First-Order Queries over Temporal Databases Inexpressible in Temporal Logic*

David Toman† and Damian Niwiński‡

Department of Computer Science, University of Toronto
Toronto, Ontario M5S 1A4, Canada

Institute of Informatics, Warsaw University
Banacha 2, 00-950 Warsaw, Poland

Abstract. Queries over temporal databases involve references to time. We study differences between two approaches of including such references into a first-order query language (e.g., relational calculus): explicit (using typed variables and quantifiers) vs. implicit (using a finite set of modal connectives). We also show that though the latter approach—a first-order query language with implicit references to time—is appealing by its simplicity and ease of use, it cannot express all queries expressible using the first one in general. This result also settles a longstanding open problem about the expressive power of first-order temporal logic. A consequence of this result is that there is no first-order complete query language subquery-closed with respect to a uniform database schema, and thus we cannot use temporal relational algebra over uniform relations to evaluate all first-order definable queries.

1 Introduction

In the last several years, various languages for querying temporal databases have been proposed in the literature. The first-order query languages can be divided into two main categories:

1. query languages based on a two-sorted version of relational calculus, e.g., TSQL2 [20] or TQUEL [19], and
2. query languages based on an extension of the relational calculus or algebra by temporal operators, e.g, HRDM's historical relational algebra [6], temporal relational algebra [23], etc.

These two approaches have been often considered equivalent in expressive power, e.g., in [23], where the temporal relational algebra based on an extension of the relational calculus by temporal operators has been proposed as a basis for first-order completeness of temporal query languages. This assumption was based on a well-known property of propositional temporal logic:

---

* Preliminary version of the results has been presented at the *Pre-PODS'95 Workshop on Theory of Constraint Databases*, TR UNL-CSE-95-08.

† Research conducted while this author was at the Dept. of Comp. and Info. Science, Kansas State University, Manhattan, KS 66506, U.S.A., `david@cis.ksu.edu`.

‡ Supported by Polish KBN grant 2 P301 009 06.

Proposition 1.1 (Kamp's Theorem) *Propositional temporal logic has an expressively complete set of temporal connectives over linear orders.*

This means that the propositional temporal logic has the same expressive power as the monadic first-order logic over linear orders. This result has been established by Kamp [15] for complete linear orders, later extended by Stavi [22] for all linear orders, and reproven several times using various proof techniques, e.g, [10, 14]. In this paper we show that this correspondence does *not* generalize to the relationship between the First-order Temporal Logic (FOTL) and the Two-sorted First-order Logic (2-FOL). The query languages based on 2-FOL are strictly more expressive than the languages based on FOTL *for any finite set of temporal connectives* defined over a (dense) linear order. We show that FOTL cannot express a large class of queries that involve references to several time instants, like equality of database relations at two different time instants. The result can be interpreted as

Temporal query languages cannot be simultaneously subquery-closed[1] *and first-order complete.*

Such a result has strong implications for the design of temporal query languages: essentially all subquery-closed relational algebra-like languages are incomplete, while all complete relational calculus-like languages do not preserve closure. This is a problem from the implementation point of view: while the subquery-closed query languages are easier to implement, essentially because all intermediate results in a bottom-up query evaluation have the same structure as database relations, there are first-order queries that cannot be formulated in such languages. Despite this drawback, FOTL was used as a basis of temporal query languages [23] or for specification of temporal integrity constraints [3, 4, 5, 17]. The main reason for this choice is a simpler and more efficient implementation that does not generalize to full FOTL. The design of an expressively complete Temporal Relational Algebra requires an unbounded number of temporal attributes in the relations to guarantee completeness and preserve closure of products [21]. However, such solution does not allow to store the intermediate results needed to evaluate the query in a uniform temporal database as auxiliary relations. In addition to the separation of FOTL from 2-FOL we show several other results, especially:

1. Expressive equivalence of the *future* temporal logic and the *full* temporal logic, i.e., with both future and past temporal connectives [12] does not generalize to the first-order case.
2. Expressive completeness cannot be achieved using more general temporal logic, e.g., the many-dimensional first-order temporal logic [13].

Note also that we restrict our attention solely to the first-order languages. Thus any higher-order properties (e.g., the *even cardinality* property of relations given in [25]) are not expressible in either 2-FOL or FOTL.

[1] Assuming closure with respect to the database schema, where all the temporal relations, including derived ones, have exactly one temporal attribute or, in general, a bounded number of such attributes. This is a natural restriction imposed on virtually all practical temporal databases [21], e.g., on the timestamp relations in TSQL2 or TQUEL.

The rest of the paper is organized as follows: Section 2 introduces the formal definition of the temporal database and first-order temporal query languages over such databases. Section 3 introduces a model-theoretic games that allow us to study the expressive power of FOTL. In Section 4 we use these games to show our main result: FOTL$\subsetneq$2-FOL. Section 5 shows several other non-expressibility results that can be established using games. In the last section of this paper we summarize the obtained results and conclude with the directions of further research.

2 Temporal Structures and Query Languages

In this section we define the precise notion of a temporal database as a temporal structure. We also define two query languages based on extensions of the first-order relational calculus.

2.1 Temporal Databases as Temporal Structures

The standard relational database can be viewed as a model-theoretic structure. For any given database schema the structure contains a finite relation over the database domain for every relational symbol in the database schema, together with the relations defined on the domain itself such as equality (such relations can be thought as *built-in* relations and their interpretation is fixed). It is also well known that relational calculus queries over such database can be thought of as formulas in a first-order language over the same set of symbols.

This model can be naturally extended to a *temporal relational database model* by augmenting every relation symbol in the database schema by an additional attribute that holds the time instant at which the original tuple is true. This extension is formally introduced in [2] and can serve as a unifying platform for various alternative approaches to the definition of temporal databases [7]. Similarly to the standard relational case such temporal databases can be treated as model-theoretic structures, in this case two-sorted.

The temporal structures considered in this section are built from the following three basic building blocks:

1. a first-order signature $\langle \rho_1, \ldots, \rho_n \rangle$ and a first-order structure $\langle T; \rho_1^T, \ldots, \rho_n^T \rangle$ to serve as the *temporal domain.*
2. a first-order signature $\langle p_1, \ldots, p_l \rangle$ and a first-order structure $\langle D; p_1^D, \ldots, p_l^D \rangle$ to serve as the *data domain* of the database.
3. a single-sorted set of predicate symbols $\langle r_1, \ldots, r_m \rangle$; the arity of the symbol r_i is v_i. This choice defines the *database schema* for our temporal database.

This arrangement abstracts from the particular choice of the data and temporal domains and thus allows a wide range of applications of the proposed method.

In the rest of this section we use γ to denote the signature $\langle \rho_1, \ldots, \rho_n \rangle$ of the temporal domain, δ for the signature $\langle p_1, \ldots, p_l \rangle$ of the data domain, and σ for the signature $\langle r_1, \ldots, r_m \rangle$ of the database schema.

Definition 2.1 (Uniform Temporal Database) *Let $\langle T, \rho_1^T, \ldots, \rho_n^T \rangle$ be a temporal domain (signature γ), $\langle D; p_1^D, \ldots, p_l^D \rangle$ a data domain (signature δ), and $\sigma = \langle r_1, \ldots, r_m \rangle$ a database schema. We define R_i to be two-sorted relation symbols of the sort $T \times D^{v_i}$ for each r_i in the database schema σ. We call R_i the* uniform temporal extension of r_i. *A* temporal signature

$$\tau = \langle \rho_1, \ldots, \rho_n, p_1, \ldots, p_l, R_1, \ldots, R_m \rangle$$

is a two-sorted signature composed of γ, δ, and the temporal extensions of all the symbols in σ. We define a temporal database *$\mathcal{A}$ to be a two-sorted τ-structure*

$$\mathcal{A} = \langle T, \rho_1^T, \ldots, \rho_n^T; D, p_1^D, \ldots, p_l^D; R_1^{\mathcal{A}}, \ldots, R_m^{\mathcal{A}} \rangle$$

such that the sets $\{(x_1, \ldots, x_{v_i}) : \mathcal{A} \models R_i(t, x_1, \ldots, x_{v_i})\}$ are finite for every $t \in T$ and $0 < i \leq m$. The extensions $R_i^{\mathcal{A}}$ of R_i in $\mathcal{A}$ define the interpretation of the symbols r_i in the database schema for every element of the time domain, formally:

$$r_i(c_1, \ldots, c_{v_i}) \text{ holds at time } t \text{ iff } \mathcal{A} \models R_i(t, c_1, \ldots, c_{v_i})$$

for $r_i \in \sigma$, $t \in T$, and $c_i \in D$.

Note that the interpretations of the predicate symbols, connected solely with the temporal domain (e.g., $<$) or the data domain (e.g., $=$), is fixed, while the interpretation of the symbols R_i depends on the actual contents of the database.

Example 2.2 Let $\langle Q; < \rangle$ be the temporal domain (dense linearly ordered rationals), $\langle D; = \rangle$ the data domain, and ρ the database schema consisting of a single binary relation $\rho = \langle \text{salary} \rangle$ holding employee IDs and salaries. Then the temporal structure representing such a temporal database is defined as

$$\text{DB} = \langle Q, <^Q; D, =^D; \text{Salary} \rangle$$

where the fact *x has salary y at time t* is encoded by

$$\text{salary}(x, y) \text{ holds at time } t \text{ in DB} \iff \text{DB} \models \text{Salary}(t, x, y)$$

Note that the relations $<^Q$ and $=^D$ have fixed meanings for the given domains, e.g., $<^Q$ is the linear order on Q and $=^D$ is the diagonal on D. On the other hand the interpretation of the relation Salary depends on the actual *contents* of the database. The relation Salary is infinite in general, but the sets $\{(x, y) : Salary(t, x, y)\}$ are finite for every fixed $t \in T$.

2.2 First-order Query Languages

The choice of a first-order query language over such extension is not as straightforward: obviously we can use the two-sorted relational calculus or equivalently the two-sorted first-order logic (2-FOL) to express queries over such structures. The disadvantage of such a solution is that it refers to the *augmented* database schema rather than to the original schema. Moreover, all the references to *time* are explicit—we use variables and quantification over the time domain.

A more elegant solution is to use the first-order temporal logic (FOTL)—the single-sorted first-order logic augmented with a finite set of *temporal connectives* like 'sometime in the past' or 'always in the future'. This solution is preferable for two reasons: first, it refers to the *original* database schema rather than to the extended schema. However, the main advantage of this solution is that the references to time and to all the properties of the underlying temporal domain are *encapsulated* inside the temporal connectives added to the language—all the references to time are *implicit*.

We define two query languages over the uniform temporal databases represented by the τ-structures from Definition 2.1:

1. the two-sorted first-order logic language $\mathcal{M}$ and
2. the first-order temporal logic language $\mathcal{T}$.

The definition of the first language is a straightforward extension of the definition of standard Relational Calculus [24]. Let t_i be variables of sort T and x_i be variables of sort D.

Definition 2.3 *Let γ be the signature of the temporal domain and δ the signature of the data domain. A two-sorted FOL language $\mathcal{M}(\gamma,\delta)$ over the database schema σ is defined by the BNF rule:*

$$\begin{aligned} M ::= & R(t_j, x_1, \ldots, x_{v_i}) \mid \rho(t_1, \ldots, t_{w_k}) \mid \\ & p(x_1, \ldots, x_{u_i}) \mid M \wedge M \mid \neg M \mid \exists x_i.M \mid \exists t_i.M \end{aligned}$$

where R is the temporal extension of r for $r \in \sigma$, $\rho \in \gamma$, and $p \in \delta$.

The semantics of the formulas in this language is the standard first-order Tarskian semantics. Note that the database schema is *monadic* with respect to the sort T, i.e., the predicate symbols in the database schema have always exactly one distinguished argument of sort T. To preserve the *closure* of the query language we restrict the set of the *valid* queries to $\mathcal{M}$-formulas that have exactly one free variable over sort T , so the result of the query can be stored in the temporal database itself. It is easy to see that the closure has been achieved artificially, and that not all $\mathcal{M}$ formulas are valid queries. Moreover, subformulas of a valid query in $\mathcal{M}$ may not be valid queries themselves.

Example 2.4 Consider the formula

$$\exists t_1, t_2. t_1 < t < t_2 \wedge \forall x. P(t_1, x) \iff P(t_2, x).$$

This formula defines the query *"find all the intervals such that P contains exactly the same values on both the ends of each of the intervals"*. Clearly, this is a valid $\mathcal{M}$ query. However, the subformula

$$\forall x. P(t_1, x) \iff P(t_2, x)$$

is not a valid query as it contains two free variables of sort T: t_1 and t_2. Moreover, in Section 4 we show that there is no $\mathcal{M}$ formula with the property that all its subformulas are valid queries and that expresses the original query.

In the definition of the second query language we need to be more careful. The language is defined in two steps: first the temporal connectives are defined. Note that the definition of the temporal connectives is independent of both the data domain and the database schema. The definition of the temporal connectives is purely syntactical and depends only on the chosen temporal domain. The semantics of the connectives is defined by a translation to 2-FOL (cf. Definition 2.10). This approach differs from the usual definition of such connectives given in [10] or [11], that is based on the intended semantics and uses truth tables.

Definition 2.5 (Temporal Connective) *Let γ be the signature of the temporal domain and*

$$O ::= \rho(t_1, \ldots, t_{w_k}) \mid O \wedge O \mid \neg O \mid \exists t_i.O \mid X_i$$

where X_i are predicate variables and $\rho \in \gamma$. A (k-ary) temporal connective *(over γ) is an O-formula with exactly one free variable t_0 and k free predicate variables $X_1, \ldots, X_k$. We denote such connective by $\omega(X_1, \ldots, X_k)$. Let $\Omega(\gamma)$ denote a finite set of (names for) temporal connectives over γ.*

The superscripts t_i denote explicitly the temporal contexts of the appropriate subformulas that the temporal connective glues together.

Definition 2.6 *If every quantification in a temporal connective $\omega \in \Omega(\gamma)$ is of the form $\exists t_i.(\rho(t_0, t_i) \wedge O)$ then ω is a ρ-*restricted temporal connective*.*

Example 2.7 We can express the usual temporal connectives given in [13] in the temporal signature $\langle < \rangle$ of linear orders as follows:

$$\begin{aligned} X_1 \, \mathbf{until} \, X_2 &\triangleq \exists t_2.t_0 < t_2 \wedge X_2 \wedge \forall t_1(t_0 < t_1 < t_2 \rightarrow X_1) \\ \Diamond X_1 &\triangleq \exists t_1.t_0 < t_1 \wedge X_1 \\ \Box X_1 &\triangleq \forall t_1.t_0 < t_1 \rightarrow X_1 \end{aligned}$$

Similarly we can express the past temporal connectives **since**, $\blacklozenge$, and $\blacksquare$. Note that the connectives **until**, $\Diamond$, and $\Box$ are $>$-restricted while the connectives **since**, $\blacklozenge$, and $\blacksquare$ are $<$-restricted. We call the temporal logics that use only the $>$-restricted (the $<$-restricted) temporal connectives the *Future* FOTL (the *Past* FOTL), respectively.

The second step in the definition of the temporal extension to the language of first-order logic is straightforward: we augment the syntax of the FOL over the signatures of the data domain and the database schema by a finite set of temporal connectives:

Definition 2.8 *Let $\Omega(\gamma)$ be a finite set of temporal connectives over γ and δ be the signature of the data domain. A* First-order Temporal Logic Language *$\mathcal{T}(\Omega(\gamma), \delta)$ over the database schema σ is defined by the BNF rule:*

$$F ::= r(x_1, \ldots, x_{v_i}) \mid p(x_1, \ldots, x_{u_i}) \mid \omega(F_1, \ldots, F_k) \mid F \wedge F \mid \neg F \mid \exists x.F$$

for $r \in \sigma$, $p \in \delta$ and $\omega \in \Omega(\gamma)$.

In the rest of the section we use $\mathcal{M}$ and $\mathcal{T}$ in place of $\mathcal{M}(\gamma, \delta)$ and $\mathcal{T}(\Omega(\gamma), \delta)$, respectively.

Definition 2.9 *We say that $\mathcal{T}$ is a* Propositional Temporal Logic (PTL) *if all the relations in the database schema σ are 0-ary.*

It is easy to see that all the $\mathcal{T}$-formulas can be naturally embedded into the language of $\mathcal{M}$-formulas. This embedding also defines the semantics of the $\mathcal{T}$-formulas relatively to the semantics of $\mathcal{M}$:

Definition 2.10 (Embedding of $\mathcal{T}$ into $\mathcal{M}$) *Let* Embed *be a mapping of formulas in the language $\mathcal{T}(\Omega(\gamma), \delta)$ to the language $\mathcal{M}(\gamma, \delta)$ defined as follows:*

$$\begin{aligned} &\mathrm{Embed}(r_i(x_1, \ldots, x_{v_i})) = R_i(t_0, x_1, \ldots, x_{v_i}) \\ &\mathrm{Embed}(p_i(x_1, \ldots, x_{v_i})) = p_i(x_1, \ldots, x_{v_i}) \\ &\mathrm{Embed}(F_1 \wedge F_2) \quad\;\; = \mathrm{Embed}(F_1) \wedge \mathrm{Embed}(F_2) \\ &\mathrm{Embed}(\neg F) \qquad\quad\; = \neg\, \mathrm{Embed}(F) \\ &\mathrm{Embed}(\exists x.F) \qquad\;\; = \exists x.\, \mathrm{Embed}(F) \\ &\mathrm{Embed}(\omega(F_1, \ldots, F_k)) \;\, = \omega^*(\mathrm{Embed}(F_1)[t_0/t_1], \ldots, \mathrm{Embed}(F_k)[t_0/t_k]) \end{aligned}$$

where ω^* is the (O-)formula denoted by ω in $\Omega(\gamma)$ and $F[t_0/t_i]$ is a substitution of t_i for t_0 in F.

In the following development we assume that all the $\mathcal{T}$-formulas are merely restricted $\mathcal{M}$-formulas, i.e., $\mathcal{T} \subseteq \mathcal{M}$. The translation of all $\mathcal{T}$-formulas yields a valid 2-FOL query as $\text{Embed}(\varphi)$ has only one free variable of sort T, namely t_0. Thus, the translation of $\mathcal{T}$-formulas preserves the closure of the query language. Note that the translation of every subformula of the original formula has this property as well. This is obviously not true for subformulas of an arbitrary $\mathcal{M}$-formula even if the original formula is a valid query (cf. Example 2.4).

Example 2.11 Using the temporal database from Example 2.2 we can formulate the following queries in both FOTL and 2-FOL:

personal ID of everyone whose salary was negative

$\blacklozenge \exists y.\text{salary}(x, y) \wedge y < 0$
$\exists t_1.t_1 < t_0 \wedge \exists y.\text{Salary}(t_1, x, y) \wedge y < 0$

personal ID of everyone whose salary has decreased

$\exists y_1, y_2 \blacklozenge(\text{salary}(x, y_1) \wedge \blacklozenge \text{salary}(x, y_2)) \wedge y_1 < y_2$
$\exists y_1, y_2 \exists t_1(t_1 < t_0 \wedge \text{Salary}(t_1, x, y_1) \wedge \exists t_2.t_2 < t_1 \wedge \text{Salary}(t_2, x, y_2)) \wedge y_1 < y_2$

This example also illustrates why the use of FOTL is easier than the use of 2-FOL: in FOTL we do not have to introduce any of the variables t_i—the references to time are encapsulated in the temporal connectives, e.g.,$\blacklozenge$.

It is easy to see the essential difference between the full 2-FOL language $\mathcal{M}$ and the temporal extension to the first-order logic $\mathcal{T}$: each of the subformulas of a $\mathcal{T}$-formula is associated with exactly one temporal context represented by the variable t_0 in the translation to $\mathcal{M}$. Moreover the context can be changed only using a temporal connective. This distinction comes into the play when the temporal connectives, which are merely quantifiers over sort T, and the quantifiers over sort D are interleaved in a single formula.

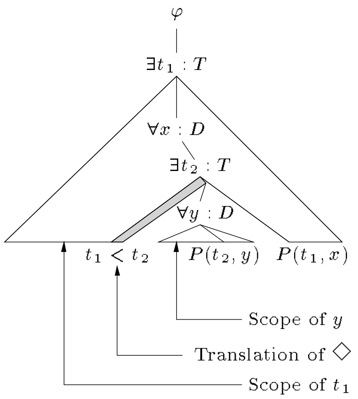
It is quite clear that any two τ-structures distinguishable by a $\mathcal{T}$-formula φ are also distinguishable by a $\mathcal{M}$-formula, namely $\text{Embed}(\varphi)$. However, two τ-structures distinguishable by a $\mathcal{M}$-formula may not be distinguishable by any $\mathcal{T}$-formula. In Section 4 we find concrete examples of structures that are distinguishable by a $\mathcal{M}$-formula but not distinguishable by any $\mathcal{T}$-formula.

3 Ehrenfeucht-Fraïssé Game for Temporal Logic

Our situation is slightly more complicated, as we try to separate FOTL *from* first-order logic, i.e., we try to show that there are 2-FOL formulas that define properties not definable in FOTL.

The rules of the game are modified to match exactly the syntactic restrictions placed on the temporal formulas by Definitions 2.5 and 2.8. The modification captures precisely the expressive power of FOTL. An easy way of thinking about the compatibility of variables is by considering the scopes[2] of variables defined by the expansion of the temporal connectives: The restrictions imposed on the scopes of the individual variable names in the formula $\text{Embed}(\varphi)$ for $\varphi \in \mathcal{T}$ can be depicted as follows:

[2] The parts of the syntactic tree of the formula, where the variable in question can legally appear.



The diagram is a graphical representation of the scopes of the individual variables of the sort T. We can easily see that the scope of the variable t_1 is not the whole subformula rooted by $\exists t_1 : T$ (as it would be in a 2-FOL formula). The scope of t_1 extends only to the point where other temporal connective (e.g., $\Diamond$, represented by the shaded area) is encountered. While t_1 can still be used in the *body* of this connective (i.e., in the expansion of its definition—$t_1 < t_2$), it cannot be used in the subformula rooted by this connective, e.g., in $P(t_2, y)$.

Fig. 1. Restriction on the scopes of the individual temporal variables.

Example 3.1 Figure 1 shows the restrictions on the scopes of the individual variables in the translation Embed(φ) for an FOTL formula φ.

In general the variables x and y can appear together as arguments of an atomic formula only if their scopes *intersect*. Note that this restriction does not have any impact on pairs of variables of sort D. This is very natural as the restrictions are connected solely with the sort T. The following definitions give a precise definition of a combinatorial game, that captures the expressive power of FOTL (similarly to the game for First-order logic in [8, 9]):

Definition 3.2 (Game for Temporal Logic) *Let A and B be two τ-structures, $a = (a_1, \ldots, a_k)$ and $b = (b_1, \ldots, b_k)$ sequences of A-elements and B-elements (respectively), and $n \in N$. A* round of the game $T_n((A, a), (B, b))$ *consists of an ordered sequence of n moves $(a_{k+1}, b_{k+1}), \ldots, (a_{k+n}, b_{k+n})$ of the form "*PLAYER I *chooses an element in the carrier of either A or B and* PLAYER II *chooses an element in the carrier of the other structure". Both the players must obey the following rules:*

1. *the corresponding elements a_i and b_i must be of the same sort, and*
2. *the pair (a_1, b_1) must be of sort T.*

We say that i (i.e., i-th move) is a data move *if both a_i and b_i are of sort D, and that i is a* temporal move *if both elements are of sort T.*

We call the pair (a_1, b_1) the initial temporal context, *and the vectors a and b* the initial assignment.

The vectors $a = (a_1, \ldots, a_k)$ and $b = (b_1, \ldots, b_k)$ are used to handle the open formulas of $\mathcal{M}$ (cf. Theorem 3.10). Except for the additional rules, the game for temporal logic is the same as the Ehrenfeucht-Fraïssé game for first-order logic [18]. The first rule enforces the sort compatibility of the individual moves in the game—this is a natural requirement for the many-sorted structures (we can assume that the relations are empty if the arguments do not match the required sorts). The second rule is needed to define the initial temporal context, the subsequent moves are then relative to this context. Note that a closed temporal

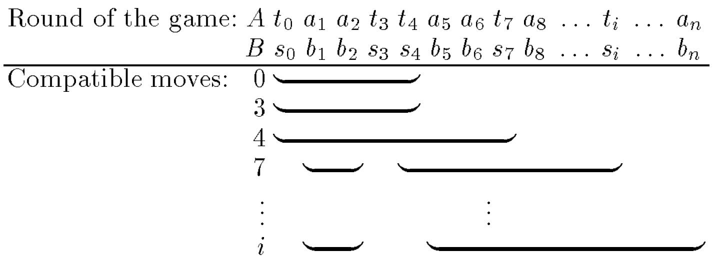


Fig. 2. A round of the game and the scopes of the temporal moves.

logic formula still has one free variable of sort T that represents the initial temporal context (cf. the semantic definition of the temporal logic in section 2 or [13]).

Besides the rules of the game itself we need to define the winning condition for the game for temporal logic. Here we use the observation from Example 3.1:

Definition 3.3 (Compatible Moves) *Let $T_n((A, a), (B, b))$ be a round of the game for temporal logic and i and j two moves in the game, such that i precedes j in T_n ($i < j$). If at least one of:*

1. *i and j are data moves,*
2. *i is a data move and j is a temporal move,*
3. *i is the last temporal move preceding the data move j,*
4. *i and j are both temporal moves, and all the moves between i and j are temporal moves,*
5. *i and j are both temporal moves, and i is the last temporal move preceding j such that there is a data move between i and j*

holds, then we say that the moves i and j are compatible. *A set of moves S is* self-compatible *if every two elements in S are compatible.*

This definition is related to the definition of the temporal connectives and Example 3.1. The blocks of consecutive temporal moves in T_n correspond to a temporal connective or several subsequent temporal connectives with the same number of nested quantifiers over T. The compatibility criterion exactly matches the compatibility of the scopes of the corresponding variables in a formula of $\mathcal{T}$:

Example 3.4 Figure 2 shows a round of the game $T_n((A, t_0), (B, s_0))$ on the structures A and B. The lower part of the figure shows moves compatible with the individual temporal moves in this particular round of the game. It is easy to see that compatibility of moves in the game exactly corresponds to scopes of variables in $\mathcal{T}$-formulas. Note that in the case of 2-FOL, the scopes of the individual variables would extend to the rightmost edge of the figure—the n-th column.

Definition 3.3 implies the following restrictions on the winning condition of the game for temporal logic:

Definition 3.5 *Let* $a = (a_1, \ldots, a_k)$ *for* $a_i \in A$, $b = (b_1, \ldots, b_k)$ *for* $b_i \in B$, *and* $T_n((A, a), (B, b))$ *be a round of the game for FOTL of length* n. *We say that* PLAYER II wins the round of the game for FOTL *if the following conditions hold:*

1. *for any* $p_i \in \delta$ *and any* $k_1, \ldots, k_{u_i}$ *such that* $0 < k_j \leq n + k$ *where all* k_j *are data moves such that* $\{k_1, \ldots, k_{u_i}\}$ *is self-compatible*
$$A \models p_i^A(a_{k_1}, \ldots, a_{k_{u_i}}) \iff B \models p_i^B(b_{k_1}, \ldots, b_{k_{u_i}})$$
2. *for any* $\rho_i \in \gamma$ *and any* $k_1, \ldots, k_{w_i}$ *s.t.* $0 < k_j \leq n + k$ *where all* k_j *are temporal moves such that* $\{k_1, \ldots, k_{w_i}\}$ *is self-compatible*
$$A \models \rho_i^A(a_{k_1}, \ldots, a_{k_{w_i}}) \iff B \models \rho_i^B(b_{k_1}, \ldots, b_{k_{w_i}})$$
3. *for any temporal extension* R_i *of* $r_i \in \sigma$ *and any* $k_0, \ldots, k_{v_i}$ *s.t.* $0 < k_j \leq n+k$ *such that* k_0 *is a temporal move where all* k_j *such that* $0 < j \leq v_i$ *are data moves and* $\{k_0, \ldots, k_{v_i}\}$ *is self-compatible*
$$A \models R_i^A(a_{k_0}, \ldots, a_{k_{v_i}}) \iff B \models R_i^B(b_{k_0}, \ldots, b_{k_{v_i}})$$

Otherwise we say that PLAYER I wins the round of the game for temporal logic. *If* PLAYER II *can always win the game* $T_n((A, a), (B, b))$ *we say that* PLAYER II *has a* winning strategy for $T_n((A, a), (B, b))$.
It is easy to verify that a winning strategy for PLAYER II defines an equivalence relation $\sim_{k,n}$ on the class of τ-structures, where k is the length of the initial assignment and n is the number of moves in the game T_n.

The connection between the games and formulas of FOTL is established as follows:

Definition 3.6 (Quantifier Depth) *We define function* $\mathrm{qd} : \mathcal{M} \to N$

1. *If* φ *is atomic then* $\mathrm{qd}(\varphi) = 0$.
2. *If* φ *is* $\neg\phi$ *then* $\mathrm{qd}(\varphi) = \mathrm{qd}(\phi)$.
3. *If* φ *is* $\phi_1 \wedge \phi_2$ *then* $\mathrm{qd}(\varphi) = \max\{\mathrm{qd}(\phi_1), \mathrm{qd}(\phi_2)\}$.
4. *If* φ *is* $\exists x.\phi$ *or* $\exists t.\phi$ *then* $\mathrm{qd}(\varphi) = \mathrm{qd}(\phi) + 1$.

Definition 3.7 *Let* $\mathcal{L}_{k,n}$ *be set of all formulas* φ *of* $\mathcal{M}$ *such that* φ *is a subformula of* $\mathrm{Embed}(\psi)$ *for some* $\psi \in \mathcal{T}$, $|FV(\varphi| = n$, *and* $\mathrm{qd}(\varphi) = n$.

In general the sets $\mathcal{L}_{k,n}$ are countably infinite, but we can always find a finite subset of this set that "represents" all the formulas in $\mathcal{L}_{k,n}$:

Lemma 3.8 *For each* $k, n \in N$ *there is a finite subset* $\Phi_{k,n}$ *of* $\mathcal{L}_{k,n}$ *such that every formula* $\phi \in \mathcal{L}_{k,n}$ *is equivalent to a formula* $\varphi \in \Phi_{k,n}$. *Moreover, all free variables in* φ *are* $\{v_1, \ldots, v_k\}$.

Notation 3.9 *Let* $\varphi \in \Phi_{k,n}$ *be a formula and* $a = (a_1, \ldots, a_k)$ *a vector of elements from the carrier of a first-order structure* A. *Then we write*
$$A; a \models \varphi \text{ for } A \models \varphi[v_1/a_1, \ldots, v_k/a_k].$$
Note that the variables v_i *are the only free variables of* φ.

Theorem 3.10 *Let A and B be τ-structures, $a = (a_1, \ldots, a_k)$, $b = (b_1, \ldots, b_k)$ sequences of A-elements and B-elements (respectively) such that $a_1 \in T^A$ and $b_1 \in T^B$, and $n \in N$. Then*

$$(A, a) \sim_{k,n} (B, b) \iff \forall \varphi \in \Phi_{k,n} : A; a \models \varphi \iff B; b \models \varphi$$

Note also that if $(A, a) \sim_{1,n} (B, b)$ then the τ-structures A and B are indistinguishable by any closed $\mathcal{T}$-formulas of quantifier depth at most n over an arbitrary set of temporal connectives built using Definition 2.5 for a given initial temporal context a, b.

4 Structures indistinguishable by FOTL formulas

We prepared all the necessary techniques to show our main result: the language $\mathcal{T}$ is strictly less expressive than the language $\mathcal{M}$. We show that for a standard choice of the data domain being the set of uninterpreted constants with equality and a standard choice of a linearly ordered temporal domain:

- the propositional temporal logic has an expressively complete set of temporal connectives [10, 14, 15, 22], but
- there is no finite set of temporal connectives expressively complete in the first-order case.

This result implies that the choice of FOTL with arbitrary finite set of connectives as a basis of a *first-order complete* query language for temporal databases is not adequate, as there are first-order queries not expressible in this language. We prove this result in two steps:

1. First we show that the FOTL language $\mathcal{L}$ is strictly less expressive than the 2-FOL language $\mathcal{M}$ for a degenerate choice of the temporal domain: a set of elements *without* any structure on the elements, even equality (i.e., the signature of the temporal domain is empty) and the standard data domain of uninterpreted constants with equality.
2. In the second step we extend this result to all dense linear orders.

4.1 Time Domain: a Set

The first claim is proven as follows: Let ϵ be an empty signature, $\Diamond_\epsilon X_1 \equiv \exists t_1.X_1$ be a temporal connective, and $\mathcal{T}(\Omega(\epsilon), =)$ be an FOTL language.

Lemma 4.1 *The propositional TL $\mathcal{T}(\{\Diamond_\epsilon\})$ is expressively complete with respect to the monadic 2-FOL $\mathcal{M}(\epsilon)$.*

P r o o f: By induction on the structure of $\varphi \in \mathcal{M}(\epsilon)$.

Note that in the propositional case the signature of the data domain is irrelevant.

Notation 4.2 *Let $\mathcal{S}_n^m = \{S_1, \ldots, S_k\}$ be the set of all n-element subsets of the m-element set $\{1, \ldots, m\}$.*

We prove that Lemma 4.1 does not generalize to the first-order case: the logic $\mathcal{T}(\Omega(\epsilon); =)$ is strictly less expressive than the logic $\mathcal{M}(\epsilon; =)$:

Theorem 4.3 *Let $m, n \in N$ such that $n < m$. We define $P_n^m = \{(t, x) : x \in S_t \in \mathcal{S}_n^m\}$ and $R_n^m = \{(-t, x) : x \in S_t \in \mathcal{S}_n^m\}$. Let*

$$\mathcal{A}_n = \langle Z; \{1, \ldots, 2n\}, =; P_{n-1}^{2n}, R_n^{2n}\rangle \text{ and } \mathcal{B}_n = \langle Z; \{1, \ldots, 2n\}, =; P_n^{2n}, R_n^{2n}\rangle$$

be two temporal structures (with the signature $\langle \epsilon; =; p/1, r/1\rangle$). Then

1. *$\mathcal{A}_n$ and $\mathcal{B}_n$ can be distinguished by a $\mathcal{M}$ formula (of quantifier depth 3), but*
2. *$\mathcal{A}_n$ and $\mathcal{B}_n$ cannot be distinguished by any $\mathcal{T}$ formula of quantifier depth less or equal to $n - 1$.*

P r o o f: (1) Let

$$\varphi \equiv \exists t_1.\exists t_2.((\exists x.P(t_1, x)) \wedge (\exists x.R(t_2, x)) \wedge (\forall x.P(t_1, x) \iff R(t_2, x)))$$

Clearly $\varphi \in \mathcal{M}(\epsilon, =)$, $\varphi \not\models \mathcal{A}_n$, and $\varphi \models \mathcal{B}_n$. To prove (2) we define a winning strategy for Player II: Let $(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)$ be all the data moves and (t, s) the last temporal move in a prefix of a round of the game T_n that satisfies the winning condition. We show that every sequence of moves, shorter than n can be extended by one move. By case analysis (assuming Player I picks an element from the carrier of the structure $\mathcal{A}_n$):

1. Player I plays a data move $a \in \{1, \ldots, 2n\}$ and $a = a_i$ for some $0 < i \leq k$. Then Player II responds by $b = b_i$.
2. Player I plays a data move $a \in \{1, \ldots, 2n\}$ and $a \neq a_i$ for all $0 < i \leq k$. Then Player II plays $b \in \{1, \ldots, 2n\}$ such that $b \neq b_i$ for $0 < i \leq k$ and if $(t, a) \in P_{n-1}^{2n}$ then $(s, b) \in P_n^{2n}$, otherwise $(s, b) \notin P_n^{2n}$. Similarly, if $(t, a) \in R_n^{2n}$ then $(s, b) \in R_n^{2n}$, otherwise $(s, b) \notin P_n^{2n}$. In all cases Player II can find such a b as there are only $k < n$ elements chosen so far.
3. Player I plays a temporal move $t' \in \{1, \ldots, |\mathcal{S}_{n-1}^{2n}|\}$. Then Player II plays $s' \in \{1, \ldots, |\mathcal{S}_n^{2n}|\}$ such that $(t', a_i) \in P_{n-1}^{2n}$ if and only if $(s', b_i) \in P_n^{2n}$ for all $0 < i \leq k$. Again, such a s' exists because $\mathcal{S}_n^{2n}$ contains all n-element subsets of $\{1, \ldots, 2n\}$ and the number of data moves played so far is less than n. The choice may not be unique—in that case Player II can choose arbitrarily as all the elements of T are indistinguishable in the signature ϵ.
4. Player I plays a temporal move $t' \in \{-1, \ldots, -|\mathcal{S}_n^{2n}|\}$. Then Player II plays $s' \in \{-1, \ldots, -|\mathcal{S}_n^{2n}|\}$ such that $(t', a_i) \in R_n^{2n}$ if and only if $(s', b_i) \in R_n^{2n}$ for all $0 < i \leq k$. Similarly to the previous case, such a s' always exists.

Similarly, if Player I chooses an element from the carrier of the structure $\mathcal{B}_n$, Player II replies with an element from $\mathcal{A}_n$ using symmetric strategy. In all cases the new move does not violate the winning condition. The conclusion of this theorem follows from Theorem 3.10.

Corollary 4.4 *FOTL $\mathcal{T}(\Omega(\epsilon), =)$ is strictly less expressive than 2-FOL $\mathcal{M}(\epsilon, =)$ for an arbitrary finite set of connectives $\Omega(\epsilon)$.*

Theorem 4.3 and Corollary 4.4 also provide an example of when expressive completeness in the propositional case does not imply the existence of a complete set of connectives in the first-order case.

4.2 Time Domain: Dense Linear Order

To show a similar result for temporal logic over linear orders is more complicated. We need to account for the structure of the temporal domain. Consider the following definition:

Definition 4.5 *Let $(Q, <)$ be a linear order, S a finite set, $I \subseteq Q$ a non-trivial interval, and $f : I \to S$ a function. We say that f is* dense over S *if for every $s, t \in I$ such that $s < t$ and for every $x \in S$ there is $r \in I$ such that $s < r < t$ and $x = f(r)$.*

Clearly, for any dense linear order we can define functions dense over any finite codomain.

Lemma 4.6 *Let $(Q, <)$ be a dense linear order and $I \subseteq Q$ an open interval such that $|I| > 1$. Then for every finite set S there is a $f : I \longrightarrow S$ that is dense over S.*

Theorem 4.7 *Let $(Q, <)$ be a dense linear order, $f^I_{m,n} : I \longrightarrow \mathcal{S}^m_n \cup \{\emptyset\}$ dense over $\mathcal{S}^m_n \cup \{\emptyset\}$ for every non-trivial interval $I \subset Q$, and let J_1 and J_2 be two nontrivial, open, and disjoint intervals in Q. We define $P^m_n = \{(t, x) : x \in f^{J_1}_{m,n}(t)\}$ and $R^m_n = \{(t, x) : x \in f^{J_2}_{m,n}(t)\}$. Let*

$$\mathcal{A}_n = \langle Q, <; \{1, \ldots, 2n\}, =; P^{2n}_{n-1}, R^{2n}_n\rangle \text{ and } \mathcal{B}_n = \langle Q, <; \{1, \ldots, 2n\}, =; P^{2n}_n, R^{2n}_n\rangle$$

be two temporal databases. Then

1. *$\mathcal{A}_n$ and $\mathcal{B}_n$ can be distinguished by a $\mathcal{M}$ formula (of quantifier depth 3), but*
2. *$\mathcal{A}_n$ and $\mathcal{B}_n$ cannot be distinguished by any $\mathcal{T}$ formula of quantifier depth less than or equal to $n - 1$.*

Corollary 4.8 *FOTL $\mathcal{T}(\Omega(<), =)$ over a dense linear order $<$ is strictly less expressive than 2-FOL $\mathcal{M}(<, =)$ for an arbitrary finite set of connectives $\Omega(<)$.*

5 Additional Results

Using our technique we can prove several additional results about FOTL. Among other results, we show that the gap between FOTL and 2-FOL cannot be bridged using a more powerful version of FOTL—the weakness is inherent to the restriction on the maximal temporal arity (dimension) of the logic FOTL.

5.1 Separation of Future FOTL from FOTL

Using a similar technique as in Theorem 4.7 we can separate the *Future* FOTL with arbitrary future temporal connectives from the *full* FOTL $\mathcal{T}(\mathbf{since}, \mathbf{until})$. These two logics are expressively equivalent in the propositional case [12].

Lemma 5.1 *Let $(Q^+; <, 0)$ be a dense linear order with a left endpoint (0). Let $J \subset Q^+$ be a non-trivial open interval and $f^I_{m,n} : I \longrightarrow \mathcal{S}^m_n \cup \{\emptyset\}$ be dense over $\mathcal{S}^m_n$. We define $P^m_n = \{(t, x) : x \in f^J_{m,n}(t)\}$ and $R_n = \{(0, x) : 0 < x \leq n\}$. Let*

$$\mathcal{A}_n = \langle Q, <; \{1, \ldots, 2n\}, =; P^{2n}_{n-1}, R_n\rangle \text{ and } \mathcal{B}_n = \langle Q, <; \{1, \ldots, 2n\}, =; P^{2n}_n, R_n\rangle$$

be two temporal structures. Then

1. *The structures $\mathcal{A}_n$ and $\mathcal{B}_n$ can be distinguished by a $\mathcal{T}(\mathbf{since}, \mathbf{until})$ formula of quantifier depth 3, but*
2. *$\mathcal{A}_n$ and $\mathcal{B}_n$ cannot be distinguished by any future $\mathcal{T}$ formula of quantifier depth less or equal to $n-1$.*

Proof: (1) Let $\varphi = (\Diamond\forall x.(p(x) \iff \blacklozenge r(x))) \in \mathcal{T}(\mathbf{since}, \mathbf{until})$. Clearly, $\varphi, 0 \not\models \mathcal{A}_n$ and $\varphi, 0 \models \mathcal{B}_n$.

(2) follows from observation that $f^I_{m,n}$ is dense over $\mathcal{S}^m_n$ (and thus PLAYER II can find an appropriate answer to every temporal move of the PLAYER I), and the fact, that the quantifiers in future FOTL formulas are restricted (this means that PLAYER I cannot return to *time* 0 once he played any other temporal move, as the sequence of temporal moves in a game for Future FOTL has to be non-decreasing sequence of elements of T).

5.2 Expressive Incompleteness for Many-dimensional FOTL

When using FOTL as a query language, we cannot express queries that reference two distinct time instants (in an essential way, cf. Section 4). This is due to the limitation on the number of temporal contexts passed to the individual subformulas of a FOTL formula. In [13] many-dimensional propositional temporal logics are studied. These logics can be easily modified to many-dimensional first-order temporal logics (k-FOTL, where k is the dimension of the logic). The main idea behind this approach is to allow k *temporal contexts* in all the temporal (sub-)formulas. Clearly, 2-FOTL can express the query from Theorem 4.3 (2) as exactly two temporal contexts are needed in this query. However, using our technique we can show that k-FOTL is strictly less expressive than $(k+1)$-FOTL and thus is also less expressive than 2-FOL for any $k > 1$.

Definition 5.2 (Many-dimensional Temporal Connective) *Let γ be a signature of the temporal domain, $k > 0$, and t_i be variables ranging over k-tuples of elements of T (time instants). Then*

$$O ::= \rho(t_1[j_1], \ldots, t_{w_k}[j_k]) \mid O \wedge O \mid \neg O \mid \exists t_i.O \mid X_i$$

where X_i are predicate variables, $\rho \in \gamma$, and $t_i[j]$ is the j-th component of the tuple variable t_i. A (n-ary) k-dimensional temporal connective (over γ) is an O-formula with exactly one free variable t_0 and n free predicate variables $X_1, \ldots, X_n$. We denote it by $\omega(X_1, \ldots, X_n)$ Let $\Omega(\gamma)$ denote a finite set of (names for) temporal connectives over γ.

Now we can define the language of k-FOTL exactly the same way as in the case of FOTL (cf. Definition 2.8). The semantics is also defined in the same way as in Definition 2.10 with a small “fix” to the base case:

$$\mathrm{Embed}(r_i(x_1, \ldots, x_{v_i})) = R_i(t_0[1], x_1, \ldots, x_{v_i}).$$

The game for k-dimensional temporal logic is similar to the game for FOTL. The only difference is that for every temporal move we always pick k-tuples of elements of sort T instead of a single element. We have to guarantee that the winning condition is met for all *components* of every tuple move.

Note that we are still using the uniform temporal databases with relations monadic in the sort T. Clearly, a higher dimension of the base relations in the temporal sort leads to the separation result immediately. However, even with the restriction on the database relations we can prove the separation of the k-FOTL from $(k+1)$-FOTL.

Theorem 5.3 *Let*

$$\varphi := \exists t.\exists t_1,\ldots,t_k.\forall x(R(t,x) \iff \bigvee_{0<i\leq k} P(t_i,x))$$

Then φ is expressible in $(k+1)$-FOTL but not in k-FOTL.

P r o o f: That φ is not expressible by k-FOTL follows from a modification of Theorems 4.3 or 4.7. To show that φ is expressible in $(k+1)$-FOTL, consider following $(k+1)$-dimensional connectives:

$$\begin{aligned} \Diamond &= \exists t_1.X_1 \\ \uparrow\downarrow_i &= \exists t_1.(t_1[1] = t_0[i] \wedge X_1) \qquad \text{for } 0 < i \leq k \end{aligned}$$

Note that the variables t_i range over $(k+1)$-tuples of temporal elements. Then

$$\varphi \text{ is equivalent to } \Diamond\forall x(r(x) \iff \bigvee_{0<i\leq k} \uparrow\downarrow_i p(x))$$

which is a $(k+1)$-FOTL formula.

Thus the many-dimensional FOTL logics form a proper hierarchy with respect to their relative expressive power:

$$\text{FOTL} \subsetneq \text{2-FOTL} \subsetneq \ldots \subsetneq (k-1)\text{-FOTL} \subsetneq k\text{-FOTL} \subsetneq \ldots \subsetneq \text{2-FOL}$$

Note that for an arbitrary finite set of connectives we can find a k-FOTL that can express all the connectives for any $k \in N$. Also, from the previous results and [16], we know that

$$\text{Future-FOTL} \subsetneq \text{FOTL} \subsetneq \text{FOTL(now)} \subseteq \text{2-FOTL}$$

On the other hand, the hierarchy of k-FOTL *approximates* 2-FOL:

Theorem 5.4 *Let ψ be a 2-FOL formula. Then there is a natural number k and a k-FOTL formula φ such that $\psi \equiv \varphi$.*

5.3 Temporal Relational Algebras

Previous results also show that every temporal relational algebra is strictly weaker the temporal relational calculus:

Theorem 5.5 *Let Γ be a first-order definable relational algebra over uniform relational types. Then the relational algebra queries over Γ cannot express all first-order queries.*

5.4 Ordered Data Domain

Theorems 4.3 and 4.7 hold even in the presence of the linear order on the data domain (which is a usual assumption in relational databases). The only difference is in the choice of the cardinality of the subsets of elements that are in the relations P and R. The subsets are chosen similarly to proof of Theorem 4.7. We choose the data domain to be $\{1,\ldots,2^{n+1}\}$ and define the τ-structures by

$$\mathcal{A}_n = \langle Q, <; D, =; P^{2^{n+1}}_{2^n-1}, R^{2^{n+1}}_{2^n}\rangle \quad \text{and} \quad \mathcal{B}_n = \langle Q, <; D, =; P^{2^{n+1}}_{2^n}, R^{2^{n+1}}_{2^n}\rangle$$

The strategy for Player II is essentially the same, as two linear orders with cardinalities 2^n and $2^n - 1$ cannot be distinguished by n data moves, and Player I cannot pick more than $n-1$ data moves in a round of the game for temporal logic of length n.

5.5 Fixed Data Domain

Corollary 4.8 may suggest that the logic FOTL is always strictly less expressive than 2-FOL in the first-order case. This is not true either:

Example 5.6 Consider the situation where D is a finite set of fixed size. In this case the proof of Lemma 4.7 will not work as we cannot build the structures $\mathcal{A}_n$ and $\mathcal{B}_n$ of arbitrary size. If, moreover, every element of D is denotable by a constant we can replace quantifiers over D as follows:

$$\forall x.\varphi = \bigwedge_{c \in D} \varphi[x/c] \qquad \exists x.\varphi = \bigvee_{c \in D} \varphi[x/c]$$

It is easy to see that this case is essentially the same as the case of propositional temporal logic. Note that the fixed size of the domain is a first-order property (i.e., it can be defined by a first-order theory of D). However, the size of the resulting formula depends on the size of D.

6 Conclusion

In this paper we have defined a model-theoretical game that captures the expressive power of k-FOTL in a very general setting—the definition is independent of the underlying signatures and theories. We have used this game to resolve several open questions about temporal logics. Our main result is the separation of FOTL from 2-FOL: 2-FOL is strictly more expressive than *any* variant of temporal logic over dense orders. Thus k-FOTL cannot be used as a basis for the design of a first-order complete temporal query language. We conjecture that the same results hold for the expressive power over all (sufficiently large) linear orders—the proof of Theorem 4.7 for discrete orders needs to be modified using a *restricted* density property that allows one to maintain the winning strategy for Player II, but makes the order isomorphic to the standard linear order on integers. We can prove Theorem 4.7 for FOTL with unary temporal connectives over discrete linear orders. However, substantiating this claim in its full generality is the goal of ongoing research.

Future research in this area will concentrate on the following topics:

1. In the case of complete first-order query languages, new and more general implementation techniques need to be developed for handling the temporal information. The intermediate results of the query evaluation algorithms may be more complicated than allowed by the uniform database schema. Thus, such intermediate results cannot be stored in a uniform temporal database itself as auxiliary relations.

2. In the area of incomplete but subquery-closed temporal query languages the relationships between the different sets of temporal connectives need to be investigated. For many applications a limited number of simultaneous temporal contexts may be sufficient. This would allow the use of a sufficiently large set of temporal connectives tailored for the specific application.

3. The temporal databases are infinite structures in general. However, all the data stored in the database must be finitely representable. This restriction disallows the use of arbitrarily complex temporal databases (cf. Definition 2.1)[3]. Sufficient restriction on the class of allowed temporal databases may be sufficient to guarantee both the closure and completeness. We have seen that bounded number of elements in the data domain is such a restriction. Are there other (nontrivial) restrictions (especially on the temporal domain) that also guarantee both the closure and completeness?

7 Related Work

The separation is proven using a modification of the Ehrenfeucht-Fraïssé Games to capture the properties of Temporal Logic. In [14] pebble games have been used to show expressivity results for the *monadic logic* over linear orders. However, our results and techniques are different, as we are interested in the *first-order* temporal logics (and the corresponding 2-FOL). The method introduced in [14] is no longer sufficient as it cannot handle unrestricted quantification over the data sort. In [16] a restricted version of 2-FOTL was presented introducing the *now* connective that allows one to reset the temporal context of a subformula to the original evaluation point. This logic was also shown to be strictly stronger than FOTL. However, the technique used in [16] does not apply to temporal databases—the proof of the fact is carried out over first-order structures that cannot be finitely encoded as the contents of the *database relation(s)* is an infinite and coinfinite[4] *at every time instant*. Such sets cannot be represented over the data domain (as the theory of equality can finitely encode only finite and cofinite sets. A recent result [1] that separates FOTL with the **since** and **until** connectives from 2-FOL in the case of finite (sufficiently large) linear orders supports our conjecture. However [1] uses a counting argument in the proof and thus it does not generalize to arbitrary linear orders. Also, the relationship between this work and various sets of temporal connectives defined in [13] is not clear.

References

1. Abiteboul, S., Herr, L., Van den Bussche, J. Temporal Connectives versus Explicit Timestamps in Temporal Query Languages. (unpublished manuscript).
2. Chomicki J. Temporal Query Languages: a Survey. Proc. *International Conference on Temporal Logic*, July 1994, Bonn, Germany, Springer-Verlag (LNAI 827), pp. 506–534.
3. Chomicki J. Efficient Checking of Temporal Integrity Constraints Using Bounded History Encoding. In *ACM Transactions on Database Systems*, (20) 2, pp. 149–186. 1995.
4. Chomicki, J.,N iwinski, D. On the Feasibility of Checking Temporal Integrity Constraints. Proc. *12th ACM Symposium on Principles of Database Systems,* pp 202–213, 1993. (full version to appear in JCSS).

[3] Note that all the databases used in our proofs are finitely representable.

[4] I.e., its complement is infinite.

5. Chomicki, J., Toman, D. Implementing Temporal Integrity Constraints Using an Active DBMS. *IEEE Transactions on Knowledge and Data Engineering*, Special section on Temporal and Real-time Databases, Vol. 7, No. 4, 1995.
6. Clifford J., Croker A. The Historical Relational Data Model (HRDM) and Algebra based on Lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, Los Angeles, CA, February 1987.
7. Clifford J., Croker A., Tuzhilin A. On Completeness of Historical Relational Query Languages. *ACM Transactions on Database Systems*, Vol. 19, No. 1, pp. 64–116, 1994.
8. Ehrenfeucht, A. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.
9. Fraisse, R. Sur les classifications des systemes de relations. *Publ. Sci. Univ. Alger*, 1:1, 1954.
10. Gabbay D. Expressive Functional Completeness in Tense Logic. In Mönnich U. *Aspects of Philosophical Logic*, 91–117, 1981.
11. Gabbay D. The Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In Banieqbal B., et al. (ed.) *Temporal Logic in Specification*, vol. 398, pp.409–448, Springer Verlag, LNCS 398, 1989.
12. Gabbay D., Pnueli A., Shelah S., Stavi J. On the Temporal Analysis of Fairness. Proc. *ACM Symposium on Principles of Programming Languages*, 1980.
13. Gabbay D., Hodkinson I., Reynolds M. Temporal Logic. Mathematical Foundations and Computational Aspects. Vol. 1. *Oxford Logic Guides* 28, Oxford Science Publications, 1994.
14. Immerman N., Kozen D. Definability with Bounded Number of Variables. Information and Computation 83, pp.121–139, 1989.
15. Kamp J.A.W. Tense Logic and the Theory of Linear Order. PhD thesis, University of California, Los Angeles, 1968.
16. Kamp J.A.W. On the Formal Properties of 'now'. *Theoria* 37:227–273, 1971.
17. Lipeck U.W., Saake, G. Monitoring Dynamic Integrity Constraints Based on Temporal Logic. *Information Systems*, 12(2):255–269, 1987.
18. Rosenstein J.G. Linear Orderings. Academic Press, New York, 1982.
19. Snodgrass R. T. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.
20. Snodgrass R.T., editor. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 674+xxiv pages, 1995.
21. Tansel A., Clifford J., Gadia S., Jajodia S., Segev A., Snodgrass R. Temporal Databases. Theory, Design, and Implementation. Benjamin Cummings 1993.
22. Stavi J. Functional Completeness over Rationals. Unpublished, Bar-Ilan University, Ramat-Gan, Israel, 1979.
23. Tuzhilin A., Clifford J. A Temporal Relational Algebra as a Basis for Temporal Completeness. Proc. *International Conference on VLDB*, 1990.
24. Ullman J. D. Principles of Database and Knowledge-base Systems, Vol. 1,2. Computer Science Systems, 1989.
25. Vardi M.Y. A Temporal Fixpoint Calculus. In *ACM Symposium on Principles of Programming Languages*, 1988.

This article was processed using the LaTeX macro package with LLNCS style