Querying Trajectories through Model Checking based on Timed Automata

Diego V. Simões S.[1,2], Henrique Viana[1], Nicolas Markey[3], Jose Antônio F. de Macedo[1]

[1] UFC - Universidade Federal do Ceará, Brazil
{diegovss, henriqueviana, jose.macedo}@lia.ufc.br
[2] UNILAB - Universidade da Integração Internacional da Lusofonia Afro-Brasileira, Brazil
[3] LSV, CNRS & ENS Cachan, France
markey@lsv.ens-cachan.fr

Abstract. The popularization of geographical position devices (e.g. GPS) creates new opportunities for analyzing behavior of moving objects. However, such analysis are hindered by a lack of semantic information associated to the basic information provided by GPS. Previous works propose semantic enrichment of trajectories. Through the semantic enrichment, we could check which trajectories have a given moving sequence in an application. Often, this sequence is expressed according to the semantic application, using the approach of semantic trajectories proposed in the literature. This trajectory can be represented as a sequence of predicates that holds in some time interval. However, the solutions for querying moving sequence proposed by previous works have a high computational cost. In this paper, we propose an expressive query language to semantic trajectories that allows temporal constraints. To evaluate a query we will use model checking based on timed automata, that can be performed in polynomial time. As this model checking algorithm is not implemented yet, we propose to use UPPAAL tool, that can be more expensive theoretically, but we expected that will be efficient for our approach. In addition, we will present a query example that demonstrates the expressive power of our language. Although in this paper we will focus on semantic trajectories data, our approach is general enough for being applied to other purposes.

Categories and Subject Descriptors: H.Information Systems [**H.m. Miscellaneous**]: Core Database Foundations and Technology

General Terms: Query Languages and User Interfaces

Keywords: query language, semantic trajectory, temporal logic, model checking, timed automata, uppaal

1. INTRODUCTION

The wide availability of geographical location data has been providing new kinds of trajectories analysis applications. Each trajectory, that has an identifier, is performed by a moving object. It is represented by an ordered sequence of location points with a time instant (i.e. latitude, longitude, instant). However, this representation has a lack of semantic information about the places that the trajectory has visited or the activities that it has performed. Some works propose semantic enrichment of trajectories, representing them as a sequence of episodes (i.e. stops and moves) with a set of annotations [Yan et al. 2010; Yan et al. 2011]. An annotation denotes which activity, transportation or location is related with a given episode. Thus, given the representation as previously exposed, efficient techniques for query evaluation involving temporal constraints are needed.

In this paper, we study the problem of query trajectories with semantic information, as the trajectories presented in [Yan et al. 2011], emphasizing how to evaluate a query language for this purpose and how hard this should be. In order to clarify our problem, consider the example showed below

Este trabalho foi parcialmente financiado pelo CNPq (473110/2008-3, 473110/2008-3, 557128/2009-9, 483552/2009-7) e pela FUNCAP (GPF 2151/22). Agradecimentos à UFC e à UNILAB pelos recursos disponibilizados.

such that each tuple has the form (s, e, Ψ), where s represents the initial time, e represents the final time and Ψ represents the set of predicates that holds during this time interval. The capital letters mean the following predicates: H - At Home, W - Working, U - At University and D - Dining.

—**Trajectory 1:** $\{(1, 3, [H]), (4, 6, [W, U]), (7, 8, [H])\}$;
—**Trajectory 2:** $\{(1, 5, [W]), (6, 7, [H]), (8, 10, [H, D])\}$.
—**Trajectory 3:** $\{(1, 2, [W]), (3, 5, [U]), (6, 7, [H]), (8, 10, [U, D])\}$;

In this application scenario, there are some queries to do. These queries must allow checking which trajectories match a given moving pattern. A moving pattern is a sequence of predicates with temporal constraints. Temporal constraint specifies the duration of a predicate and duration between predicates. For instance, we can make the following query:

> Q_1: *Which trajectories perform a working activity during at least two time units, then, after at most four time units, stay at home and then have dinner?*

For Q_1 query, the trajectories where this property is true are 2 and 3. Note that the predicate *At Home* occurs after the predicate *Working* and *At Home* holds after, at most four time units, the predicate *Working*, in trajectories 2 and 3. To perform this query, is necessary a query language and an efficient evaluation procedure of the language that allows temporal constraints. Aiming to solve this problem, we propose a query language and an evaluation involving a temporal logic model checking based on timed automata that has polynomial complexity upper-bound.

There are works in the literature that consider pattern matching techniques to evaluate queries, however either they do not allow the verification of a predicate set in a given time interval [Dindar 2008] or they do not allow temporal constraints [Cadonna et al. 2011]. Some approaches that consider trajectories, do not explain how the query will be evaluated [Bogorny et al. 2009], or do not aim semantic trajectories (e.g. our trajectory) with temporal constraints [Vieira et al. 2010] or they have a high complexity although they are automaton based [Gomez and Vaisman 2009]. Other work defines the query in an algebraic form and the query evaluation could be exponential in the worst case [Sakr and Güting 2011]. In addition, approaches that use temporal logics are not found.

The contributions of this work can be summarized as follows: (i) we introduce and define a query language that allows to express moving patterns with temporal constraints (Section 3); (ii) we propose a semantic model to trajectories (Section 3); (iii) we propose a query evaluation through Model Checking based on Timed Automata with polynomial complexity (Section 4); (iv) we show an evaluation example in UPPAAL model-checking tool-suite (Section 4.2).

2. PRELIMINARIES

Aiming to understand the approach proposed by this paper, previous concepts involving timed automata and Timed-CTL logic are necessary to summarize.

2.1 Timed Automata

A timed automaton extends classical automaton with a set of variables that represents time, called clock (X), which increases its value synchronously with time and includes constraints about these variables. Every transition of timed automata has a constraint over clock values represented by $\mathcal{C}(X)$, called guards, which indicates when the transition can be performed and are represented by the form $x \bowtie c$ with $x \in X$, $\bowtie \in \{=, <, \leq, >, \geq\}$ and $c \in \mathbb{N}$. Another label in transitions is that represents a set of clocks (2^X) to be reset [Bouyer and Laroussinie 2010]. In addition, every state $q \in Q$ in the timed automaton can be constrained by an invariant, which restricts the possible values of a clock variable (i.e. $Inv : Q \rightarrow \mathcal{C}(X)$). Finally, $L : Q \rightarrow 2^{AP}$ labels every state with a subset of atomic propositions AP (i.e. the propositions that holds in a specific state).

A state q can have many configurations defined as a pair (q, v) where $q \in Q$ and $v \in \mathbb{N}^x$ is the clock valuation defined in natural set. The initial location has the configuration (q_0, v_0) with v_0 mapping all clocks from X to 0. These configurations of each state represent the semantics of a timed automaton, defined by a Timed Transition System (TTS) [Bouyer and Laroussinie 2010]. In order to show a timed automaton and how its semantics is, consider the Figure 1.

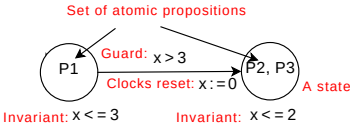


Fig. 1: Overview of a timed automaton.

Considering x the only clock variable, the first state of the automaton of Figure 1 may have the following configurations: $(q_0,\ x = 0)$, $(q_0,\ x = 1)$, $(q_0,\ x = 2)$, $(q_0,\ x = 3)$. The transition to the second state only occurs when $x > 3$. For more details see [Bouyer and Laroussinie 2010].

2.2 TCTL

To represent reactive system properties, it is common to use temporal logics to qualify propositions according to the time. In classical logic the truth value of a proposition does not change along time and has no modalities. In temporal logics the propositions can be true in some time instants as in our trajectory, where predicates hold in some time intervals [Huth and Ryan 2004]. Some examples of common propositions expressed in temporal logics are: "I will drink water", "I am always thirsty". This approach allows verifying whether a finite state system satisfies a given property through *model checking* techniques. As we are dealing with elapsing time by an event, the temporal logics common used are called *quantitative temporal logics*. For our case we will use the extension of CTL (*computation tree logic*), called TCTL (*Timed-CTL*). The syntax is defined by the following grammar, where φ and ψ are formulae of TCTL [Laroussinie et al. 2002]:

$$\varphi, \psi \rightarrow P \mid \neg\varphi \mid \varphi \wedge \psi \mid E\varphi U_{\sim c}\psi \mid A\varphi U_{\sim c}\psi$$

such that $P \in AP$ (atomic propositions), $\sim \in \{<, \leq, \geq, >\}$ and c any natural number.

For this paper, the constructions that will be used are $E\varphi U_{\sim c}\psi$ and its abbreviation $EF_{\sim c}\varphi$ (for $E\top U_{\sim c}\varphi$), as well as standard abbreviations $\top$, $\bot$, $\varphi \vee \psi$. When the modalities have no subscripts, it will be a shorthand for $U_{\geq 0}$ and $F_{\geq 0}$.

In order to illustrate the TCTL semantics, we show the following examples $EF(P_1)$, $EF_{\leq 10}P_1$ and $E(P_1 U_{\geq 5} P_2)$. The first formula means that exists a path where P_1 will occur; the second formula means that exists a path in the A where P_1 will occur in at most 10 steps and the last formula means that exists a path where P_1 will occur during at least 5 steps until P_2 occurs. The full definition of TCTL as well as its semantics can be seen in [Laroussinie et al. 2004].

3. DATA MODEL

We assume that our trajectory is a sequence of events $evn = (s, e, \Psi)$, such that s and e are respectively the start and the end time of the event with $s < e$ such that $s, e \in T \subset \mathbb{N}$, and Ψ is a subset of *Atoms*, a collection of atomic descriptions of predicates that holds in a time interval. More formally, a semantic trajectory is represented as $s = \{(s_1, e_1, \Psi_1), (s_2, e_2, \Psi_2), ..., (s_n, e_n, \Psi_n)\}$, where Ψ_i is a subset of *Atoms* related to event $i \in [1, n]$ and T is the time domain of the trajectory. This event sequence has a time constraint between its time values; for two consecutive events (s_i, e_i, Ψ_i) and $(s_{i+1}, e_{i+1}, \Psi_{i+1})$ we have that $s_{i+1} = e_i + 1$. In other words, the start time of the last event is the subsequent instant after the end time of the first event.

Our query can be seen as a sequence of predicates that holds in some part of a semantic trajectory, as a regular expression based language isomorphic to the proposed model. In particular, a query is a moving pattern M expressed as a path expression where B is a basic moving pattern, P_1 and P_2 are atomic predicates and c_1 and c_2 are natural numbers:

$$M \leftarrow B \mid (B \sim c_1) \mid B; M \mid (B \sim c_1); M \mid B; [\sim\ c_1]M \mid (B \sim c_1); [\sim\ c_2]M$$
$$B \leftarrow P_1 \mid (P_1\ AND\ P_2) \mid (P_1\ OR\ P_2)$$

A basic moving pattern B can be P_1, $(P_1\ AND\ P_2)$ or $(P_1\ OR\ P_2)$. The atomic predicate P_1 means that $\exists t \in \mathbb{N}$ such that P_1 holds in the instant t. The conjunction $(P_1\ AND\ P_2)$ (resp. disjunction $(P_1\ OR\ P_2)$) means that $\exists t \in \mathbb{N}$ such that P_1 and P_2 (resp. P_1 or P_2) holds in the instant t.

A moving pattern M is defined by the complex constructors B, $(B \sim c_1)$, $B; M$, $(B \sim c_1); M$, $B; [\sim\ c_1]M$ or $(B \sim c_1); [\sim\ c_2]M$. The semantic of the sequence operators vary according to each case. The constructor $(B \sim c_1)$ is a duration constraint of B and it is represented by a relational operator $\sim$ followed by a natural number c_1. The duration of a predicate can be at least equal ($<=$), at least ($<$), at most equal ($>=$) or at most ($>$) to a determined time unit. The semantics of the constructor $(B \sim c_1)$ is defined as $\exists t_1, t_2 \in T$ s.t. $t_1 < t_2$ and B holds in each instant $t \in [t_1, t_2]$ such that $t_2 - t_1 \sim c_1$.

The constructor $B; M$ means that B holds in some instant of the trajectory and M holds in a future instant. Semantically $B; M$ is defined as $\exists t_1 \in T$ s.t. B holds in the instant t_1 and $\exists t_2 \in T$ such that M holds in the instant t_2 and $t_1 < t_2$. On the other hand, the constructor $(B \sim c_1); M$ means that $\exists t_1, t_2 \in T$ s.t. $t_1 < t_2$ and B holds in each instant $t \in [t_1, t_2]$ such that $t_2 - t_1 \sim c_1$ and $\exists t_3 \in T$ such that M holds in the instant t_3 and $t_2 < t_3$. The constructor $B; [\sim\ c_1]M$ (resp. $(B \sim c_1); [\sim\ c_2]M$) is defined similar to $B; M$ (resp. $(B \sim c_1); M$) with the additional constraint $t_2 - t_1 \sim c$ (resp. $t_3 - t_2 \sim c_2$) in the instant times.

As a way to show an application to the proposed language, consider the query Q_1 showed in Section 1, this query can be represented according our proposal as: $(W >= 2); [<= 4]H; D$ (using acronyms previously presented). The expression $(W \geq 2)$ indicates that the trajectory has a *working* activity for at least two time units. The expression $(W >= 2); [<= 4]H$ indicates that the trajectory stay at home (H) after at most four time units from the occurrence of the first predicate $(W >= 2)$. The expression $H; D$ indicates that the trajectory have a dinner (D) after stay at home (H).

4. PROPOSED APPROACH

4.1 Query Evaluation based on Timed Automata

In this paper, we propose the TCTL model checking based on timed automata to evaluate a query wrote in our language. Our approach consists in modeling each semantic trajectory as a timed automaton and translating a query expressed in our language as a TCTL formula. To discover which trajectories satisfy a given query, a model checking will be performed for each timed automaton that represents a given trajectory and the TCTL formula that represents the query. For each model checking process that it answers TRUE, the trajectory related to the timed automaton of the process will be included in the result set. In order to clarify our proposal, see Figure 2. This figure shows that a query wrote in our language is translated to a TCTL formula and that each trajectory from the database is translated to a timed automaton. Then, the trajectories (i.e. the timed automaton representation) which answers true for the model checking are returned.

A trajectory defined by the model of Section 3 will be represented as a timed automaton A with just one clock $x \in X$ where each event $evn = (s, e, \Psi)$ will be a state $q \in Q$ with $Inv(q) = x \leq e$ and $L(q) = \Psi$ (i.e. the predicates holding to the respective event). The first state q_0 will be the first event of the semantic trajectory. The transitions E of the timed automaton will be related to the successor of events in the trajectory. Consider two events $evn_1 = (s_1, e_1, \Psi_1)$ and $evn_2 = (s_2, e_2, \Psi_2)$ such that

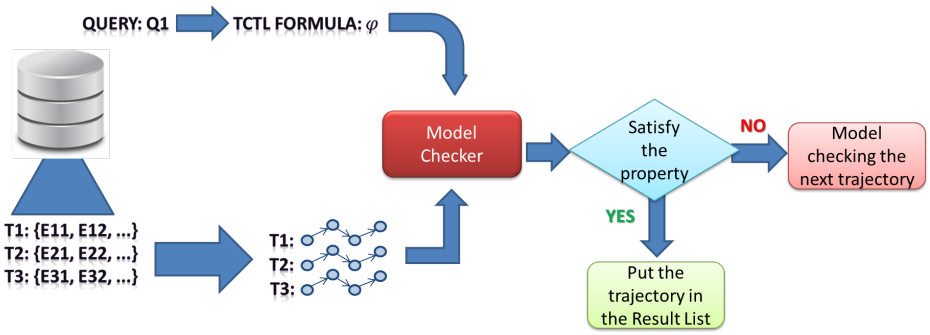


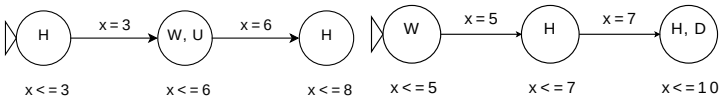
Fig. 2: Overview of the proposed solution.

Query Constructor	TCTL Formula
B	$EF(E(B\ U\ \top))$
$(B_1 \sim c)$	$EF(E(B\ U_{\sim c}\ \top))$
$B; M$	$EF(E(B\ U\ EF(M)))$
$(B \sim c_1); M$	$EF(E(B\ U_{\sim c_1}\ EF(M)))$
$B; [\sim c]M$	$EF(E(B\ U\ EF_{\sim c}(M)))$
$(B \sim c_1); [\sim c_2]M$	$EF(E(B\ U_{\sim c_1}\ EF_{\sim c_2}(M)))$

Table I: Query language constructors and the equivalent TCTL formula.

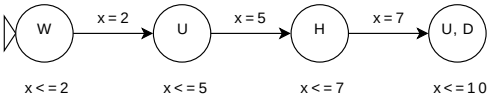
evn_2 is the successor of evn_1. We will build a transition from the state of the first event (i.e. evn_1) q_{env_1} to the state of the second event (i.e. evn_2) q_{evn_2} such that the guard constraint g is $x = e_1$. In a simple way, $q_{env_1} \xrightarrow{g} q_{env_2}$.

As a way to show the timed automaton modeling of the trajectories, consider the timed automata related to the trajectories showed in the Section 1, their automata are showed in the Figure 3.



(a) Trajectory 1

(b) Trajectory 2



(c) Trajectory 3

Fig. 3: Trajectories modeled as a timed automaton

Each constructor of the query language can be interpreted as a TCTL formula as showed in the Table I. Thus, the queries wrote in our language will be translated to TCTL formula aiming performs model checking to evaluate a query. Note that the translation of each query constructor to a TCTL formula is direct and that the expressiveness of our language is limited to TCTL formulae that represent each language constructor.

Complexity Analysis: Considering a Database with m semantic trajectories, our approach performs one model checking process to each trajectory, which result a solution with $O(m \times k)$, where k is the time complexity of the TCTL model checking based on timed automata. As our approach considering timed automata with just one clock variable and a TCTL formulas with constraints $\sim = \{\leq, <, \geq, >\}$, theoretically according to [Laroussinie et al. 2004] the time complexity of this

model checking problem is polynomial, however the practical solutions in model checking were built for generals cases and can have exponential complexity in the worst case. Therefore, our approach is polynomial upper-bound.

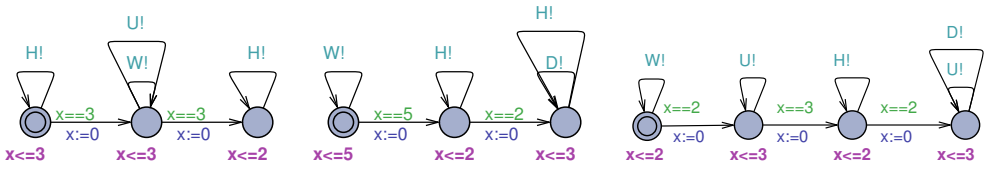
4.2 Experiment in UPPAAL

In this section, we use the UPPAAL model-checking tool-suite to implement the approach proposed in Section 4.1. UPPAAL consists of a graphical user interface and a model checker engine based on the theory of timed automata for modeling, simulation and verification of real-time systems. The tool handles with a small subset of TCTL, allowing the construction of large models by modeling a system as a network of several timed automata in parallel [Behramm et al. 2004]. The model checking in UPPAAL is based on the checking the reachability of a state in a timed automata [Larsen et al. 1997], due this the tool can performs a model checking in PSPACE-complete upper-bound [Bouyer and Laroussinie 2010].

Although the UPPAAL can be cost more than our proposal theoretically, we decide to model our approach in this tool because it is widely used and has many optimizations, resulting from 15 years of active development. In addition, there is no implementation of the model checking algorithm for our specific case. As our approach handle with a finite path performed by a semantic trajectory and use a small set of TCTL logic (i.e. $EF_{\sim c}\varphi$ and $E(\varphi\ U_{\sim c}\ \psi)$ operators), we expect that our approach will be efficient in practice.

In UPPAAL, the transitions can have an action being represented through synchronization channels. Thus, when two automata are synchronized on a channel "a" meaning that a transition labeled with "$a!$" of one automaton occurs simultaneously with a transition "$a?$" of another automaton. A label with "!" represents the component that takes the "initiative", while with "?" represents the "passive" component.

As UPPAAL cannot handle full TCTL, especially nesting of path formulas (i.e. $EF_{\sim c}\varphi$ and $E(\varphi\ U_{\sim c}\ \psi)$), we have to change ours trajectories timed automata and have to model our queries expressed in TCTL formula into a timed automaton. A semantic trajectory will be represented by a timed automaton with just one clock x where each event $evn = (s, e, \Psi)$ will be a state with an invariant $x <= e$. Each predicate $P_1 \in \Psi$ will be a channel in UPPAAL and it is represented in the timed automaton as a self-transition to the correspondent state of the event with a synchronization label $P_1!$. The transition between states following the successor order between events, for two successor events $evn_1 = (s_1, e_1, \Psi_1)$ and $evn_2 = (s_2, e_2, \Psi_2)$ we will build a transition from the state of the first event to the state of the second event such that the guard constraint g is $x = e_1$ and the clock variable x is reset to 0. The UPPAAL timed automata related to the trajectories showed in Section 1 are showed in Figure 4.

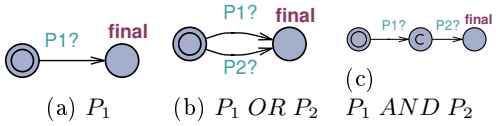


(a) Trajectory 1 (b) Trajectory 2 (c) Trajectory 3

Fig. 4: Trajectory modeling as a timed automata in UPPAAL. The node with double circle means the initial state.

A query in UPPAAL will be a timed automaton where each basic moving pattern B can be translate into three different automata: i) when he have just a predicate P_1, we will build a transition from an initial state to a state labeled *final*; ii) when we have $P_1\ OR\ P_2$, we will build a transition from one state to another for each predicate; iii) when we have $P_1\ AND\ P_2$, we will build a transition

from the initial state to a committed state with a synchronization label $P_1?$ and a transition from the committed state to another state with a synchronization label $P_2?$. Committed states (C) in UPPAAL cannot delay and require the next action to involve a transition whose source state is the committed location, in other words, the incoming transition involves an action to the committed location and the outgoing transition involves an action from the committed location [Behrmann et al. 2006]. The timed automata related to a basic moving pattern are showed in Figure 5



(a) P_1 (b) $P_1\ OR\ P_2$ (c) $P_1\ AND\ P_2$

Fig. 5: Timed automata related to basic moving patterns

For a basic moving pattern with a duration constraint ($B \sim c$) we will build the respective timed automaton for B and will include a guard constraint $x \sim c$ to each transition. For the sequencer operators $B; M$ and $(B \sim c); M$ we will concatenate the automata correspondent to each component of the operator ";". While for the sequencer operators $B; [\sim c]M$ and $(B \sim c_1); [\sim c_2]M$ we will add a new clock t to the query timed automaton. This clock t will be reset in each transition of the automaton related to the first component of operator "$; [\sim c]$" and guard constraint $t \sim c$ will be included in the first transition of the timed automaton related to the second component. For instance, consider the timed automaton related to query Q_1 showed in Figure 6

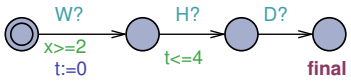


Fig. 6: The query Q_1 as a timed automaton.

To query the system represented in UPPAAL, we construct the following temporal formula that will be, jointly with the automatons, the input of a model checker process: $E <> Q1.final$ (i.e. *exists a trace where the final state is reached in the query automaton*). To query all trajectories we must do the model checker to each trajectory (i.e. for each system: $\{T1, Q_1\}, \{T2, Q_1\}, \{T3, Q_1\}$) and return the name of each trajectory where the system satisfies the temporal formula. Thus, we propose to build a tool that uses UPPAAL for perform this process automatically.

5. RELATED WORK

Aiming to verify the occurrence of certain movements in a trajectory set, it was considered the pattern matching area related with stored data. In [Dindar 2008] is implemented a subset of the SQL extension using a finite state automata. In this work, the pattern matching is performed in databases to catch a sequential pattern described by a regular expression and does not allow to check the occurrence of a predicate set in some time interval. In [Cadonna et al. 2011] they use a similar approach to our work, however they use an automaton-based approach without real-time constraints. In the approach from [Cadonna et al. 2011] the time constraints is not a crucial factor. In addition, the algorithm proposed by this work has exponential complexity instead of polynomial complexity verifiers by our work approach.

Works related with querying trajectories were verified. In [Vieira et al. 2010], the motion "pattern" queries are defined based on regular expressions. To evaluate the query, this work proposes two approaches with polynomial complexity in the worst case. However, this work does not handle semantic trajectories and does not allow expressing duration's thresholds as "at most X hours" or "at least X hours". RE-SPaM [Gomez and Vaisman 2009] is a language, based on regular expressions proposed

with the aim of pruning the candidate data obtained during a mining process. The query evaluation is made through an automaton, however this automaton does not check temporal constraints and has exponential time complexity. ST-DMQL in [Bogorny et al. 2009] is a data mining query language to spatial temporal data that aims to answer many queries about semantic trajectories, however is not showed in this work how this query is evaluated and how hard this process is. In [Sakr and Güting 2011] is defined a novel approach to express and evaluate spatiotemporal pattern queries. These patterns specify temporal order constraints and other temporal constraints between predicates. In this work, the evaluation of the patterns is seen as Constraint Satisfaction Problem (CSP), performed in exponential time.

6. CONCLUSION AND FUTURE WORK

In this paper we have presented the problem of query a semantic trajectory involving temporal constraints. We propose a query language that allows temporal constraints and a TCTL model checking based on timed automata to evaluate this query language. In theoretical view, the approach proposed by this paper is polynomial. In practical view, we propose UPPAAL tool to evaluate our query language. This approach can be PSPACE-complete, however has been optimized for more than 10 years and we expect that UPPAAL tool can be efficient. Finally, we show an example of our solution conducted in UPPAAL model checker tool.

The main **negative aspect** of solution is that our approach consider in memory evaluation, which is infeasible to large semantic trajectories data. While the main **positive aspect** is that the solution is based in a theoretical approach widely studied in the field of Logics that is polynomial upper-bound.

Future work: We are intend to build a tool to query semantic trajectories that uses UPPAAL and we will do some experiments aiming to prove its efficiency. Other works can be done with the objective to improve the expressiveness of the query language and to use this solution in a DBMS.

REFERENCES

Behramm, G., David, A., and Larsen, K. A tutorial on uppaal. *proceedings of the 4th International School on Formal Methods for the Design of Computer*, 2004.

Behrmann, G., David, A., and Larsen, K. A tutorial on uppaal 4.0, 2006.

Bogorny, V., Kuijpers, B., and Alvares, L. St-dmql: A semantic trajectory data mining query language. *International Journal of Geographical Information Science* 23 (10): 1245–1276, 2009.

Bouyer, P. and Laroussinie, F. Model checking timed automata. *Modeling and Verification of Real-Time Systems*, 2010.

Cadonna, B., Gamper, J., and Böhlen, M. H. Sequenced Event Set Pattern Matching Categories and Subject Descriptors. In *Proceedings of the 14th EDBT*. pp. 33—-44, 2011.

Dindar, N. *Pattern Matching over Sequences of Rows in a Relational Database System*. M.S. thesis, ETH Zurich, Switzerland, 2008.

Gomez, L. I. and Vaisman, A. a. Efficient constraint evaluation in categorical sequential pattern mining for trajectory databases. In *12th EDBT '09*. ACM Press, New York, NY, USA, pp. 541–552, 2009.

Huth, M. and Ryan, M. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge Univ Pr, 2004.

Laroussinie, F., Markey, N., and Schnoebelen, P. On model checking durational kripke structures. In *Foundations of Software Science and Computation Structures*. Springer, pp. 291–311, 2002.

Laroussinie, F., Markey, N., and Schnoebelen, P. Model checking timed automata with one or two clocks. *CONCUR 2004-Concurrency Theory*, 2004.

Larsen, K., Pettersson, P., and Yi, W. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)* 1 (1): 134–152, 1997.

Sakr, M. and Güting, R. Spatiotemporal pattern queries. *Geoinformatica* 15 (3): 497–540, 2011.

Vieira, M. R., Bakalov, P., and Tsotras, V. J. Querying Trajectories Using Flexible Patterns. In *Proceedings of the 13th EDBT*. pp. 406–417, 2010.

Yan, Z., Chakraborty, D., Parent, C., and Spaccapietra, S. SeMiTri: A Framework for Semantic Annotation of Heterogeneous Trajectories. In *Proceedings of the 14th EDBT*, 2011.

Yan, Z., Parent, C., Spaccapietra, S., and Chakraborty, D. A Hybrid Model and Computing Platform for Spatio-semantic Trajectories. *The Semantic Web: Research and Applications*, 2010.