Query temporal databases First-Oder Logic vs Temporal logic

Dariusz Surowik

Department of Logic, Informatics and Philosophy of Science

Białystok University

Technologies of Knowledge Exploration and Representation

Hołny-Mejera 05-09 September 2007

Theorem (Kamp,1968)

Propositional TL(Since, Until) is expressively equivalent to monadic FOL over complete linear order.

Some theorems

Theorem (Kamp,1968)

Propositional TL(Since, Until) is expressively equivalent to monadic FOL over complete linear order.

Theorem

TL and TS-FO have the same expressive power in the propositional case.

Definition

A database schema S is a finite set of relation names, where each relation name has an associated arity.

Definition

A database schema S is a finite set of relation names, where each relation name has an associated arity.

Definition

A temporal database over a database schemata S is a non-empty finite sequence $I = I_1, I_2, I_3, ..., I_n$ $(n >= 1)$ of instances of S. Every $j \epsilon \{1, ..., n\}$ is called a state of I.

Definition

A database schema S is a finite set of relation names, where each relation name has an associated arity.

Definition

A temporal database over a database schemata S is a non-empty finite sequence $I = I_1, I_2, I_3, ..., I_n$ $(n >= 1)$ of instances of S. Every $j \epsilon \{1, ..., n\}$ is called a state of I.

Definition

Temporal database I with a two-sorted relational structure is called the timestamp representation of I.

The language of $TS - FO$

TS-FO

$TS - FO$ - a query language of first-order logic used on the timestamp representation of a temporal database.

TS-FO

$TS - FO$ - a query language of first-order logic used on the timestamp representation of a temporal database.

EXAMPLE

Let S be relation such that $S(x, t)$ is interpreted as x is a patient of a hospital in time t.

The formula

$$(\exists t_1)\,(\exists t_2)\,(\exists t_3)\,(t_1 < t_2 < t_3 \wedge S(x, t_1) \wedge \sim S(x, t_2) \wedge S(x, t_3))$$

is interpreted as follow: **x was a patient of a hospital at least twice.**

The syntax of ETL

The syntax of ETL is over some database schema S is obtained by using the formation rules for standing first-order logic over S together with additional formation rule:

R: Let L be are regular language over the finite alphabet $(v_1, v_2, ..., v_p)$, let $(\varphi_1, \varphi_2, ..., \varphi_p)$ be formulas.

Then

$$L^+(\varphi_1, \varphi_2, ..., \varphi_p)$$

and

$$L^-(\varphi_1, \varphi_2, ..., \varphi_p)$$

are formulas too.

The semantics of ETL

Let $I = I_1, I_2, I_3, ..., I_n$ $(n >= 1)$ be a temporal database over schema S. Let $\varphi[\overline{x}]$ be formula of ETL with free variables $\overline{x} = (x_1, x_2, ..., x_k)$. Let $\overline{a} = (a_1, a_2, ..., a_k)$ be data elements in the active domain I, and let $j \epsilon \{1, ..., n\}$ be a state. The truth of $\varphi[\overline{a}]$ in database I in state j $(I, j \models \varphi[\overline{a}])$ is defined as follows:

The semantics of ETL

Let $I = I_1, I_2, I_3, ..., I_n$ $(n >= 1)$ be a temporal database over schema S. Let $\varphi[\overline{x}]$ be formula of ETL with free variables $\overline{x} = (x_1, x_2, ..., x_k)$. Let $\overline{a} = (a_1, a_2, ..., a_k)$ be data elements in the active domain I, and let $j \epsilon \{1, ..., n\}$ be a state. The truth of $\varphi[\overline{a}]$ in database I in state j $(I, j \models \varphi[\overline{a}])$ is defined as follows:

- If φ is atomic formula, or φ is a one of a form: $\sim \phi, \phi \wedge \psi, \exists x \phi(x), \forall x \phi(x)$, then definition is as usual.

The semantics ETL...

The semantics ETL...

- If ϕ is of the form $L^+(\varphi_1, \varphi_2, ..., \varphi_p)$, where L is a regular over the alphabet $(v_1, v_2, ..., v_p)$, then

  $I, j \models \varphi[\bar{a}]$, if there exists a word $w = v_{w_j} ... v_{w_n}$ of length $(n - j + 1)$ in L such that:

  $I, j \models \varphi_{w_j}[\bar{a}]$ and $I, j + 1 \models \varphi_{w_{j+1}}[\bar{a}]$ and ... and $I, n \models \varphi_{w_n}[\bar{a}]$.

The semantics ETL...

- If ϕ is of the form $L^+(\varphi_1, \varphi_2, ..., \varphi_p)$, where L is a regular over the alphabet $(v_1, v_2, ..., v_p)$, then

  $I, j \models \varphi[\bar{a}]$, if there exists a word $w = v_{w_j} ... v_{w_n}$ of length $(n - j + 1)$ in L such that:

  $I, j \models \varphi_{w_j}[\bar{a}]$ and $I, j+1 \models \varphi_{w_{j+1}}[\bar{a}]$ and ... and $I, n \models \varphi_{w_n}[\bar{a}]$.

- If ϕ is of the form $L^-(\varphi_1, \varphi_2, ..., \varphi_p)$, then

  $I, j \models \varphi[\bar{a}]$, if there exists a word $w = v_{w_j} ... v_{w_1}$ of length j in L such that :

  $I, j \models \varphi_{w_j}[\bar{a}]$ and $I, j-1 \models \varphi_{w_{j-1}}[\bar{a}]$ and ... and $I, 1 \models \varphi_{w_1}[\bar{a}]$.

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

n=10, j=3

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

n=10, j=3

- aaaaabaa

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

n=10, j=3

- aaaaabaa
- aabaaaaa

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

n=10, j=3

- aaaaabaa
- aabaaaaa
- aaaabaaa

Examples

Example 1

The formula $L_1^+(true, \varphi)$ of ETL, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in time j iff there is some time in the future of j (including j itself) where φ is true.

n=10, j=3

- aaaaabaa
- aabaaaaa
- aaaabaaa

Example 2

The formula $L_1^-(true, \varphi)$, where L_1 is a language a^*ba^* over the alphabet (a, b) is true in j iff there is some time in the past of j (including j itself) where φ is true.

Example 3

x was a patient of a hospital at least twice.

$$\sim S(x) \wedge L_1^-(S(x)) \wedge L_1^+(S(x))$$

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Until,n=10, j=3

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Until,n=10, j=3

- aaaaabcc

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Until,n=10, j=3

- aaaaabcc
- aabccccc

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Until,n=10, j=3

- aaaaabcc
- aabccccc
- aaaabccc

Since i Until

φ **since** $\psi \equiv L_4^-(\phi, \psi, true)$
φ **until** $\psi \equiv L_4^+(\phi, \psi, true)$,

where L_4 is the language a^*bc^* over the alphabet (a, b, c)

Until,n=10, j=3

- aaaaabcc
- aabccccc
- aaaabccc

Next i Previous

next $\varphi \equiv L_5^+(true, \phi)$
previous $\varphi \equiv L_5^-(true, \phi)$

where L_5 is the language aba^* over the alphabet (a, b)

Boolean query

Boolean queries in TBD

Boolean query

- A Boolean query on a class of temporal databases over some fixed schema is a mapping assigning true or false to each database in the class.

Boolean queries in TBD

Boolean query

- A Boolean query on a class of temporal databases over some fixed schema is a mapping assigning true or false to each database in the class.
- Every TS-FO sentence defines a Boolean query in the obvious way.

Boolean queries in TBD

Boolean query

- A Boolean query on a class of temporal databases over some fixed schema is a mapping assigning true or false to each database in the class.
- Every TS-FO sentence defines a Boolean query in the obvious way.
- Every ETL sentence φ defines a Boolean query Q as follows: For a temporal database I, $Q(I) = \mathit{true}$ iff φ is true in I at every state.

Expressive power - TS-FO vs ETL

Lemma 1

There are some queries expressible in ETL and not expressible in TS-FO.

Expressive power - TS-FO vs ETL

Lemma 1

There are some queries expressible in ETL and not expressible in TS-FO.

Example

"The length of the temporal database is even" is expressible in ETL ($L_3^+(true)$, where L_3 is a language $(aa)^*$) and it is not expressible in TS-FO (since parity of a linear order is well-known not to be first-order definable).

Definition

Let P be a binary predicate on sets of sets of data elements. We say that P has constant communication complexity if there exist natural numbers k and r and communication protocol between two parties (denoted by A and B) that, for each finite set D of data elements, can evaluate $P(X, Y)$ on any sets X and Y of non-empty subsets od D as follows:

Definition

Let P be a binary predicate on sets of sets of data elements. We say that P has constant communication complexity if there exist natural numbers k and r and communication protocol between two parties (denoted by A and B) that, for each finite set D of data elements, can evaluate $P(X, Y)$ on any sets X and Y of non-empty subsets od D as follows:

1. A gets X and B gets Y. Both parties know D

Definition

Let P be a binary predicate on sets of sets of data elements. We say that P has constant communication complexity if there exist natural numbers k and r and communication protocol between two parties (denoted by A and B) that, for each finite set D of data elements, can evaluate $P(X, Y)$ on any sets X and Y of non-empty subsets od D as follows:

1. A gets X and B gets Y. Both parties know D
2. A sends a message a_1 to B, and B replies with a message b_1 to A.

Definition

Let P be a binary predicate on sets of sets of data elements. We say that P has constant communication complexity if there exist natural numbers k and r and communication protocol between two parties (denoted by A and B) that, for each finite set D of data elements, can evaluate $P(X, Y)$ on any sets X and Y of non-empty subsets od D as follows:

1. A gets X and B gets Y. Both parties know D
2. A sends a message a_1 to B, and B replies with a message b_1 to A.
3. A again sends a message a_2 to B, and B again replies with a message b_2

Definition

Let P be a binary predicate on sets of sets of data elements. We say that P has constant communication complexity if there exist natural numbers k and r and communication protocol between two parties (denoted by A and B) that, for each finite set D of data elements, can evaluate $P(X, Y)$ on any sets X and Y of non-empty subsets od D as follows:

1. A gets X and B gets Y. Both parties know D
2. A sends a message a_1 to B, and B replies with a message b_1 to A.
3. A again sends a message a_2 to B, and B again replies with a message b_2
4. After r message exchanges, both A and B have enough information to evaluate $P(X, Y)$ correctly.

Example

Example

- Let $P(X, Y)$ be true if the maximal cardinality of an element in X is larger than the maximal cardinality of an element in Y.

Example

- Let $P(X, Y)$ be true if the maximal cardinality of an element in X is larger than the maximal cardinality of an element in Y.
- P has constant communication complexity with $k = 1$ and $r = 1$.

Example

- Let $P(X, Y)$ be true if the maximal cardinality of an element in X is larger than the maximal cardinality of an element in Y.
- P has constant communication complexity with $k = 1$ and $r = 1$.
- A sends to B an element of X with maximal cardinality, and B replies with an analogous element for Y.

Example

- Let $P(X, Y)$ be true if the maximal cardinality of an element in X is larger than the maximal cardinality of an element in Y.
- P has constant communication complexity with $k = 1$ and $r = 1$.
- A sends to B an element of X with maximal cardinality, and B replies with an analogous element for Y.
- Both A and B can then evaluate $P(X, Y)$ by comparison of cardinalities.

Example

- Let $P(X, Y)$ be true if the maximal cardinality of an element in X is larger than the maximal cardinality of an element in Y.
- P has constant communication complexity with $k = 1$ and $r = 1$.
- A sends to B an element of X with maximal cardinality, and B replies with an analogous element for Y.
- Both A and B can then evaluate $P(X, Y)$ by comparison of cardinalities.

Lemma 2

The equality, inclusion and disjointness predicates do not have constant communication complexity.

Split Databases

Definition

A temporal database is called split if there is exactly one state whose instance is empty. (This state is called the middle state of the split database.

Split Databases

Definition

A temporal database is called split if there is exactly one state whose instance is empty. (This state is called the middle state of the split database.

Remark

If $I = I_1, I_2, ..., I_n$ is a split database with middle state m then its right part $I_m, ..., I_n$ is denoted by I_{right} and its left part $I_1, ..., I_m$ by I_{left}.

The Language Split-ETL

The Language Split-ETL is a ETL language, whose semantics is only defined on split databases.

The Language Split-ETL

The Language Split-ETL is a ETL language, whose semantics is only defined on split databases.

Remark

Syntactically, Split-ETL differs from ETL only in that each temporal operator $L^+(L^-)$ is split into a "left" and a "right" version L^+_{left} and L^+_{right} $\left(L^-_{left} \text{ and } L^-_{right}\right)$.

Definition

Let I be a split database of length n with middle state m.
For each state j of I we define:

$$left(j) := \begin{cases} j & \text{if } j \leqslant m \\ m & \text{if } j \geqslant m \end{cases}$$

and

$$right(j) := \begin{cases} 1 & \text{if } j \leqslant m \\ j - m + 1 & \text{if } j \geqslant m \end{cases}$$

Semantics of the split temporal operators

Semantics of the split temporal operators is defined as follows:

Semantics of the split temporal operators

Semantics of the split temporal operators is defined as follows:

- $I, j \models L^{-}_{left}\theta$ if $I_{left}, left(j) \models L^{-}\theta$

Semantics of the split temporal operators

Semantics of the split temporal operators is defined as follows:

- $I, j \models L^{-}_{left}\theta$ if $I_{left}, left(j) \models L^{-}\theta$
- $I, j \models L^{+}_{left}\theta$ if $I_{left}, left(j) \models L^{+}\theta$

Semantics of the split temporal operators

Semantics of the split temporal operators is defined as follows:

- $I, j \models L^{-}_{left}\theta$ if $I_{left}, left(j) \models L^{-}\theta$
- $I, j \models L^{+}_{left}\theta$ if $I_{left}, left(j) \models L^{+}\theta$
- $I, j \models L^{-}_{right}\theta$ if $I_{right}, right(j) \models L^{-}\theta$

Semantics of the split temporal operators

Semantics of the split temporal operators is defined as follows:

- $I, j \models L^{-}_{left}\theta$ if $I_{left}, left(j) \models L^{-}\theta$
- $I, j \models L^{+}_{left}\theta$ if $I_{left}, left(j) \models L^{+}\theta$
- $I, j \models L^{-}_{right}\theta$ if $I_{right}, right(j) \models L^{-}\theta$
- $I, j \models L^{+}_{right}\theta$ if $I_{right}, right(j) \models L^{+}\theta$

Lemma 3

On split databases, each ETL formula is equivalent to a split-ETL formula.

Boollean query Q_P

Let P be a binary predicate on sets of sets. Let us consider the Bollean query Q_P on split databases defined as follows. For a split database $I = I_1, ..., I_n$ with middle state m
$Q_P(I) = true$ if $P(L, R)$ holds, where
$L = \{I_j : 1 \leqslant j < m\}$ and $R = \{I_j : m < j \leqslant n\}$

Lemma 4

If Q_P is expressible in ETL, then P has constant communication complexity.

Inexpressibility

Lemma 4

If Q_P is expressible in ETL, then P has constant communication complexity.

Theorem

Over schemas containing at least one relation of non-zero arity, there are queries expressible in TS-FO but not in ETL.

Inexpressibility

Lemma 4

If Q_P is expressible in ETL, then P has constant communication complexity.

Theorem

Over schemas containing at least one relation of non-zero arity, there are queries expressible in TS-FO but not in ETL.

Proof (Sketch)

Query Q :

Are there two different states with the same instance?

is expressible in TS-FO but not in ETL.

Query Q is expressible in TS-FO:

$$\exists t \exists t' (t \neq t' \wedge (\forall x)(S(x,t) \leftrightarrow S(x,t')))$$

Proof (Sketch) ...

On the class of split databases whose left and right parts do not contain repetitions, Q corresponds to Q_P, where P is the predicate. By lemma 2 P does not have constant communication complexity. Hence, by lemma 4 Q is not expressible in ETL.

Proof (Sketch) ...

On the class of split databases whose left and right parts do not contain repetitions, Q corresponds to Q_P, where P is the predicate. By lemma 2 P does not have constant communication complexity. Hence, by lemma 4 Q is not expressible in ETL.

Corollary

Over schemas containing al least one relation of non-zero arity, TL is strictly less expressive than TS-FO.

Conclusions

Conclusions

- Queries in TL (ETL) are simpler than queries in TS-FO.

Conclusions

- Queries in TL (ETL) are simpler than queries in TS-FO.
- There are queries expressible in ETL and inexpressible in TS-FO.

Conclusions

- Queries in TL (ETL) are simpler than queries in TS-FO.
- There are queries expressible in ETL and inexpressible in TS-FO.
- There are queries expressible in TS-FO and inexpressible in ETL.

The End