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

*Fully Symbolic Model Checking of* Timed Systems using

*Di erence Decision Diagrams*

# *Jesper M ller Jakob Lichtenberg Henrik R Andersen and* Henrik Hulgaard

*The IT University in Copenhagen Glentevej*

*DK Copenhagen NV Denmark E mail jm jl hra henrik itu dk*

*Abstract*

*Current approaches for analyzing timed systems are based on an explicit enumera tion of the discrete states and thus these techniques are only capable of analyzing systems with a handful of timers and a few thousand states We address this lim itation by describing how to analyze a timed system fully symbolically i e by representing sets of discrete states and their associated timing information implic itly We demonstrate the e ciency of the symbolic technique by computing the set of reachable states for a non trivial timed system and compare the results with the state of the art tools Kronos and Uppaal With an implementation based on dif ference decision diagrams the runtimes are several orders of magnitudes better The key operation in obtaining these results is the ability to advance time symbolically We show how to do this e ciently by essentially quantifying out a special variable z which is used to represent the constant zero The symbolic manipulations given in this paper are su cient to verify TCTL formulae fully symbolically*

*Introduction*

*Model checking is today used extensively for formal veri cation of nite* state systems such as digital circuits and embedded software The success of the technique is primarily due to the use of a symbolic representation of sets of states and relations between states as predicates over Boolean variables

*using for instance binary decision diagrams BDDs By representing*

*This work was carried out while the authors were at the Department of Information Technology Technical University of Denmark and was nancially supported by a grant from the Danish Technical Research Council*

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

*Q f s V g*

*R s V*

*R s for all s s while Q do*

*Remove some s V from Q*

*k*

*Q*

*R Q*

*while is satisfiable Q do Q Next Q*

*f s V s*

*k*

*V g Next s V*

*Q Q R*

*for i to k do*

*if Vi R si then Add si Vi to Q R si R si Vi*

*a*

*R R Q*

*b*

*Fig Two di erent approaches for constructing the set of reachable states R a Outline of the algorithm used in current tools such as Kronos and Uppaal and*

*b a fully symbolic algorithm*

*for example the set of reachable states as a predicate instead of explicitly* enumerating the elements of the set it is possible to verify systems with a very large number of states However these symbolic methods do not easily generalize to models that contain continuous variables ranging over non countable domains like for example real time systems where time is modeled using continuous variables and the behavior of a system is speci ed using constraints on these variables One problem is how to succinctly represent the usually in nite number of states of such systems another problem is how to perform the basic veri cation operations resetting clocks advancing the time of clocks etc symbolically on this representation in order to compute the reachable state space or to verify a temporal property of the system

*Current Approaches*

*A state in a timed system is a pair s v where s is a discrete state e g markings of Petri nets or locations in timed automata and v is the associated* timing information i e a value assignment to the clocks in the system To analyze timed systems which have an in nite number of states due to the dense nature of the clocks clock assignments are grouped into sets This allows the state space of a timed system to be represented as a nite set of pairs s V of discrete states and their associated group of clock valuations The reachable states space R for a timed system can be determined by the generic algorithm in Fig a here we view R as a mapping from discrete states s to their associated group of clock valuations V The function Next

*res all possible transitions and advances time from the set of states s V*

*Current state of the art techniques for verifying timed systems e g*

*are based on representing each set of clock assignments using a set* of di erence bound matrices DBMs Each di erence bound matrix can

*represent a convex set of clock assignments thus to represent V in general a number of matrices is needed i e representing V as a union of convex* sets The function Next constructs the set of new states such that each *Vi*

*is a single DBM The test in the line marked is performed by checking*

*whether the DBM Vi is not contained in any of the DBMs used to represent* R si

*Although DBMs provide a compact representation of a convex set of clock* con gurations there are several serious problems with the approaches based on DBMs the number of DBMs for representing the timing information

*V can become very large there is no sharing or reuse of DBMs among the* di erent discrete states and each discrete state is represented explicitly thus these approaches are limited by the number of reachable states of the system the well known state explosion problem

*A Symbolic Approach*

*The rst two problems can be addressed by representing the set V as a propo sitional formula over inequalities of the form x y d x and y are clock* variables and d is a constant If we have a compact representation of such formulae and can decide valid implications for performing the check in the line marked with we can use the algorithm in Fig a immediately Dif ference decision diagrams are a candidate for such a data structure which furthermore allows reuse of sub formula among the discrete states Initial ex periments with this approach implemented in Uppaal show a signi cant improvement in memory consumption even though the discrete states still are enumerated explicitly

*In this paper we address all three problems by constructing the set of* reachable states R in a fully symbolic manner i e without enumerating the discrete states and without representing the timing information as a set of DBMs In our approach both the discrete part of a state and the associated timing information are represented by a formula That is sets of states s V are represented by a single formula similar to how sets of discrete states are represented bya formula when performing traditional symbolic model checking of untimed systems Using such a representation the set of reachable states R can be computed using the standard xed point iteration shown in Fig b A core operation when performing symbolic model checking is to determine

*a formula representing the set of states reachable by ring any transition*

*or advancing time from a state satisfying i e the function Next in* Fig b Firing the transitions is straightforward but advancing time is more involved We introduce a variable z denoting zero or current time and express all constraints of the form x d as x z d The use of a designated variable representing zero for eliminating absolute constraints is used both in DBMs and also when solving systems of di erence constraints A key contribution of this paper is that we show how the z variable in addition

*to making the representation more uniform also makes it possible to advance* time in a set of states represented by a formula essentially by performing an existential quanti cation of z Let Pnext denote a predicate stating whether it is legal to advance time by changing the reference point from z to z Thus P will require that z z since advancing time by some amount corresponds to decreasing the reference point z by Typically Pnext will also include constraints expressing state invariants and urgency predicates Now a formula representing the set of states reachable from by advancing time by is determined from

*next*

*z P z z z z*

*next*

*More generally the set of states reachable from by advancing time by an arbitrary amount is determined from*

*z P z z*

*next*

*Another key contribution of this paper is that we show that performing* fully symbolic model checking of timed systems amounts to representing and deciding validity of formulae in a simple rst order propositional logic over inequalities of the form x y d

*x y d j j j x*

*where x and y are real valued variables and d R is a constant A practical* model checking algorithm therefore requires a compact representation of for mulae of the form and an e cient decision procedure to determine validity of such formulae

*In Section we introduce a simple model of timed systems called timed* guarded commands and sketch how it can represent timed automata Sec tion shows how to symbolically compute the set of reachable states of such timed systems and sketch how to perform a fully symbolic model checking of TCTL formulae Section introduces a data structure called di erence deci sion diagrams for representing and deciding validity of formulae of the form

*In Section we demonstrate the e ciency of the symbolic approach by*

*computing the set of reachable states for a non trivial timed system and com pare the results with the state of the art tools Kronos and Uppaal Finally Section summarizes the contributions*

*Related Work*

*Model checking of timed systems timed automata in particular see for a* survey has been extensively studied and a number of tools exist for verifying such systems One approach is based on making the dense domains discrete by assuming that timers only can take integer or rational values Such a discretization makes it possible to use BDDs for representing both the discrete

*states and the associated timing information However this way* of representing dense domains is often ine cient the BDD representation is very sensitive to the granularity of the discretization and to the size of the delay ranges

*The unit cube approach models time as dense but represents the timing* information using a nite number of equivalence classes Again the number of timed states is dependent on the size of the delay ranges and easily becomes unmanageable As mentioned above more recent timing analysis methods use di erence bound matrices DBMs for representing the timing informa tion One can see the use of DBMs as expanding formulae of the form into disjunctive normal form and representing each conjunction of di erence constraints using a di erence bound matrix Several attempts have been made to remedy the shortcoming of DBMs discussed above for example by using partial order methods or by using approximate methods Although these approaches do address the problem that the number of DBMs for representing the timing information can become very large they still enumerate all discrete states

*Henzinger et al describe how to perform symbolic model checking* of timed systems Although apparently similar to our approach there are a number of signi cant di erences First we show that the simple rst order logic with only one type of clock constraints x y d is su cient for representing the set of states of a timed system This allows us to represent sets of states e ciently using an implicit representation of formulae e g di erence decision diagrams Secondly we show how to perform all operations needed in symbolic model checking within this logic A core operation is advancing time which we show can be performed within the logic by introducing a designated variable z and using existential quanti cation

*Based on the initial ideas of di erence decision diagrams DDDs Behr mann et al have implemented a minor variation of DDDs allowing a fanout* of more than two which they call CDDs They have shown a signi cant im provement in memory consumptions in Uppaal even though the experiments in contrast to ours do not use a fully symbolic approach the discrete states are enumerated explicitly Thus this approach will not be able to handle the larger instances of the timed system described in Section

# *Modeling Timed Systems*

*Timed guarded commands are a simple notation for modeling systems* with time The notation is su cient for encoding popular notations for sys tems with time such as timed automata and timed Petri nets

*Timed Guarded Commands*

*A timed guarded command program G is a tuple B C T I where B is a set of Boolean variables C is a set of continuous variables called clocks T is a set of timed guarded commands and I is a state invariant A timed guarded command t T has the form g v d where g is a guard and v d is a multi assignment of n constant values d B R n to Boolean variables and clocks v B C n Guards and state invariants are expressions*

*constructed from the following grammar*

*F j T j x d j x y d j b j j j b j x*

*where x y C are clocks b B is a Boolean variable d R is a con stant and is a relational operator from f g The symbols* F and T denote false and true respectively and the symbols negation

*conjunction and existential quanti cation have their usual meaning*

*Example An example of a program is G fbg fx yg T I where T contains the two guarded commands*

*b x b F*

*b x b y F*

*and the state invariant is I b x b x*

*Transitional Semantics of Timed Guarded Commands*

*A state of the program G B C T I is an interpretation i e a value* assignment of the Boolean variables and the clocks For each Boolean variable b B s b B denotes the interpretation of b in the state s and for each clock x C s x R denotes the interpretation of x in the state s We use the notation s x y d to denote the state s equivalent to s except that s x s y d A state and sets of states can be represented by an expression of the form The state s satis es an expression written s j if evaluates to true in the state s and we write for the set of states that satisfy

*The semantics of a timed guarded command program G B C T I is* a transition system S where S is the set of states of the program and

*is the transition relation In each state the program can either execute a* command t T if its guard is true a discrete transition or let time pass time units a timed transition Executing a command changes the value of some or all of the variables according to the multi assignment and letting time pass uniformly increases the values of all clocks by We use the notation

*t*

*s s*

*for a discrete transition from the state s to s*

*obtained by executing*

*the command t and the notation s*

*s for a timed transition obtained by*

*t*

*increasing all clocks by The discrete transition for a timed command*

*t T of form g v d is de ned by the following rule*

*s j g s v d j I t*

*s s v d*

*The timed transition for advancing all clocks by is de ned by the following*

*rule*

*s c c j I*

*s s c c*

*where R c denotes a vector of all clocks in C and c denotes the* vector where is added to the clocks in c

*Example Consider the timed guarded command program G from Exam ple and let s be a state satisfying b x There are in nitely many* timed transitions from s in the transition system for G but none of these timed transitions leads to a state where x because the state invariant

*b x must hold continuously*

*Given a transition system S for a timed guarded command program* G B C T I and a set of states S S Next discrete S denotes the set of states reachable from S by executing any timed guarded command in T

*Next*

*discrete*

*t*

*S fs s S s s g*

*t T*

*Similarly the set of states reachable from S by advancing time by an arbitrary* amount is given by

*Next*

*timed*

*S fs s S s s g*

*R*

*The set of states reachable from S denoted Reachable S is de ned as the* least xed point of the function F X S Next X where

*Next X Next discrete X Next timed X*

*Encoding Timed Automata*

*Timed guarded command programs can be used to model popular notations* for timed systems such as timed automata A timed automaton over a set of clocks consists of a set of locations a set of events and a set of timed transitions Each location is associated with a location invariant over the clocks and each timed transition from location l to location l is labeled with an event a and has a guard g over the clocks Furthermore each of the timed

*This is an example of a program with a time blocked state*

*a x*

*x*



*l*

*a x*

*l*

*fyg*

*x*

*Fig The timed automaton in Example*

*transitions has a set of clocks fcg to be reset when the timed transition is*

*red*

*l*

*a g*

*fc g*

*l*

*A timed automaton can be encoded as a timed guarded command program Each location is encoded as a Boolean variable In a shared variable model* as ours the presence of an event from an alphabet can be modeled by a global event variable e taking on any of the values in This variable can for instance be encoded using a logarithmic number of Boolean variables Each timed transition in the automaton corresponds to a timed guarded command

*l e g l l c F T*

*a*

*The guard of the command is the guard of the timed transition g conjoined* with the source location l of the timed transition and a condition ea requiring the event variable e to have the value a The multi assignment assigns F to the source location l and T to the destination location l of the timed transition and resets the relevant clocks

*Example Fig shows an automaton over the clocks fx yg with two* locations and two timed transitions Encoding this automaton as a timed guarded command program yields the program G from Example when ignoring the event a and encoding the two locations l and l logarithmically using a Boolean variable b

# *Analyzing Timed Guarded Commands*

*To verify properties of a timed guarded command program G B C T I we symbolically analyze the corresponding transition system S That is given a set of states represented by a formula we determine a formula that* represents the set of states reachable by executing timed guarded commands according to the inference rule or by advancing time according to the inference rule As we will show this formula is obtained by manipulations entirely within the logic

*This is sometimes referred to as a one hot encoding of the locations In practice a logarithmic encoding may be more e cient*

*Di erence Constraint Expressions*

*Any expression generated by the grammar can be represented by a dif ference constraint expression z of the form The expression z is obtained* by introducing a new variable z denoting zero and performing the following three steps First encode each Boolean variable bi B in as a di erence

*constraint x x where x x C are clocks only used in the encoding*

*i i i* *i*

*of b Second replace each occurrence of a constraint of the form x d*

*i*

*in with the di erence constraint x z d Third express each di erence* constraint of the form x y d in terms of the relational operator

*We de ne two useful operators on di erence constraint expressions re placement and assignment Replacement syntactically substitutes all occur rences of a variable x by a variable y plus a constant d in an expression denoted by y d x If x and y are di erent variables the replacement*

*y d x can be expressed in the grammar as x x y d Oth*

*erwise x d x is de ned as t x x d t where t is a variable di erent* from x and not occurring in Assignment gives a variable x the value of a variable y plus a constant d denoted by x y d If x and y are di erent variables the assignment x y d is expressed in the grammar as

*x x y d Otherwise the assignment x x d is de ned as*

*x d x which might seem counter intuitive Assignment and replacement* of Boolean variables are de ned in the standard way

*To formally expresses the symbolic manipulations we introduce some use ful shorthands We use z as a shorthand for z z that is z* is the set of states that satisfy when z is equal to It is easy to prove that

*z z for any Eliminating the constraints of the form x d from* the grammar in makes it possible to add to all clocks simultaneously by decreasing the common reference point z by

*c c z z z z*

*Furthermore as will be shown in the following the set of states reachable by* advancing time by any value can be computed by an existential quanti cation of z

*Reachability Analysis*

*Given an expression of the form representing a set of states z S we* now show how to determine an expression representing the set of states reach able from z The set of states reachable by ring the timed guarded com mand t from any state in z is determined by the function Nextdiscrete t The function restricts to the subset where the guard g holds performs the

*It turns out that when using di erence decision diagrams see Section with this ap parently strange encoding of Boolean variables the Boolean manipulations can be done as e ciently as when using BDDs*

*assignment of the constants d to the variables v and restricts the resulting* set to the subset where the state invariant I holds

*Nextdiscrete g v d gz v d Iz*

*where the assignment v d is a shorthand for ci z di for each of* the clocks ci in v and bi di for each of Boolean variables bi in v The set of states that can be reached from the set z by ring any timed guarded command in T is given by

*Nextdiscrete Nextdiscrete t*

*t T*

*The z variable plays a central r le when determining the set of states that* can be reached from z by ring a timed transition We advance time by changing the reference point from z to z with z z since decreasing the reference point by corresponds to increasing the values of all clocks by Often the system will restrict the valid choices for z by requiring that the state invariant holds in z and at all intermediate points in time This is expressed by the predicate

*Pnext z*

*z Iz z*

*z z*

*z Iz*

*If the state invariant Iz only expresses upper bounds on the clocks the uni versal quanti cation is implied by Iz and can be omitted*

*Now to advance time by in all states z we simply decrease the* reference point z by z z which can also be written as z z

*z z z The set of states reachable from z that also satisfy P is*

*next*

*given by z z z P z z Thus the set of states reachable*

*next*

*from z by advancing time by an arbitrary amount is given by*

*Next z z z P z z*

*timed next*

*R*

*z P z z*

*next*

*That is we advance time in a set of states by performing a single existential* quanti cation The correctness of the next state functions is proved in

*Example If the state invariant is x the predicate Pnext is given by*

*P z z x z*

*next*

*z z z z x z*

*z z x z x z*

*Consider the set of states satisfying x x The set of* states obtained by advancing time from is thus given by Nexttimed z z where

*Nexttimed z z z Pnext z z x z x z*

*A timed guarded command t T is called urgent if it is required to re* instantaneously whenever the guard becomes true Modifying Pnext to handle urgent commands is straightforward Given a set T T of urgent timed guarded commands we let U denote the predicate

*U g g v d T*

*Consider a state s We can only re a timed transition s s if there*

*z*

*are no urgent transitions enabled in s Thus an additional requirement is*

*added to Pnext ensuring that no urgent transitions are enabled when advancing* time except in the endpoint i e the revised Pnext becomes

*Pnext z*

*z Iz z*

*z z*

*z Iz Uz*

*If the urgency predicate does not refer to z Pnext is simpli ed to*

*Pnext z*

*z Iz z*

*z z*

*z Iz*

*Uz*

*The functions de ned in and form the basis for constructing the set* of reachable states symbolically Let Next be a function which determines the set of states which can be reached by ring either a discrete or a timed transition from a state in z

*Next Nextdiscrete Nexttimed*

*The set of states reachable from z denoted Reachable is the least*

*xed point of the function F X Next X which can be determined* using a standard xed point iteration Detecting that a xed point has been reached is done by checking that two successive approximations i and i are semantically equivalent i e that i i is a tautology It is well known that there exists contrived timed systems where the computation of the xed point does not terminate for example if the di erence between two clocks increase ad in nitum As in the traditional analysis of timed automata it is possible to determine subclasses of timed guarded commands for which termination is ensured

*Example Consider again the program from Example The set of* states reachable from b x y is Reachable z z where

*Reachable z b x y x z*

*b x y x z x y x z*

*Symbolic Model Checking*

*To perform symbolic model checking for example of a TCTL formula the set* of states that can reach a given set z needs to be determined The set of

*states that can reach z by ring any timed guarded command g v d* in T is given by

*Prevdiscrete*

*g v d T*

*v v d gz Iz*

*where the expression v d is a shorthand for ci z di for each of the clocks* ci in v and bi di for each of Boolean variables bi in v The set of states that can reach z by advancing time is determined analogously to the forward case

*where*

*timed prev*

*Prev z P z z*

*Pprev z z Iz z*

*z z*

*z Iz*

*The set of states that can reach a state in z by ring either a discrete or a* timed transitions is

*Prev Prevdiscrete Prevtimed*

*Thus we can construct the set of states that can reach a state satisfying as* the least xed point of the function B X Prev X Moreover Prev can be used to perform symbolic model checking of TCTL TCTL is a timed version of CTL obtained by extending the logic with an auxiliary set of clocks called speci cation clocks These clocks do not appear in the model and are used to express timing bounds on the temporal operators The atomic predicates of TCTL are di erence constraints over the clocks from the model and the speci cation clocks Semantically the speci cation clocks become part of the state they proceed synchronously with the other clocks but are not changed by the model A speci cation clock u can be bound and reset by a reset quanti er u

*Symbolically we can nd the set of states satisfying a given TCTL formula*

*by a backward computation using a xed point iteration for the temporal* operators For instance the set of states satisfying the formula EU is computed symbolically as the least xed point of the function B X

*Prev X The set of states satisfying u is computed symbolically*

*as u u z i e the reset quanti er corresponds to restricting* the value of u to zero and then remove it by existential quanti cation The atomic predicates and the Boolean connectives correspond precisely to the corresponding di erence constraint expressions

*Above we have determined the set of states using a constrained image ap proach To compose systems synchronously as used for instance in timed au tomata a timed guarded command program can be encoded using a transition* relation R over present state variables V B C fzg and the next state

*variables V fv v V g as traditionally done in symbolic model checking* of discrete systems but including the reference points z and z The rela tion R is constructed by combining the transitions of each automaton using disjunctions and then combining the automata using conjunctions Thus the parallel composition of a set of timed automata can be analyzed fully symboli cally i e both symbolically with respect to the parallel composition and with respect to the representation of sets of clock valuations and discrete states

*see Using a transition relation we get the bene t that well known and*

*very useful tricks from the work on BDDs such as early variable quanti ca tion and partitioned representation of the transition relation are immediately* applicable

# *Di erence Decision Diagrams*

*The previous sections show that to perform symbolic analysis of timed systems* we need a data structure for representing di erence constraint expressions and a decision procedure to determine validity of such expressions Di erence decision diagrams DDDs are a candidate for such a data structure Similar to how a BDD represents the meaning of a Boolean formula implicitly a DDD represents the meaning of a di erence constraint expression of the form using a decision diagram in which the vertices contain di erence constraints

*A DDD is a directed acyclic graph V E with two terminals and and* a set of non terminal vertices Each non terminal vertex corresponds to the if then else operator de ned as where the test expression is a di erence constraint and the high branch and low branch

*are other DDD vertices Each vertex v in a DDD denotes a di erence*

*constraint expression v given by*

*v v high v low v*

*where v is the di erence constraint of v and high v and low v are the* high and low branches respectively

*As an example of a DDD consider the following expression over x y z R*

*x z y z y x Fig shows z as an x y plot and the corresponding DDD*

*As shown in DDDs can be ordered and reduced making it possible*

*to check for validity and satis ability in constant time Furthermore the op erations for constructing and manipulating DDDs according to the syntactic* constructions of are easily de ned recursively on the DDD data structure thus making it simple to specify and implement algorithms for these opera tions The function Apply op u v is used to combine two ordered locally

*y*

*x z*

*x z*

*y x*

*a*

*x*

*b*

*y z*

*Fig The expression in as a an x y plot for z and b a di erence decision diagram*

*reduced DDDs rooted at u and v with a Boolean operator op e g the negation* and conjunction operations in Apply is a generalization of the version used for ROBDDs and has running time O jujjvj where j j denotes the number of vertices in a DDD The function Exists x u is used to quantify out the variable x in a DDD rooted at u The algorithm is an adoption of the Fourier Motzkin method removing all vertices reachable from u con taining x but keeping all implicit constraints induced by x among the other variables e g x z x x y is equivalent to z y Exists computes the modi ed and additional constraints in polynomial time but has an exponential worst case running time since the resulting DDD must be ordered

*Recall that Boolean variables in are encoded as xi x This*

*i*

*encoding allows us to represent and manipulate both real valued and Boolean* variables in a homogeneous manner Furthermore the encoding has the ad vantage that any Boolean expression will have a canonical DDD representation

*because of the DDD reduction rules and can be manipulated as e ciently*

*as when represented by a BDD*

*Experimental Results*

*We demonstrate the applicability and e cacy of the symbolic approach by* analyzing two di erent versions of Milner s scheduler with time We compare the runtimes of the symbolic approach using DDDs with those obtained with two state of the art tools Kronos and Uppaal The two ver sions of Milner s scheduler are simple regular and highly concurrent systems and they illustrate the advantages of our symbolic approach based on DDDs over state of the art tools

*Milner s Scheduler with One Clock*

*Milner s scheduler consists of N cyclers connected in a ring cooperating* on controlling N tasks We associate three Boolean variables ci hi and ti with each cycler and use a global clock H to ensure that a cycler passes the token on to the following cycler within a bounded amount of time The ith cycler is described by two guarded commands and the task is modeled by a



*D*

*i mod N*

*H H*

*l*

*H H*

*u*

*Ci*

*H*

*H H*

*u*

*D i mod N*

*H H*

*l*



*Ci*

*Di*

*a b*

*Fig The ith cycler of Milner s scheduler with one clock The cycler is modeled with two timed automata synchronizing via CCS like channels as in Uppaal Initial locations are drawn with double circles for the rst cycler the initial location in b is the opposite location The locations of the automaton in a represent the four possible combinations of the variables hi and ti and the automaton in b represents the variable ci*

*third guarded command*

*ci ti H ti ci hi T F T*

*h Hl H c*

*i*

*i mod N*

*hi T F*

*ti ti F The state invariant is given by*

*N*

*I h H Hu i*

*i*

*expressing that cycler i must pass on the token no later than time Hu Thus the amount of time a cycler can keep the token is determined by the interval*

*Hl Hu Furthermore the rst guarded command is urgent thus the urgency* predicate is

*N*

*U ci ti*

*i*

*It is straightforward to model Milner s scheduler with timed automata Fig shows the ith cycler in the notation used by Uppaal*

*We have computed the reachable state space for increasing N using a xed point iteration with front sets The results are shown in Table a together* with the runtimes obtained with Kronos version b and Uppaal ver sion This version of Milner s scheduler has a number of discrete states which is exponential in N since a task can terminate independently of the other tasks Thus state space exploration based on enumerating all discrete states as in Uppaal and Kronos only succeeds for small systems In the symbolic approach using DDDs discrete states are represented implicitly as when using BDDs for purely discrete systems and choosing a good ordering of the variables gives polynomial runtimes and state space representations

*Table*

*Experimental results for Milner s scheduler with a one clock using the bounds*

*Hl Hu and b one clock per task using the bounds*

*Hl Hu and T l T u The rst column shows the number of cyclers and the following three columns show the CPU time in seconds to build the reachable state space using Kronos Uppaal and DDDs respectively The results were obtained on a Pentium II PC with MB of memory running Linux A denotes that the analysis did not complete within an hour*

|  |  |  |  |
| --- | --- | --- | --- |
| *N* | *Kronos* | *Uppaal* | *DDD* |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| *N* | *Kronos* | *Uppaal* | *DDD* |
|  |  |  |  |
|  |  |  |  |

*b*

*a*

*Milner s Scheduler with a Clock per Task*

*We now restrict the time a task can be executing by introducing a clock Ti that* measures the execution time of each task ti The task ti must terminate within a certain bound T l T u after it is started The resulting system potentially has N concurrently running clocks one for each task plus one for the token but the system will have fewer discrete states than the previous version since the bounded execution time of the tasks limits the number of reachable discrete states The ith cycler is now described by the guarded commands

*ci ti H Ti ti ci hi T F T*

*hi i mod N*

*H H c*

*l*

*hi T F*

*t T l T t F*

*i i* *i*

*Introducing the new clocks changes the state invariant to*

*N*

*I hi H Hu ti Ti T u i*

*Fig shows the ith cycler in the notation used by Uppaal*

*The runtimes for computing the reachable state space for increasing N are*



*D*

*i mod N*

*H H*

*l*

*T*

*l*

*i*

*T*

*H H*

*u*

*Ci*

*H*

*Ti*

*Ti T*

*u*

*Ti T*

*l*

*H H*

*Ti T*

*u*

*u D i mod N*

*H H*

*l*



*Ci*

*Di*

*a b*

*Fig The timed automata modeling the ith cycler of Milner s scheduler with a clock for each task Initial locations are drawn with double circles for the rst cycler the initial location in b is the opposite location*

*shown in Table b Again the runtimes of Kronos and Uppaal are expo nential in N while the symbolic approach using DDDs results in polynomial* runtimes In this version of Milner s scheduler the problem for Kronos and Uppaal is the large number of clock variables This is handled in the symbolic approach using DDDs by eliminating unused clocks from the representation

*i e we quantify out Ti whenever the guarded command that sets ti to false*

*is red in Nextdiscrete*

*As for BDDs the size of a DDD depends on the chosen variable ordering In the two versions of Milner s scheduler experiments show that the Boolean* variables should precede the clocks in the decision diagram The Boolean variables are ordered as t c h tN cN hN Pairs of clocks

*x x are ordered reversed lexicographically using the ordering z H*

*i j*

*T TN There are a number of techniques to avoid BDD size blow up*

*that also apply to DDDs For example instead of building a DDD for I we* build a list of N implicitly conjoined DDDs as described in When we build the DDD for the set of discrete next states de ned in we conjoin each element hi H Hu ti Ti T u of the list to the expression

*gz v d We use the same technique when we build the DDD for the set*

*of timed next states de ned in which is possible because I only expresses* upper bounds on the clocks and thus P is given by z z I U

*next z z*

*Conclusion*

*We have shown how di erence constraint expressions can be used to fully* symbolically represent and verify concurrent timed systems A key idea is to avoid representing absolute constraints Instead these constraints are ex pressed relative to a special variable z which allows us to advance all clocks synchronously by performing a single existential quanti cation

*Our results show that an e cient implementation of di erence constraint* expressions is highly desirable and we propose an implementation using dif ference decision diagrams Di erence decision diagrams DDDs possess the same abilities as BDDs of providing a compact representation while admit

*That is x x iff x x*

*or x*

*x x*

*i j xi xj*

*j j j*

*j i* *xi*

*ting an e cient validity check Just as BDDs provide an implementation of* quanti ed Boolean logic which allows the symbolic veri cation of discrete sys tems DDDs provide an e cient implementation of di erence constraint expressions which allows the symbolic veri cation of timed systems

*Continuing extending the power of the underlying Boolean logic di erence* constraints could be replaced by the more powerful linear inequalities yield ing Presburger formulae An e cient representation of Presburger formulae would therefore along the lines of this paper immediately provide a symbolic veri cation of a guarded command language with Boolean combinations of linear inequalities as guards and linear expressions in assignments including the extensions to automata and concurrent compositions

*References*

*M Abadi and L Lamport An old fashioned recipe for real time In REX Workshop Real Time Theory in Practice volume of LNCS pages Springer Verlag*

*R Alur and D Dill The theory of timed automata In Real Time Theory in Practice volume of LNCS pages Springer Verlag*

*E Asarin M Bozga A Kerbrat O Maler A Pnueli and A Rasse Data structures for the veri cation of timed automata In Int Workshop on Hybrid and Real Time System March*

*F Balarin Approximate reachability analysis of timed automata In Proc Real Time Systems Symposium pages IEEE*

*G Behrmann K G Larsen J Pearson C Weise and Wang Yi E cient timed reachability analysis using clock di erence diagrams Technical Report DoCS Technical Report Uppsala University*

*W Belluomini and C J Myers E cient timing analysis algorithms for timed state space exploration In Proc Int Symposium on Advanced Research in Asynchronous Circuits and Systems April*

*W Belluomini and C J Myers Veri cation of timed systems using POSETs In Proc Computer Aided Ver cation CAV June*

*Johan Bengtsson Kim G Larsen Fredrik Larsson Paul Pettersson and Wang Yi Uppaal in In Proc International Workshop on Tools and Algorithms for the Construction and Analysis of Systems TACAS volume of LNCS pages Springer Verlag March*

*B Berthomieu and M Diaz Modeling and veri cation of time dependent systems using time Petri nets IEEE Transactions on Software Engineering*

*M Bozga O Maler A Pnueli and S Yovine Some progress in the symbolic veri cation of timed automata In Proc Ninth International Conference on Computer Aided Veri cation CAV volume of LNCS pages*

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

*J R Burch Trace Algebra for Automatic Veri cation of Real Time Concurrent Systems PhD thesis Carnegie Mellon August*

*J R Burch E M Clarke K L McMillan D L Dill and L J Hwang Symbolic model checking states and beyond In Proceedings Fifth Annual IEEE Symposium on Logic in Computer Science pages IEEE Computer Society Press*

*S V Campos E Clarke W Marrero M Minea and H Hiraishi Computing quantitative characteristics of nite state real time systems In Real Time Systems Symposium pages IEEE December*

*E M Clarke and E A Emerson Design and synthesis of synchronization skeletons using branching time temporal logic In Workshop on Logics of Programs volume of LNCS pages Springer Verlag*

*T H Cormen C E Leiserson and R L Rivest Introduction to Algorithms MIT Press*

*C Daws A Olivero S Tripakis and S Yovine The tool Kronos In Proc of Workshop on Veri cation and Control of Hybrid Systems volume of LNCS pages Springer Verlag October*

*C Daws A Olivero and S Yovine Verifying ET LOTOS programs with Kronos In Proc th IFIP WG G International Conference of Formal Description Techniques FORTE pages Chapman Hall October*

*C Daws and S Yovine Two examples of veri cation of multirate timed automata with Kronos In Proc th IEEE Real Time Systems Symposium*

*RTSS pages IEEE Computer Society Press December*

*D L Dill Timing assumptions and veri cation of nite state concurrent systems In Automatic Veri cation Methods for Finite State Systems volume*

*of LNCS Springer*

*J B J Fourier Second extrait In G Darboux editor Oeuvres pages Paris Gauthiers Villars*

*T A Henzinger Z Nicollin J Sifakis and S Yovine Symbolic model checking for real time systems Information and Computation*

*Alan J Hu and David L Dill E cient veri cation with BDDs using implicitly conjoined invariants In Proc Fifth International Conference on Computer Aided Ver cation CAV volume of LNCS pages*

*K G Larsen P Pettersson and W Yi Model checking for real time systems In Proc of the th Int Conference on Fundamentals of Computation Theory volume of LNCS pages August*

*Kim G Larsen Paul Pettersson and Wang Yi Uppaal in a nutshell Springer International Journal of Software Tools for Technology Transfer*

*Robin Milner Communication and Concurrency Prentice Hall*

*J M ller J Lichtenberg H R Andersen and H Hulgaard Difference decision diagrams Technical Report IT TR Department of Information Technology Technical University of Denmark February A revised version will appear in proc of Computer Science Logic CSL Madrid September*

*J M ller J Lichtenberg H R Andersen and H Hulgaard On the symbolic veri cation of timed systems Technical Report IT TR Department of Information Technology Technical University of Denmark February*

*T G Rokicki Representing and Modeling Digital Circuits PhD thesis Stanford University*

*T G Rokicki and C J Myers Automatic veri cation of timed circuits In D L Dill editor Proc Sixth International Conference on Computer Aided Veri cation CAV volume of LNCS pages*

*E Verlind G de Jong and B Lin E cient partial enumeration for timing analysis of asynchronous systems In Proc ACM IEEE Design Automation Conference*

*H Wong Toi and D L Dill Approximations for verifying timing properties In Theories and Experiences for Real Time Systems Development World Scienti c Publishing*

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

*S Yovine Model checking timed automata In Embedded Systems volume of LNCS Springer Verlag*