Non-elementary Lower Bound for Propositional Duration Calculus

A. Rabinovich
Department of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel

1 Introduction

The Duration Calculus (DC) [5] is a formalism for the specification of real time systems. DC is based on interval logic [6] and uses real numbers to model time. DC was successfully applied in case studies of software embedded systems, e.g., a gas burner [10], a railway crossing [12] and was used to define the real time semantics of other languages.

Peter Sestoft [11] has shown that the satisfiability problem for the Propositional fragment of Duration Calculus (PDC) under non-standard discrete time interpretation has a non-elementary complexity. We show that PDC under the standard continuous time interpretation has also a non-elementary complexity. In order to prove this lower bound we provide a linear time reduction from the equivalence problem for star free expressions to the equivalence problem for PDC. Meyer and Stockmeyer proved in [14] that the equivalence problem for star free expressions is non-elementary. In our proof we rely on the relationship between PDC under the standard continuous time interpretation and star-free expressions under (non-standard) stuttering free interpretation [9].

2 Star free expressions

The (extended) star free regular expressions over an alphabet Σ [8] are defined by the following grammar:

$$E ::= \ l \mid E + E \mid E; E \mid \neg E, \text{ where } l \text{ ranges over } \Sigma.$$

In this paper we will use "star free expression" for "extended star free regular expression".

The standard interpretation assigns to a star free expression a set of string (language) over Σ. In this interpretation sum $(+)$ is interpreted as union, sequential composition $(;)$ is interpreted as concatenation and negation $(\neg)$ is interpreted as the complementation relative to the set of all finite strings (excluding the empty string ϵ).

We use $[\![E]\!]$ for the language assigned to the star free expression E under the standard interpretation. Expressions E^1 and E^2 are said to be equivalent (under the standard interpretation) if $[\![E^1]\!] = [\![E^2]\!]$.

Theorem 1 *[14] The equivalence problem for star free expressions over the alphabet* $\{0, 1\}$ *has a non-elementary complexity.*

Remark 2 *(1) The lower bound given in [14] for the space complexity of the equivalence problem for star free expressions is $exp_{log_2(n)}(1)$, where $exp_m(k)$ is the m-times iterated exponential function (e.g. $exp_2(k) = 2^{2^k}$). This lower bound was improved in [4] to $exp_{\frac{n}{log(n)}}(1)$. The straightforward decision algorithm translates the expressions to finite state automata and then tests the equivalence of the languages defined by these automata. The complexity of this algorithm is $exp_n(1)$. An excellent survey of lower bounds is provided in [3].*

(2) The star free expressions defined above do not contain a symbol for the empty string. Also, the languages assigned to the expressions do not contain the empty string. The star free expressions considered in [14] contain a symbol for the empty word; their complementation is defined relatively to the set of all finite strings including the empty string. Remark 4.23 in [13] shows that these differences do not influence the lower bound.

3 Stuttering Free Interpretation

Definition 3 *(Stuttering [7]) A string $l_0 l_1 \ldots l_n$ is stuttering free if $l_i \neq l_{i+1}$ for $i < n$. A language is stuttering free if it contains only stuttering free strings.*

Let us consider stuttering free interpretations of negation and of sequential composition symbols. Namely, let $\neg$ be the complementation relative to the set of stuttering free strings and let the sequential composition be interpreted as the following operation $\star$ on strings:

$$l_0 \ldots l_p \star m_0 \ldots m_k = \begin{cases} l_0 \ldots l_p m_1 \ldots m_k & \text{if } l_p = m_0 \\ l_0 \ldots l_p m_0 \ldots m_k & \text{otherwise.} \end{cases}$$

Sum, like before, is interpreted as union. The stuttering free interpretation assigns to a star free expression E the stuttering free string language which will be denoted by $[\![E]\!]^{stut}$.

Expressions E^1 and E^2 are said to be equivalent under the stuttering free interpretation if $[\![E^1]\!]^{stut} = [\![E^2]\!]^{stut}$.

Proposition 4 *There exists a linear time reduction from the equivalence problem for star free expressions over the alphabet $\{0,\ 1\}$ under the standard interpretation to the equivalence problem for star free expressions over the alphabet $\{0,\ 1,\ 2\}$ under the stuttering free interpretation.*

Proof: Given a star free expression E over $\{0,\ 1\}$, we construct (in linear time) a star free expression E_{sf} over $\{0,\ 1,\ 2\}$ such that

$$l_0 l_1 \ldots l_p \in [\![E]\!] \text{ if and only if } l_0 2 l_1 2 \ldots l_p 2 \in [\![E_{sf}]\!]^{stut} \tag{1}$$

Let ALL be the star free expression $\neg 0 + \neg 1$. It is clear that under the stuttering free interpretation ALL describes the set of all stuttering free strings over the alphabet $\{0,\ 1,\ 2\}$. We say that a string is well-formed if it has the form $l_0 2 l_1 2 \ldots l_p 2$, where $l_i \in \{0,\ 1\}$. Let WF be the star free expression $\neg((ALL; 0; 1; ALL) + (ALL; 1; 0; ALL) + (2; ALL) + (ALL; 0) + (ALL; 1))$. It is clear that under the stuttering free interpretation WF describes the set of all well-formed strings.

We define E_{sf} by the structural induction as follows:

1. $0_{sf} \triangleq 0; 2$.

2. $1_{sf} \triangleq 1; 2$.

3. $(E^1 + E^2)_{sf} \triangleq (E^1)_{sf} + (E^2)_{sf}$.

4. $(E^1; E^2)_{sf} \triangleq (E^1)_{sf}; (E^2)_{sf}$.

5. $(\neg E)_{sf} \triangleq \neg((E)_{sf} + \neg WF)$, i.e., $(\neg E)_{sf}$ denotes the complementation of $(E)_{sf}$ with respect to the set of well formed strings.

By the structural induction on the expressions it is easy to show that (1) indeed holds and that all strings in $[\![(E)_{sf}]\!]^{stut}$ are well-formed. Hence, $[\![E^1]\!] = [\![E^2]\!]$ if and only if $[\![(E^1)_{sf}]\!]^{stut} = [\![(E^2)_{sf}]\!]^{stut}$. It is also clear that E_{sf} is computable from E in the time proportional to the size of E. $\square$

We do not know whether, the reductions in Proposition 4 can be "reversed", i.e., whether there exists a polynomial reduction from the equivalence problem for star free expressions under the stuttering free interpretation to the equivalence problem for star free expressions under the standard interpretation.

4 Propositional Duration Calculus

The Duration Calculus [5] is a formalism for the specification of real time systems. The Propositional Duration Calculus (called the restricted duration calculus in [1, 2]) is a fragment of the duration calculus where metric properties are ignored.

A run of a real time system is represented by a function from non-negative reals into a set of values - the instantaneous states of a system. Such a function will be called a signal. Usually, there is a further restriction on the behavior of continuous time systems. For example, a function that gives value q_0 for the rationals and value q_1 for the irrationals is not accepted as a 'legal' signal.

A requirement that is often imposed in the literature is that in every finite length time interval a system can change its state only finitely many times. This requirement is called finite variability requirement.

Below we first describe a connection between finite variability functions and stuttering free strings. Then we recall the syntax and the semantics of the Propositional fragment of Duration Calculus (PDC). Finally, we give a linear time reduction from the equivalence problem for star free expressions under the stuttering free interpretation to the equivalence problem for PDC.

4.1 Finite variability functions

A function η from a subinterval $[a,\ b]$ of the reals into a finite set Σ has finite variability if there exists a finite increasing sequence $a = a_0 < a_1 < a_2 \ldots < a_n = b$ such that η is constant on every interval (a_i, a_{i+1}).

The restriction of η on an interval [c, d] is denoted by $\eta \restriction [c,\ d]$. Notice that if $\eta :\ [a,\ b] \to \Sigma$ has finite variability and $[c,\ d] \subseteq [a,\ b]$, then $\eta \restriction [c,\ d]$ has finite variability.

The following lemma is straightforward.

Lemma 5 *Suppose that $\eta :\ [a,\ b] \to \Sigma$ has finite variability, then there exists a unique increasing sequence $a = a_0 < a_1 < a_2 \ldots < a_n = b$ such that*

1. *η is almost constant on every interval $(a_i,\ a_{i+1})$, i.e., for every i there is $l_i \in \Sigma$ such that the set $\{x \in (a_i,\ a_{i+1})\ :\ \eta(x) \neq l_i\}$ is finite.*

2. *For every $i < n-1$, the value of η on (a_i, a_{i+1}) differs from the value of η on (a_{i+1}, a_{i+2}).*

Definition 6 *(Trace of a finite variability function.) Let η be a finite variability function over $[a,\ b]$ and let $a_0, \ldots, a_n$ be as in lemma 5. Let l_i be the values of η on (a_i, a_{i+1}). The trace of η (notations* $\mathbf{trace}(\eta)$*) is the stuttering free string $l_0 l_1 \ldots l_{n-1}$.*

Lemma 7 *Suppose $\eta :\ [a,\ b] \to \Sigma$ and $c \in (a,\ b)$. Then $\mathbf{trace}(\eta)= \mathbf{trace}(\eta \restriction [a,\ c]) \star \mathbf{trace}(\eta \restriction [c,\ b])$, where $\star$ is stuttering free concatenation (see section 3).*

Remark 8 *(Trace of a tuple.) Let $\langle \eta_1, \ldots, \eta_n \rangle$ be an n-tuple of finite variability functions from $[a,\ b]$ into $\{0,\ 1\}$. With this n-tuple we associate a function η from $[a,\ b]$ into $\{0, 1, \ldots, 2^n - 1\}$ defined as $\eta(t) = i$ if $\langle \eta_1(t), \ldots, \eta_n(t) \rangle$ is the binary representation of i. The above mapping defines a one-to-one correspondence between the set of n-tuple of finite variability functions from $[a,\ b]$ into $\{0, 1\}$ and finite variability functions from $[a,\ b]$ into $\{0, \ldots, 2^n - 1\}$. The trace of an n-tuple (notations $\mathbf{trace}(\eta_1, \ldots, \eta_n)$) is defined as the trace of the corresponding function.*

4.2 Syntax of PDC

PDC has two syntactical categories: state expressions and formulas. The state expressions and the formulas over a set *VAR* of variables are defined as follows:

State Expressions: The *state expressions* are constructed from the state variables by propositional connectives. We will use S to range over the state expressions which are defined by the following grammar:

$$S ::= X \mid S \vee S \mid \neg S, \text{ where } X \text{ is a state variable.}$$

Atomic Formulas of PDC: if S is a state expression, then $\lceil S \rceil$ is an atomic formula of PDC.

Formulas: The formulas of PDC are defined by the following grammar:

$$D ::= \ At \ | D^\frown D \mid \neg D \mid D \vee D \text{ , where } At \text{ ranges over the atomic formulas of PDC.}$$

The binary operation $^\frown$ is called chop.

4.3 Semantics of PDC

A valuation θ over an interval $[a,\ b]$ is a function that assigns to every state variable X a finite variability function from $[a,\ b]$ into $\{0,\ 1\}$. A valuation θ straightforwardly extends to state expressions using the meaning of the propositional connectives pointwise. We use the notation $[\![S]\!]^{DC}\theta$ for the function assigned to the state expression S under the valuation θ. It is clear that $[\![S]\!]^{DC}\theta$ has finite variability.

The satisfaction relation $\models$ between PDC formulas and valuations over a positive length interval $[a,\ b]$ is defined as follows:

PDC Atomic Formulas: $\theta,\ [a,\ b] \models \lceil S \rceil$ if there is no positive length subinterval of $[a,\ b]$ where $[\![S]\!]^{DC}\theta$ is constant and equal to 0.

The meaning for disjunctions and negation is defined as usual.

$\theta, [a,\ b] \models D_1 \vee D_2$ iff $\theta, [a,\ b] \models D_1$ or $\theta, [a,\ b] \models D_2$.

$\theta, [a,\ b] \models \neg D$ iff not $\theta, [a,\ b] \models D$.

Let us denote by $\theta \upharpoonright [c,\ d]$ the valuation that maps every state variable X to the restriction of $\theta(X)$ on $[c,\ d]$.

Chop: $\theta,\ [a,\ b] \models D_1 \frown D_2$ if $\theta \upharpoonright [a,\ m],\ [a,\ m] \models D_1$ and $\theta \upharpoonright [m,\ b],\ [m,\ b] \models D_2$ for some $m \in (a,\ b)$.

PDC formulas D^1 and D^2 are said to be equivalent if for every interval $[a,\ b]$:

$$\theta, [a,\ b] \models D^1 \text{ whenever } \theta, [a,\ b] \models D^2.$$

The decision algorithm from [1] for the equivalence (satisfiability) of PDC formulas is not elementary and has the space complexity $exp_n(1)$.

4.4 Reduction

A non-elementary lower bound for the equivalence (as well as for the satisfiability) problem for PDC follows from theorem 1, proposition 4 and the following

Proposition 9 *There exists a linear time reduction from the equivalence problem for star free expressions over the alphabet* $\{0,\ 1,\ 2\}$ *under the stuttering free interpretation to the equivalence problem for PDC.*

Proof: Given a star free expression E over the alphabet $\{0,\ 1,\ 2\}$, we construct (in linear time) a PDC formula E_{DC} which will contain two state variables X_0 and X_1. Hence, to a valuation θ for E_{DC} corresponds a pair $\langle \theta(X_0),\ \theta(X_1) \rangle$ of finite variability functions. The trace of θ is a string over $\{0,1,2,3\}$ defined as the trace of $\langle \theta(X_0),\ \theta(X_1) \rangle$ (see Remark 8 in Section 4.1).

Our translation will guarantee that

$$\theta,\ [a,\ b] \models E_{DC} \text{ if and only if } \mathbf{trace}(\theta) \in [\![E]\!]^{stut}. \tag{2}$$

Let $TRUE$ be the PDC formula $\lceil X_0 \vee \neg X_0 \rceil$. Let *Legal* be the formula $\neg(TRUE \frown \lceil X_0 \wedge X_1 \rceil \frown TRUE \vee TRUE \frown \lceil X_0 \wedge X_1 \rceil \vee \lceil X_0 \wedge X_1 \rceil \frown TRUE)$. Observe that $\theta,\ [a,\ b] \models$ *Legal* if $trace(\theta)$ is a stuttering free string over $\{0,\ 1,\ 2\}$.

Define E_{DC} by the structural induction on the expressions:

1. $0_{DC} \triangleq \lceil \neg X_0 \wedge \neg X_1 \rceil$. Hence $\theta, [a,\ b] \models 0_{DC}$ iff $\mathbf{trace}(\theta) = 0$.
2. $1_{DC} \triangleq \lceil \neg X_0 \wedge X_1 \rceil$. Hence $\theta, [a,\ b] \models 1_{DC}$ iff $\mathbf{trace}(\theta) = 1$.
3. $2_{DC} \triangleq \lceil X_0 \wedge \neg X_1 \rceil$ Hence $\theta, [a,\ b] \models 2_{DC}$ iff $\mathbf{trace}(\theta) = 2$.
4. $(E^1 + E^2)_{DC} \triangleq (E^1)_{DC} \vee (E^2)_{DC}$.
5. $(\neg E)_{DC} \triangleq \neg(E)_{DC} \wedge$ *Legal*. Hence, θ satisfies $(\neg E)_{DC}$ if it does not satisfy $(E)_{DC}$ and $\mathbf{trace}(\theta)$ is a stuttering free string over $\{0, 1, 2\}$.
6. $(E^1; E^2)_{DC} \triangleq D_1 \frown D_2$, where $D_i = (E^i)_{DC}$ for $i = 1,\ 2$.

By the structural induction it is easy to establish that condition (2) indeed holds; The only non-immediate case is the case for sequential composition which follows from lemma 7. It is also clear that E_{DC} is computable from E in the time proportional to the size of E.

From the observation that for every stuttering free string s there exists a valuation θ such that $trace(\theta) = s$ and from (2) above, it follows that $[\![E^1]\!]^{stut} = [\![E^2]\!]^{stut}$ if and only if $(E^1)_{DC}$ is equivalent to $(E^1)_{DC}$. This completes the proof of the proposition. □

Finally, let us note that the reduction in Proposition 9 can be "reversed". Namely, there exists a linear reduction from from the equivalence problem for PDC to the equivalence problem for star free expressions under the stuttering free interpretation.

Acknowledgments

I would like to thank the anonymous referees for their helpful comments.

References

[1] Zhou Chaochen, M. R. Hansen and P. Sestoft. Decidability and undecidablity results for Duration Calculus. In *STACS'93,* Lect. Notes in Comp. Sci. vol 665, pages 58-68, 1993.

[2] Zhou Chaochen and M. R. Hansen. M. Hansen and Zhou Chaochen. Duration Calculus: Logical Foundations. In *Formal Aspects of Computing*, 9:283-330, 1997.

[3] K. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories, Annals of Pure and Applied Logic, 48:1-79, 1990.

[4] M. Fürer. Nicht-elementare untere Schranken in der Automaten-theorie. Doctoral Thesis, ETH, Zurich, 1978.

[5] Zhou Chaochen, C.A.R. Hoare and A. P. Ravn. A calculus of Duration. *Information processing Letters,* 40(5):269-279, 1991.

[6] J. Halperin, B. Moszkowski and Z. Manna. A propositional modal logic of time intervals. In *LICS*, 1986, pp. 279-292.

[7] L. Lamport. The Temporal Logic of Actions. ACM Transactions on Programming Languages and Systems, 16(3), pp. 872-923, 1994.

[8] R. McNaughton and S. Papert. Counter-free automata. The MIT Press, 1971.

[9] A. Rabinovich. On expressive completeness of Duration and Mean Value Calculi. In *Proceedings of EXPRESS*, Electronic Notes in Theoretical Computer Science, Vol. 7, 1997.

[10] A. Ravn, H. Richel and K. Hansen. Specifying and verifying requirement of real time systems. *IEEE Transaction on Software Eng.*, 1993.

[11] P. Sestoft. Personal communication.

[12] J. Skakkebak, A. Ravn, H. Richel, Zhou Chaochen. Specification of Embedded Real time Systems. In Proc. Workshop on Real Time Systems. IEEE Computer Society Press, 1992.

[13] L. Stockmeyer. The complexity of decision problems in automata and logic, Ph.D. Thesis, MIT, 1974.

[14] L. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time: Preliminary Report. In *Proc. 5th AMS Symposium on Theory of Computing*, 1973.