occur synchronously that is all processes must*

*execute a unit transition in order for time to elapse Zero length transitions on the other hand occur asynchronously When a process performs a zero length transition all other processes are not executing As a consequence of* this zero length transitions are always enabled Unit transitions however are only enabled when there is at least one unit transition enabled in each process This parallel composition model satis es one important invariant passage of time is identical in all processes

*A symbolic implementation of this parallel composition model is straight forward given the traditional parallel composition algorithms used in BDD based tools conjunction of transition relations for synchronous composition* and disjunction for asynchronous composition

*Under the new model we must rst di erentiate between unit and zero length transitions Given T Ra we de ne T R a T R a as the transition re lation for zero length unit transitions in Pa We can then de ne the global* transition relation for a model with processes Pa and Pb as

*TR T R a T R b T R a T R b*

*From this expression we can see that whenever unit transitions are enabled* in all processes they are also enabled in the composed model The expression also guarantees that zero length transitions enabled in some process are also always enabled in the composed model The only other condition that must be imposed in this model is that time eventually change This can be ensured by forbidding zero length loops which can be enforced by a syntactic check

*To determine how much time has elapsed between events we use the con dition counting algorithms For example mincount a t true b determines* the minimum time between events a and b Similarly the maxcount algorithm can be used to determine the longest time between a and b

## *Expressive Power of the Proposed Method*

*The proposed method does not have the same expressive power as a dense* time model Our method uses a di erent discretization of dense time but the nal model is still discrete It has been proven that there exist systems which cannot be discretized without changing their behavior In it is shown that the following circuit has behaviors that cannot be captured by any discretization It has four signals x x x and x and transitions which assign values to them as x x x x and x x Each transition takes time between and units to occur Let t t and t be the times when each transition occurs A possible behavior of the circuit could have transitions times satisfying t t t In a discrete time model the only values allowed for ti are or it is impossible to assign three di erent values for t t and t This behavior cannot exist in a discrete model

*The result of is that only models without strict inequalities can be* guaranteed to be discretized correctly Only a weaker notion of behavior preservation can be maintained during discretization It is possible that events

*that occur at di erent time instants in the dense time model occur at the same* time instant in the discretized model This is also true for our model It is frequently argued that because of this problem systems modeled using discrete time cannot capture the essential properties of a design We argue however that the key feature is not an arbitrary accuracy for the representation of t t and t but rather an appropriate discretization together with their ordering In fact in the commonly used continuous time models the constants used in specifying properties can only be integers and exact values for the timepoints ti are not expressible

*With the use of transitions that take zero time to occur our method pro vides a way of preserving the same ordering of events as dense time models We claim then that the essential properties of a design are preserved by our* method in a similar way as by methods that use dense time For example one property that would capture the behavior above can be written in CTL augmented with the freeze operator described in as where ei is the event corresponding to the transition of signal xi

*x e EF y e EF z e x y z This can be expressed in our method by the property*

*e e E t U e e t E t U e*

*where t is true in unit length transitions and false in zero length ones Frequently the fact that the total time elapsed is less than one time unit*

*is not encoded in the formula In this case the formula can be simpli ed to*

*e e EF e e EF e*

*One important consideration is that this property can be veri ed using dis crete time models by simply doubling the time quantum This is implemented* by changing all transitions into two consecutive ones that is one transition in the new model takes half a unit instead of one unit This however has two serious problems One it adds a signi cant overhead to veri cation The second one is that it is not possible to know by how much we should decrease the time quantum because in general there is no way to nd out when events that happened in di erent times have been considered simultaneous by the model Because of this we cannot determine when the results of a veri cation using discrete time would be di erent if the model was re ned Our method does not su er from these problems There is no signi cant overhead added since only one additional variable is created in the model and all possible ordering of events are represented in the model making re nements in the time quantum unnecessary

## *Examples*

*Steam Boiler*

*In order to demonstrate the expressive power of our method we have veri ed* the steam boiler example described in Steam boilers are mostly used in

*thermoeletrical power plants It is extremely important to keep a steam boiler* working correctly since any malfunction may cause an accident with serious consequences The system modeled consists of a water tank two pumps and sensors that measure the pumping rates the steam evacuation rate and the water level A controller oversees the operation of the system The controller must guarantee that the water level is always between two values M and M at all times and should try to maintain water level between the normal operating levels N and N as much as possible The controller and the phys ical plant communicate in discrete intervals once every seconds During each communication phase all units send information to the controller which responds by sending messages to the units All communication takes place instantaneously

*The controller decides to turn the pumps on or o based on the water* level w The two pumps need ve seconds to start pumping water in the tank because of the high pressure inside the tank The pumps are turned o immediately after receiving a message to stop pumping from the controller Four values are used by the controller to decide how many pumps should be active Depending on these values and the current water level the controller turns one or both valves on or o details about the system can found in We have modeled the high level interactions between discrete control decisions and the continuous aspects of the underlying physical plant We concentrate on the continuous aspects of the system and their modeling with the method described in the previous section

*We have set the values of the system constants as follows sampling time*

*seconds maximal steam rate W liters per second pumping capacity* P liters per second interval of normal water levels N N liters interval of acceptable water levels M M liters These constants have the same values as in allowing direct comparison of results In our model unit transitions model the passage of one second and zero length transitions are used to model nondeterministic events and decisions taken by the controller Notice that veri cation can be performed very accurately even though we use a coarse discretization of time

*The most important property of the steam boiler is that the water level is* always between M and M We also require that the emergency stop mode is never entered Therefore the unsafe states are those that satisfy the formula

*w M w M emergency stop Using Verus we have been able to*

*verify that the controller maintains the water level within the required bounds This result is the same obtained in The veri cation took seconds and*

*MBytes of memory on a Pentium II system*

*We have also veri ed other properties of the steam boiler using the min count and maxcount algorithms For example an important parameter of the* system is the size of the frequency of communication between controller and plant Using Verus we have been able to determine that also sat is es the safety requirements but does not If communication between

*controller and units is delayed by up to one second safety is maintained but* longer delays can cause safety problems Several other parameters have been identi ed including e g the minimum and maximum times needed for water to go from the minimum to the maximum level The interval is sec onds meaning that the water may never reach the maximum water level from the minimum water level but it never takes less then seconds

*Automotive Engine Controller in Cuto Model*

*In order to demonstrate the e ciency of the method we have veri ed an au tomotive engine controller in cuto mode described in and veri ed by* HyTech We have studied the cuto mode where we consider control of the engine once the driver has released the accelerator pedal The system must then guarantee that the engine will deliver zero torque within a certain time The control objective is to reach injection cuto while minimizing acceleration discomfort If fuel injection is abruptly cut o the vehicle may exhibit very undesirable acceleration oscillations If fuel injection remains on for a long time the car does not decelerate In order to minimize these problems the controller makes intelligent decisions about when and how to cut o fuel

*The system consists of the engine which includes the driveline and the* cylinders and its controller The engine has four cylinders each of which cycles in lockstep through four phases in the following order intake I com pression C expansion E and exhaust The controller must make its deci sion on injection modeled by the binary output variable j at the beginning of the preceding exhaust phase If fuel is injected into a cylinder the cylinder produces torque on its next expansion phase Thus the driveline does not react to a control decision until three phases later

*The controller sets the value of j at each phase change with the function* F modeling the decision to inject fuel or not The function F is de ned over a transformed state space over the variables x x x x that helps isolate the fundamental modes related to acceleration oscillations Powertrain oscillations are due to the pair of complex conjugate poles which are related to x and x components Thus our analysis concentrates on the x x subspace where the encirclements of the origin correspond to oscillations more details about the system can be found in

*The automotive engine controller should meet the requirement that for a* given initial condition the state is close to the origin injection cuto within a bounded number of phases convergence To show the convergence re quirement using the same parameters described in we have computed the maximum time from an initial state until a trajectory is close to the origin

*We have used Verus to verify the requirements The code for the example* has been generated automatically from the HyTech original code using a perl script written for this purpose We have divided the x x state space into

*x partitions increasing the accuracy of the rectangular approximations*

*In our model phase changes occur in unit time and all other events happen in time zero We have determined that the maximum time until a trajectory is close to the origin is steps the same result obtained by HyTech Veri cation was performed very e ciently but at the same time it has shown a limitation of our method The source le for this example is extremely large it has more than lines of Verus code It is to the authors knowledge the largest example veri ed by symbolic model checking It took Verus several hours to compile this code into a transition graph representing the system Once the model was generated however veri cation was performed in only seconds The reason for the long compilation time seems to be related to the fact that for systems which involve large constants discretization can lead to a large state space representation even when using BDDs This is caused by the binary encoding of integer values used In some of these cases continuous time models may be more e cient since the representation is less dependent on time granularity However for models whose timing constants are well behaved a discrete time model with a uniform BDD based representation can present signi cant gains in e ciency In this case it seems that both e ects were present The values represented for x and x are well behaved but their values are large as well as the number of operations that have to be performed on them making the generation of the model slow but possible Veri cation on the other hand was performed extremely fast showing that the complexity is related to the manipulation of large integer values not to*

*the representation of time*

# *Conclusions*

*In this work we propose a new algorithm to perform quantitative timing analy sis of models that is more e cient than its predecessor This algorithm called condition counting counts the minimum and maximum number of occurrences of events between two events start and nal The algorithm is used to imple ment an alternative method to represent time which enables the veri cation of systems that were previously considered to require dense time models Veri cation under the new model can be performed as e ciently as for discrete time models The proposed method has been implemented in Verus but it can be used in most BDD based symbolic model checkers Two examples that had previously been veri ed by the dense time tool HyTech have been modeled and veri ed in Verus Future work includes a more accurate characterization of the expressive power of the method*

# *Acknowledgments*

*We would like to thank Howard Wong Toi for the many useful discussions about the examples that have been veri ed in HyTech*

*References*

*Rajeev Alur Costas Courcoubetis and David Dill Model checking for real time systems In Proc th Annual IEEE Symp on Logic in Computer Science pages Philadelphia PA USA June IEEE Press*

*Eugene Asarin Oded Maler and Amir Pnueli On discretization of delays in timed automata and digital circuits In D Sangiorgi and R de Simone editors CONCUR Concurrency Theory th Int Conf Proc volume of LNCS pages Nice France September Springer*

*S Campos E Clarke W Marrero and M Minea Verifying the performance of the PCI local bus using symbolic techniques In Proc IEEE Int Conf on Comput Design pages Austin TX USA October IEEE Press*

*S V Campos A Quantitative Approach to the Formal Veri cation of Real Time Systems PhD thesis School of Computer Science Carnegie MellonUniv*

*S V Campos E M Clarke W Marrero and M Minea Timing analysis of industrial real time systems In Proc Workshop on Industrial strength Formal Speci cation Techniques pages Boca Raton FL April IEEE Press*

*S V Campos E M Clarke W Marrero M Minea and H Hiraishi Computing quantitative characteristics of nite state real time systems In Proc th IEEE Real Time Systems Symp pages San Juan Puerto Rico December IEEE Press*

*E M Clarke and E A Emerson Design and synthesis of synchronization skeletons using branching time temporal logic In Logic of Programs Workshop volume of LNCS pages Yorktown Heights NY USA Springer*

*E M Clarke E A Emerson and A P Sistla Automatic veri cation of nite state concurrent systems using temporal logic speci cations ACM Transactions on Programming Languages and Systems*

*H De Leon and O Grumberg Modular abstractions for verifying real time distributed systems Formal Methods in System Design*

*David L Dill Timing assumptions and veri cation of nite state concurrent systems In J Sifakis editor Proceedings of the International Workshop on Automatic Veri cation Methods for Finite State Systems volume of LNCS pages Grenoble France June Springer*

*T A Henzinger P H Ho and H Wong Toi HyTech the next generation In Proc th IEEE Real Time Systems Symp pages Pisa Italy December*

*IEEE Press*

*Thomas A Henzinger Zohar Manna and Amir Pnueli What good are digital clocks In W Kuich editor Automata Languages and Programming th*

*International Colloquium Proceedings volume of LNCS pages Wien Austria July Springer*

*Thomas A Henzinger Xavier Nicollin Joseph Sifakis and Sergio Yovine Symbolic model checking for real time systems In Proc th Annual IEEE Symp on Logic in Computer Science pages Santa Cruz CA USA June IEEE Press*

*Thomas A Henzinger and Howard Wong Toi Using HyTech to synthesize control parameters for a steam boiler In Formal Methods for Industrial Applications Specifying and Programming the Steam Boiler Control volume*

*of LNCS pages Springer*

*K G Larsen P Pettersson and W Yi Compositional and symbolic model checking of real time systems In Proc th IEEE Real Time Systems Symp pages Pisa Italy December IEEE Press*

*K L McMillan Symbolic Model Checking Kluwer Academic Publishers*

*Tiziano Villa Howard Wong Toi Andrea Balluchi Joerg Preussig Alberto Sangiovanni Vincentelli and Yosinori Watanabe Formal veri cation of an automotive engine controller in cuto mode In CDC IEEE Conference on Decision and Control Tampa Florida December*

*S Yovine Kronos A veri cation tool for real time systems Springer International Journal of Software Tools for Technology Transfer October*