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

*Model Generation of a Fictitious Clock* Real Time Logic Using Sharing Trees

# *Laurent Ferier a Jean Fran cois Raskin b c* Pierre Yves Schobbens a

*a Institut d Informatique F U N D P Namur Belgium*

*b Max Planck Institut fu r Informatik Saarbru cken Germany*

*c EECS University of California at Berkeley USA*

*Abstract*

*We present rst the logic MTL a real time temporal logic that is at the heart of the real time speci cation language Albert Since this logic is undecidable we approximate it using the theory of Abstract Interpretation by its ctitious clock counterpart MTLFC*

*We then present a symbolic tableau based model generation decision procedure in ExpSpace which is theoretically optimal In practice however we see that the introduction of integer valued prophecy variables will make it more e cient From these variables we reconstruct by reverting the process a logic that we call ECLFC which can be decided in PSpace and has the same expressivity as MTLFC*

*Theory thus shows that memory space is the critical factor However the classical compaction of memory space by BDDs is not ideal here since our integer variables would need to be encoded by booleans Therefore we use Sharing Trees instead a compact data structure that can accommodate arbitrary data types Preliminary results on this implementation are reported*

*Introduction*

*Computer systems are now pervasive i e used in all domains of human ac tivity In most cases these systems e g nuclear plant controllers airplane piloting systems train security systems medical monitoring computers are*

*critical i e errors in such systems are not acceptable due to the potentially dramatical consequences in terms of human lives human health or nancial losses*

*distributed i e implemented by several independent computers This im plies that the traditional testing method is hopeless due to the combina torial complexity of concurrent executions*

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

*reactive i e the evolution of these systems over time is more important* than their nal result making the classical alternative to testing namely the proof by pre and post conditions inapplicable

*real time i e the timely delivery of results is as important as the results* themselves This renders the recently developed techniques of temporal logic incomplete and has thus triggered foundational research to add real time to temporal formalisms

*We see thus a large gap between the strong social demand for safe reliable* computer systems and the weak methods now used to build them We are thus investigating methods to develop real time systems at the foundational

*methodological and practical level*

*In this paper we relate the foundational level of deciding real time logics* with the practical aspects of their implementation We do not talk about symbolic model checking SMC which is only applicable when the system has been implemented to the point where a model can be extracted from the implementation but about symbolic model generation SMG a germane technique that allows to work on systems described by logical formulae thus already applicable in the early stages of real time software development such as requirements speci cation and architectural speci cation This technique is well known for linear temporal logic given a logical formula it builds an automaton that accepts exactly the models of this formula The relations be tween formulae entailment equivalence etc are then decided by operations on automata A special kind of automata is needed for each type of logic

*DT*

*for instance discovered the automata corresponding to MITL How*

*ever using non classical automata has its costs in terms of implementation since baroque data structures are often needed Thus some researchers have proposed to model real time by a state counter a fairly counter intuitive* model that has the advantage of an immediate implementation in known tem poral techniques Here we propose a median way We use dense time at the speci cation level due to its naturalness and semantical advantages including compositionality ease of temporal re nement etc

*The resulting logic is undecidable but for all practical purposes can be* approximated by its ctitious clock counterpart presented in Section The

*ctitious clock model approximates real time intuitively by considering that* a perfect discrete clock is started together with the system All times are read on the integer clock display which is a truncation of the real time The clock display is incremented at clock ticks Therefore the time of events is only known up to one tick delay This imprecision makes the logic discrete and decidable by classical techniques adapted in Section while the error can be made as small as wanted by increasing the tick rate

*Looking into the technique we see that the introduction of prophecy vari ables Subsection improves the e ciency of the procedure We can design* a logic called ECLFC from these variables Subsection and as expected this logic is more e cient but surprisingly perhaps keeps the same expressive

*power in this discrete model In we show that in contrast expressive* power is lost in the pointwise model

*In Section we consider how to implement the decision procedure Since* it is complete for memory space we must use memory compression techniques The classical technique is ordered binary decision diagrams OBDDs but it has the drawback of accommodating only booleans It is still usable at the price of an encoding of integer prophecy variables by booleans However the related technique of Sharing Trees STs allows to accommodate di rectly integers and has otherwise similar performances for the needs of model generation When dealing with state by state exploration it is even much more e cient The logical operations translate simply on this data structure We conclude by reporting preliminary experiments on our imple mentation and plans for further experiments

## *Real time speci cation*

*Our approach to real time distributed reactive systems is currently centered* around the speci cation language Albert Agent oriented Language for Build ing and Eliciting Requirements for Real Time It is built on basis of the logic MTL extended with rst order structuring actions with duration and communication concepts and with a number of patterns that allow a direct representation of common sentences without the need to delve into the joys of nested untils The de nition of Albert is too large to be presented here

*MTL is an extension of linear temporal logic where the temporal operator* is extended by a constraint on the distance at which the event will occur

*MTL can be given a number of semantics just gives basic requirements* on the domain of time use the real numbers as time domain and prove that the logic is undecidable in this case Nevertheless we used this latter logic as the basis of Albert since the extension to rst order renders the logic undecidable anyway and is needed for most practical applications We consider thus decidability as a problem for tools tools that require some form of decidability will select an adequate fragment of the language and perform an abstract interpretation to get rid in a meaningful way of the speci cation parts that they cannot deal with Even when the speci cation logic is theoretically decidable this approach has to be followed to obtain results in a reasonable amount of time Conversely some tools do not require decidability and work satisfactorily in practice This is thus also the approach of this paper we present a logic MTLFC that approximates MTL The theory is established in MTL is itself an approximation of Albert by using known techniques to simplify the rst order part of Albert In subsection

*we approximate MTLFC by ECLFC for e ciency*

*Fictitious clock*

*Given a real time temporal logic where time is represented by real numbers we can always de ne its ctitious clock FC counterpart In the semantics wherever a time was used we replace it by a natural number that is the* truncated value of the real time for which we use the metaphor of the clock display In the syntax we add a special proposition tick that indicates when the FC will tick Ticks are exactly separated by a distance of one tick delay a parameter that is set freely by the user A small tick delay will give rise to more precise but more complex computations The tick rate is the inverse of the tick delay Each element of a life model will describe an constant state of a airs in which neither the FC nor the system evolves It can thus never last more than one tick delay The relation between the two logics is thus established at the levels of models For all practical purposes however we need a relation between formulae which is deduced by the theory of abstract interpretation Thus we compute recursively the best upper approxima tion and the best lower approximation of each real valued formula roughly the lower approximation implies the real valued formula which implies its upper approximation For instance if we want to check whether a real valued formula is satis able we compute its lower and upper approximations and check their satis ability in the decidable FC logic If the lower approximation is satis able then so is the real valued formula If the upper approximation is not satis able then so is the real valued formula It is also possible that none of the above cases applies in which case the real valued formula is unde cided and we recommend to start again with a faster FC The computation performed with the slower FC can be reused see for the exact conditions and the complete theory

*In the speci c case of MTL the resulting logic is thus MTLFC*

*De nition MTLFC syntax A formula of MTLFC abides the following syntax*

*p j j j Uc j j tick*

*where*

*are recursively formulae of MTLFC*

*is a constraint taken from f g*

*c is a rational constant*

*We use the usual temporal abbreviations and precedences page Inter alia U abbreviates U c abbreviates Uc Note that* since the distance is always a natural value we can de ne c as c and

*c as c which would not be possible for real time*

*The semantics of MTLFC is based on in nite timed sequence of states Timed sequences of states are sequences of pairs composed of a state as* in temporal logic and a time which is a natural number for a FC logic Thus such a pair actually describes an interval of real time during which nothing

*changes neither the system nor the FC Each change triggers a new state and tick is true in the previous state when the FC changes*

*De nition A FC timing sequence is an in nite sequence ti i N of nat ural numbers where*

*ti N*

*t initially time is zero*

*ti ti or ti ti time advances of one time unit or stays constant*

*t i ti t time never stops*

*De nition A FC timed trace is an in nite sequence si ti i N*

*s P that is each s is the subset of propositions that are true in the ith* observation of the trace

*i* *i*

*ti i N is a FC timing sequence giving the FC displays*

*De nition MTL semantics A MTL formula holds in at position i noted i j i*

*FC FC*

*i i j p i p si for p P*

*ii i j i i j or i j*

*iii i j i not i j*

*iv i j i i j*

*v i j Uc i k i such that k j and tk ti c and*

*j i j k j j*

*vi i j tick i ti ti*

*Example MTL formulae*

*FC*

*p q a p position is always followed by a q position before the* FC ticks times ensuring thus a real delay strictly less than tick delays Such a formula allows the speci cation of bounded response time though the actual bound is slightly unexpected

*p q a p position is always followed by a q position ticks later Since the FC cannot identify precisely the time of p and q the real delay is* between and

*This logic is thus somewhat imprecise but this is necessary to ensure* decidability see also for a di erent but similarly motivated logic In the next section we show decidability by giving practical algorithms

## *A decision procedure for MTL*

*FC*

*The real time logics are often undecidable and even when they are decidable*

*their decision procedures requires non classical data structures e g open closed hyperpolyhedra The FC logics have a much more classical struc ture than their real time counterpart all techniques from usual temporal logic* apply directly Below we present the classical tableau technique adapted to the case of MTLFC Let us recall that the tableau technique will build a nite

*automaton that recognize exactly the models of the original formula To this* purpose it selects the formulae relevant to the formula to be decided essen tially its sub formulae and considers all possible combinations and all possible transitions between them according to the semantics of the logic

*Fictitious Clock Timed Automata*

*De nition TAFC A FC timed automaton A is composed of*

*i Q a nite set of states*

*ii Q Q the initial states*

*iii E Q Q is the transition relation*

*iv Q P ftickg is a labelling function that labels each location with* a subset of the propositions and the special proposition tick

*v Q Q is a generalized Bu chi acceptance condition de ned below*

*F*

*Note that we do not introduce FCs in the automata but a special propo sition tick FCs would give more succinct automata with the same expressive* power

*The operational semantics of these automata is given below*

*De nition A timed run of a FC automaton A is an in nite sequence of pairs q t q t qn tn where*

*i qi Q*

*ii t t tn is a timing sequence*

*iii q Q the rst state is an initial state of A*

*iv qi qi E the transition rule of A is respected*

*v tick qi i ti ti i e the FC ticks as noted in A Furthermore the run is accepted i for every set F QF there exists* in nitely many positions i such that qi F

*De nition Trace of a run The trace s t s t sn tn of a timed run q t q t qn tn is given by si qi P*

*Theorem The emptiness problem for TAFC i e deciding whether there exists an accepted trace is NLogSpace Complete*

*Note that if we would have introduce discretely valuated clocks the empti ness problem would have been PSpace as automata with clocks are exponen tially more succinct*

*Tableau Procedure*

*We ll show that using a classical tableau method for every formula of the* logic MTLFC we can construct a TAFC A that accepts exactly the models of

*In the next subsection we decide the satis ability of MTLFC by solving the* emptiness problem for TAFC

*The basic sub formulae will give the information to keep in a state*

*De nition Basic sub formulae The set of basic sub formulae of a MTLFC formula is*

*basic p fpg if p P*

*basic p basic p*

*basic basic basic*

*basic basic f g*

*basic Uc basic basic i c f U i g ftickg*

*In the sequel we ll write q also for boolean combinations of basic* formulae the boolean components of are rst evaluated according to the boolean rules and only then basic formulae are checked

*The following equivalences show how the requirement expressed by an* until formula can be decomposed into a requirement on the present state and a requirement on the following state Those equivalences will be used to de ne the transition relation of the automaton A for the formula

*De nition Expansion Formulae Let p be a positive natural number*

*i a U p U p*

*b U*

*ii a U p tick U p tick U p*

*b U tick U*

*iii a U p tick U p tick U p*

*b U U*

*We are now equipped to de ne the FC timed automaton A that accepts* exactly the models of the formula

*De nition MTLFC A A has the following elements*

*i the set of states Q are subsets of basic ftickg*

*ii the initial states contain Q fq Qj qg*

*iii the transition relation is de ned according to the expansion formulae*

*q q E i for every basic its expansion rule is veri ed when* the formulae appearing in the expansion are evaluated in q For example if q q E and U tick q then the expansion formulae of de nition tell us that U q

*iv the labelling function is obvious q q P ftickg*

*v the set of accepting locations QF contains the sets*

*a Ftick fq j tick qg this set ensures that time goes beyond any* bounds i e that the FC ticks in nitely often

*b for every formula g of the form U c basic with f g a set Fg fq j q g qg*

*Theorem The FC timed traces that are accepted by the automaton A are exactly the models of*

*Prophecy Variables*

*In the construction given in de nition c subformulae are introduced for* each formula of the form U So each such formula multiply by c the number of states in A This explosion is not necessary if f g For instance let us examine the formula c U c If this formula is true in position i of timed trace then the formulae d with d c are also true in that position i So instead of maintaining the truth value of all those d with d c we simply have to remember the time distance e that separates the position i from the rst instant when will be true Then we can reconstruct all the information If U is currently true then for all d such that d e d is false and for all d such that d e d is true Otherwise all U are false e is called the value of the prophecy variable for

*c*

*Event Clock Logic*

*This simple idea is captured by the logic of event clocks De nition ECLFC syntax The grammar of ECLFC is*

*p j j j U j y c j tick*

*De nition ECLFC semantics The semantics is as for MTLFC except of course for prophecy variables The value of y in at i noted val i y is*

*val i y tj ti i j i j j and k i k j j j There might be no such j in which case the value is unde ned and any* constraint on an unde ned value is conventionally false The semantics of the prophecy formulae is now obvious

*i j y c i val i y c*

*We now rede ne an optimised notion of basic for ECL*

*FC*

*De nition basic ECLFC formulae and variables The basic sub formulae* of a ECL formula are de ned by

*FC*

*i basic p fpg if p P*

*ii basic basic*

*iii basic basic basic*

*iv basic basic f g*

*v basic U basic basic f Ug*

*vi basic y c fy cg ftickg basic*

*From the basic formulae of the form y c we extract the basic variable y and its maximum m which is the maximum of all c to which y is compared Its range is f m m g m is used to represent any value greater*

*than m represents the unde ned value used when no occurs in the*

*future The value of the basic prophecy variable y determines all its basic* prophecy formulae thus there is no need to record basic prophecy formulae

*De nition The expansion formulae for the temporal operators U* are as for MTLFC The expansion formulae for a prophecy variable y are

*if v then y tick y*

*if v m then y v tick y v tick y v*

*if v m then y*

*v y*

*m y*

*m*

*if v then y v y*

*It simply expresses that the display is incremented at ticks and thus the* distance between the current display and the event is then decremented

*As usual we can de ne for each ECL formula a FC timed automaton* A that accepts exactly the models of

*FC*

*De nition ECLFC A A has the following elements*

*i the set of states Q are pairs s v where s is a subset of the non prophecy* basic formulae the ones that are true now

*ii the initial states contain Q fq Qj qg*

*iii the transition relation is de ned from the expansion as usual For exam ple if s v is such that v y and tick s then for every s v such that s v s v E the expansion formulae of de nition tell us that v y*

*iv the labelling function is de ned by s v s P ftickg*

*v the set of accepting locations QF contains*

*a for every formula g of the form U or y m basic coded*

*as y*

*g*

*m a set F*

*fq j q g qg each promising formula*

*g is followed later by its promised formula*

*b a set Ftick fq j tick qg time goes beyond any bounds*

*Note that although y c is a promise for we do not need to include* speci c accepting locations since the ticks will eventually bring about

*Using ECLFC to decide MTLFC*

*Most operators of MTLFC translate e ciently into ECLFC*

*i U c U y c*

*ii U c U y c*

*iii U c c U*

*iv U c c U*

*In contrast the translation of the equality is necessarily exponential as* expected from the fact that MTLFC is complete in ExpSpace while ECLFCis only complete in PSpace A possible translation

*i U tick U*

*ii U p tick U tick U p*

*Here one can see that the source of the exponential is simply the translation* of the binary constant p into some form of unary notation

*Using our abstract interpretation framework it is possible also to use* approximations By doing so we gain e ciency but loose precision in solving MTLFC Anyway in our approach MTLFC is already an approximation of Albert For instance we can use the following linear translation

*i Uc U y c*

*ii U c U c U c*

# *Sharing trees*

*Introduction*

*Constructing explicitly the automata presented is clearly not e cient enough* to be implemented since the number of their states is exponential in the formula So in this section we present a so called symbolic procedure that permits to implement e ciently the tableau procedure The procedure is symbolic in the sense that neither individual states nor pairs of states of the transition relation of the automaton are explicitly manipulated We use Sharing Trees STs a data structure to represent compactly sets of tuples For the de nition of STs and a comparison with BDDs see The main idea is to ensure su x merging of tuples that share equal ends and pre x merging of tuples that share equal beginnings STs are very close to minimized deterministic nite state automata without loops

*A set of operations are available over STs to e ciently manipulate set of* tuples Here we will use the following operations

*i Union ST ST fx j x ST x ST g*

*ii Intersection ST ST fx j x ST x ST g*

*iii Matching M atch ST i a f x xn j x xn ST xi ag*

*iv Projection x ST fx j y y ST var x var y x var x y g*

*v Empty Empty*

*Those operators are implemented by symbolic algorithms in the sense that* tuples are never explicitly handled but only global operations on the ST are performed In this way the improvement in memory space also yields an improvement in execution time as for BDDs However operations such as matching and projection are more e cient Applications of STs to represent automata can be found in

*Using Sharing Trees in ECLFC Model Generation*

*In the following we use STs to represent symbolically the automaton of a* ECLFC formula The set of states of the automaton can be represented by a

*var x returns the set of variables of the vector x x var x y expresses the fact that x and y give the same value to common variables*

*ST that contains tuples which are a valuation of the basic formulae except* prophecy formulae and variables of

*Given the nite domains of tuples each element xi either boolean for* basic non prophecy formulae or in f m m g for prophecy formulae

*we can add basic operations*

*i True F ullST is the ST that contains all possible combinations*

*ii Complement Compl ST fx j x ST g It represents the negation*

*iii F orm ST i returns the formula of the ith layer of ST*

*iv M atch ST abbreviates M atch ST i where F orm ST i*

*Now we de ne the symbolic representation by STs of the automaton A*

*Q Q T F of a ECL formula*

*FC*

*The set of states Q can simply be represented by F ullST Sometimes we will use the set of reachable states the set of states that can reach an* accepting state the locally consistent states i e the states that can have transitions instead for e ciency reasons The other sets will then be inter sected with Q or more e ciently we can use Q as a relevance set allowing the implementation to choose the most compact representation that preserves the intersection with Q

*Let us now see how to compute symbolically the set of initial states* Q of A The function ST does this by recursion on the formula

*De nition Sharing Tree of a ECL formula The ST of a ECL formula*

*FC FC*

*contains the states that verify*

*if basic y c ST M atch F ullST*

*if y c ST S M atch F ullST y d*

*d c*

*ST Compl ST*

*ST ST ST*

*ST ST ST*

*Note that modal formulae are always basic so that they are already con sidered in this case analysis*

*The transition relation a set of pairs of states will be represented by* a ST with two alternated sets of variables representing respectively the old and the new state as is classical for BDDs in order to take advantage of the direct dependencies between the old and new values of a variable

*There is no need to represent the labelling which is xed*

*Finally we represent symbolically the sets of accepting states simply by* the corresponding STs as for Q

*We use compute symbolically because the ST will be obtained only by performing global operation on STs and no direct manipulations of the tuples that represents atoms*

*Algorithm Transition relation T x x Q Q*

*basic do*

*T x x*

*T x x*

*ST Exp x x*

*In the actual implementation Exp x x is not a function but the expan sions of all possible are pre computed Subformulae beginning with in* the expansion law are evaluated in x

*Computing the emptiness of the symbolic automaton*

*We have shown how to construct symbolically A So to decide whether a* formula is valid we check the emptiness of A

*There exists an algorithm linear in the number of states but since* the number of states is usually exponential this procedure is not interesting here For e ciency the emptiness check must be performed symbolically Two slightly di erent approaches exist an algorithm using the transitive closure of T and another one based on that we use in practice and present here The Emerson Lei formula searches for a reachable accepted cycle i e a cycle which passes through at least one state of each accepting set Due to the generalised Bu chi acceptance condition this is exactly what is needed to

*construct an accepted in nite sequence*

*De nition Accepted cycle An accepted cycle in an automaton is a set* of states C such that

*i Fi F C Fi accepted*

*ii c c C y yn C s t c y T yn c T cycle*

*Algorithm Reaching states*

*This x point computes the set of states that can reach states of Fc by using* edges of some transition relation Tc

*function CanReach Fc Tc S Fc*

*repeat*

*P rec S*

*Sx x x y y*

*S x T S*

*c*

*until S P rec return S*

*We have implemented the two algorithms The algorithm based on the* Emerson Lei formula has given far better results The problem with the al gorithm based on the transitive closure is that it requires the computation of STs with n layers where n is the number of basic formulae This produces

*Algorithm Emptiness based on Emerson Lei C N*

*repeat*

*P rec C*

*F Fdo*

*Tc*

*T x y Cx*

*Fc F C*

*S CanReach Fc Tc*

*C C S until C P rec return C*

*a blow up of the number of nodes such that the computation stops by lack of* memory

*Example Re ning data transmission by CSMA CD*

*A re nement step consists here of two ECLFC formulae and such that re nes i L L Typically is a more operational description of* the system described by the requirements Note that the two descriptions are logical and can thus be of a fairly high level in contrast with model checking that needs an operational description and is thus only applicable after the development

*Example Let us consider a system composed of two computers M M and a single line L that enables the two computers to send messages to the* outside We will specify the requirements of a protocol that will allow the two computers to send messages via the line as well as a description of a solution for these requirements The decision procedure will be used to automatically prove that the described solution respects the formulated requirements

*Let us rst formulate the requirements for the protocol*

*R WToSent BeginToSend InSending U EndSending When the computer M wants to send some message WToSent it eventu ally succeeds to send the message correctly We consider that a transmission of M is correct if it begins with a BeginToSend event and terminates with a EndSending event Between the two events M is in transmission state*

*InSending*

*R is as R but for M*

*R InSending U EndSending InSending U EndSending*

*This requirement imposes that the protocol takes into account that the*

*communication medium the line is not able to transmit messages at the* same time Thus succeeding communications can not overlap each other Those three formulae are high level requirements for the protocol Those

*requirements are highly declarative We think that it would be helpful to*

*derive more detailed and more operational requirements for the protocol before* trying to give an operational solution for example in term of a nite state machine that implements the requirements

*In the following we will concentrate on the re nement of requirement R A possible solution to this requirement is given by the protocol CSMA CD*

*De nition CSMA CD When a computer wants to send a message it* tests whether the line is busy If the line is not busy it begins to send else it waits A collision occurs when more than one computer are transmitting at the same time The delay of propagation plays an important role in the protocol If one computer begins to send the other computers will not immediately see that the computer is sending but they will see it at most later where is the propagation delay between the two most distant computers Consequently a collision may occur between and after a computer has begun to send The noise of the collision can also take to reach the computer which is sending A computer is sure that no collision will occur only after

*In term of the CSMA CD the requirement R can be rewritten in*

*R CD InSending InSending*

*There is a collision when two computers are sending at the same time*

*R InSending U EndSending CD*

*R InSending U EndSending CD*

*If there is a collision during a transmission it cannot succeed*

*Let us now try to re ne the requirement R with the solution proposed* by the CMAS CD protocol We must thus describe a solution or a more operational requirement which implies the requirement R

*I CD MSeeCD*

*When there is a collision the computers see it at latest time units later*

*we take here the propagation delay as time unit Due to the use of the FC* semantics the formula really expresses that the collision is detected before the rd tick

*I InSending InSending BeginToSend The de nition of BeginToSend*

*I InSending InSending EndSending MSeeCD*

*If the computer stops sending either it is because it has nished to send its* message EndSending or because it has detected a collision MSeeCD

*I EndSending InSending InSending*

*To stop sending M must be sending and just after M does not send*

*I MSeeCD InSending*

*If M detects a collision it stops to send*

*I EndSending MSeeCD*

*A collision can not be considered as a normal end of transmission*

*I BeginToSend EndSending*

*To ensure that a computer always detects a collision during one of its trans missions the duration of the messages must be at least Due to the FC* semantics we have modeled this requirement by constraining the begin and the end of a transmission to be distant of at least ticks

*I InSending CD CD U InSending*

*If M is in transmission state at least during the time that separates ticks* of the FC and no collision occurs then no collision will occur before M ends to send All other computers have seen that the line was busy

*I InSending*

*Initially the computers do not send a message*

*We must now show that I I is a re nement of requirement R We* have submitted the formula I I R to our decision procedure it was proven instantaneously

*This shows that although this problem is not naturally expressed in the* FC semantics it can be translated automatically and solved there We hope to generalize these ndings to equip our real time speci cation language with a tool box of translators and solvers integrated by an interactive proof system

*Conclusions*

*Summary*

*We have explored techniques to re ne real time requirements We have rst* de ned a FC real time logic called MTL It is used as an abstraction of the Albertlanguage We have shown the ability of the logic for the speci cation of properties of concurrent and reactive systems We have presented a procedure to construct the automaton that accepts exactly executions corresponding to the models of a formula of MTL logic The theoretical analysis showed that a slightly di erent logic ECL would be more e cient without loss of expressiv ity In practice we use the union of the two logics that are intertranslatable Due to the high memory consumption foreseen by theory this procedure has been implemented symbolically using STs A small example coming from prac tice which is only proved with di culty by humans showed the applicability of the approach

*FC*

*FC*

*FC*

*Future work*

*The following problems are addressed in other papers*

*Logics with real can be decidable present a decidable continuous time* logic constructs the associated automata and monadic logics which are

*also decidable Thus we could extend our logics in the spirit of without* loosing decidability

*The theory of abstract interpretation that we use is presented in We* are preparing a more general version

*The logic is propositional and has no structuring means In we de ne a* continuous time structured rst order speci cation language that shares its semantic foundations with the logics presented here

*The methodology to obtain formal requirements and its integration into ex isting methods are currently under investigation see for preliminary* thoughts

*We intend to deal with the following problems in the future*

*the state explosion problem limits the formulae that can be treated auto matically So there is no hope to deal with complete problems with this* type of technique alone

*It is why we think that the decision procedure should better be used within* an interactive theorem prover equipped with the theory of abstract inter pretation the type of procedure presented here would serve as a means to abstract more expressive logics and to automate part of proofs The abstraction can either be

*i automatic as we intend to do for the FC abstraction where a good tick* rate can be obtained by doubling it until success or memory exhaustion

*ii manual for many abstractions a proof showing the validity of the tech nique is required*

*Thus an interactive prover provides the glue needed to put the results of* automatic techniques into a safe use Too often the proponents of model checking use models that have been devised manually and whose correspon dence with the real system is questionable

*References*

*R Alur Techniques for Automatic Veri cation of Real Time Systems PhD thesis Stanford University*

*R Alur C Courcoubetis and D Dill Model checking in dense real time Information and Computation Preliminary version appears in the Proc of th LICS*

*R Alur and D Dill A theory of timed automata Theoretical Computer Science Preliminary version appears in Proc th ICALP*

*LNCS*

*R Alur T Feder and T Henzinger The bene ts of relaxing punctuality Journal of the ACM*

*R Alur and T Henzinger Real time logics complexity and expressiveness Information and Computation Preliminary version appears in the Proc of th LICS*

*R Alur and T Henzinger A really temporal logic Journal of the ACM*

*Preliminary version appears in Proc th FOCS*

*H Barringer R Kuiper and A Pnueli A really abstract concurrent model and its temporal logic In Proceedings of the th Annual Symposium on Principles of Programming Languages pages ACM Press*

*R Bryant Symbolic boolean manipulation with ordered binary decision diagrams ACM Computing Surveys*

*J Burch E Clarke K McMillan D Dill and L Hwang Symbolic model checking states and beyond In Proceedings of the th Symposium on Logic in Computer Science pages Philadelphia June*

*S Campos E Clarke W Marrero M Minea and H Hiraishi Computing quantitative characteristics of nite state real time systems In Proceedings of the th Annual Real time Systems Symposium IEEE Computer Society Press*

*F Chabot L Ferier J F Raskin and P Y Schobbens The formal semantics of Albert II Technical report University of Namur*

*P Cousot and R Cousot Abstract interpretation A uni ed lattice model for static analysis of programs by construction or approximation of xpoints In Conference Record of Fourth ACM Symposium on Programming Languages*

*POPL pages Los Angeles California Jan*

*P Du Bois Intuitive de nition of the Albert II language Technical report Computer Science Department University of Namur Namur Belgium February*

*E Dubois P Du Bois and M Petit Albert an Agent oriented Language for Building and Eliciting Requirements for real Time systems In Proc of the*

*th Hawaii International Conference on System Sciences HICSS Maui*

*Hawaii January IEEE*

*E Dubois P Du Bois and A Rifaut Elaborating structuring and expressing formal requirements of composite systems In P Loucopoulos editor Proc of the th conference on advanced information systems engineering CAiSE pages Manchester UK May LNCS Springer Verlag*

*E Emerson and C L Lei Temporal model checking under generalized fairness constraints In Proc th Hawaii International Conference on System Sciences Hawaii*

*F Gagnon J C Gregoire and D Zampunieris Sharing trees for on the y veri cation In Proceedings of the th International IFIP Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*

*FORTE Montreal Quebec IEEE*

*T Henzinger P H Ho and H Wong Toi HyTech the next generation In Proceedings of the th Annual Real time Systems Symposium pages IEEE Computer Society Press*

*T Henzinger J F Raskin and P Y Schobbens The regular real time languages In K Larsen S Skyum and G Winskel editors ICALP Automata Languages and Programming Lecture Notes in Computer Science*

*pages Springer Verlag*

*P Heymans and E Dubois Scenario based techniques for supporting the elaboration and the validation of formal requirements Requirements Engineering Journal to appear*

*R Koymans Specifying real time properties with metric temporal logic Real time Systems*

*R Koymans Specifying message passing and time critical systems with temporal logic LNCS Springer Verlag*

*R Koymans J Vytopil and W de Roever Specifying message passing and time critical systems with temporal logic Doctoral dissertation Eindhoven University of Technology Eindhoven The Netherlands*

*Z Manna and A Pnueli The Temporal Logic of Reactive and Concurrent Systems Speci cation Springer Verlag Berlin January*

*J S Ostro Temporal Logic for Real Time Systems Advances Software Development Series Research Studies Press Ltd Wiley*

*R Paquay and D Zampunieris Mec with sharing trees Research Paper RP*

*Computer science department FUNDP Namur Belgium mai*

*A Pnueli The temporal logic of programs In Proc th IEEE Symposium on Foundation of Computer Science pages*

*J F Raskin Logics Automata and Classical Theories for Deciding Real Time PhD thesis Institut d Informatique FUNDP Namur June*

*J F Raskin and P Y Schobbens Real time logics Fictitious clock as an abstraction of dense time In Proc Third International Workshop on Tools and Algorithms for the Construction and Analysis of Systems TACAS volume*

*of Lecture Notes in Computer Science LNCS pages Springer*

*J F Raskin and P Y Schobbens State clock logic a decidable real time logic In Proc International Workshop on Real time and Hybrid Systems HART volume of Lecture Notes in Computer Science LNCS pages Springer*

*J F Raskin P Y Schobbens and T Henzinger Axioms for real time logics In D Sangiorgi and R de Simone editors Proceedings of CONCUR th International Conference on Concurrency Theory volume of Lecture Notes in Computer Science LNCS Springer*

*P Y Schobbens Albert at the age of ve In Object Oriented Software Development number in Dagstuhl Seminar Bericht April*

*P Y Schobbens and J F Raskin The logic of event clocks Journal of Algebra Languages and Combinatorics To appear*

*P Y Schobbens J F Raskin and T A Henzinger Axioms for real time logics Theoretical Computer Science Submitted*

*R Tarjan Depth rst search and linear graph algorithms SIAM Journal of Computing*

*P Wolper Temporal logic can be more expressive Information and Control*

*P Wolper The tableau method for temporal logic An overview Logique et Analyse*

*E Yu P Du Bois E Dubois and J Mylopoulos From organization models to system requirements a cooperating agents approach In Proc of the Third International Conference on Cooperative Information Systems CoopIS*

*Vienna Austria May University of Toronto Press inc*

*D Zampuni eris The Sharing Tree Data Structure Theory and Applications in Formal Veri cation PhD thesis Institut d Informatique FUNDP Namur May*

*D Zampuni eris and B Le Charlier An e cient algorithm to compute the synchronized product In Proceedings of Modeling Analysis and Simulation of Computer and Telecommunication Systems MASCOTS Durham North Carolina USA Jan IEEE*

*D Zampuni eris and B Le Charlier E cient handling of large sets of tuples with sharing trees In Proceedings of the Data Compression Conference DCC Snowbird Utah USA Mar IEEE*