PSPACE hardness of mixed world query answering for Atomic Queries under Guarded TGDs

Michael Benedikt and Pierre Bourhis

1 Introduction

Consider a vocabulary divided into open-world relations O and closed-world relations C, a set of logical sentences ("integrity constraints" henceforward) Σ, a boolean query Q, and an instance $\mathcal{D}$. We say that Q is *certain with respect to* $O, C, \Sigma, \mathcal{D}$ if:

for every instance $\mathcal{D}'$ that extends $\mathcal{D}$ and agrees with $\mathcal{D}$ on relations in C, Q holds on $\mathcal{D}'$.

As in most prior work, we focus on the case in which Q is a *conjunctive query*, an existential quantification of conjunctions of atoms, and the case where Q is a disjunction of conjunctive queries.

We will be interested in the *data complexity* of the problem: fixing O, C, Σ, Q and varying $\mathcal{D}$. It is well-known that even when the constraints Σ have a very restricted form, the data-complexity can be coNP-hard [AD98]. coNP-hardness in data complexity can even occur when the arity of relations is at most 2 [LSW15, LSW13]. [BBPtC16] provided examples of schemas with arity larger than 2 where the situation is even worse. We can get O, C, Σ, Q such that the data complexity is ExpTime-hard and: Q is a UCQ, while Σ consists of *linear TGDs*. Linear TGDs are sentences of the form

$$\forall \vec{x}\, A(\vec{x}) \rightarrow \exists \vec{y}\, B(\vec{x}, \vec{y})$$

where $A(\vec{x})$ and $B(\vec{x}, \vec{y})$ are relational atoms with variables coming from $\vec{x}$ and $\vec{x} \cup \vec{y}$ respectively.

[BBPtC16] uses the terminology "visible relation" for a closed-world relation, and "invisible relation" for open-world relation, and considers only inputs where the open-world relations are empty. Clearly, if one has hardness for this case, one has hardness for the general case.

The ExpTime-hardness argument in [BBPtC16] required both a higher-arity schema and Q a UCQ. However we can show that it can be adapted to an atomic query. In this document we do not prove a tight bound, but we provide a detailed argument that the data complexity can be PSpace-hard for O, C, Σ, Q even if Q is an *atomic* 0*-ary query* (a single 0-ary relation). This cannot happen for linear TGDs, but we show that it can occur even when the constraints Σ can be *Guarded TGDs*: of the form

$$\forall \vec{x}\, A(\vec{x}) \wedge \varphi(\vec{x}) \rightarrow \exists \vec{y} B(\vec{x}, \vec{y})$$

where φ is a conjunction of atoms and all variables of φ occur in the relational atom $A(\vec{x})$.

That is, we show:

Theorem 1. *There is a schema consisting of closed and open world relations, a set of guarded TGD constraints Σ, and an atomic query Q such that the problem of determining whether Q is certain with respect to $O, C, \Sigma, \mathcal{D}$ is is* PSpace*-hard. That is, the problem can be* PSpace*-hard in data complexity.*

2 Construction

We will encode the halting problem for deterministic PSpace Turing machines whose tape alphabet has only two values 0 and 1. Thus the transition function δ of a generic machine takes as input a control state x and a number in $\{0, 1\}$, returning a head movement ρ ($\leftarrow$ or $\rightarrow$), a new value in $\{0, 1\}$ to be written, and a new state y. A machine M comes with a polynomial space bound $p_M(n)$, and we will assume that on any input to such a machine of size n, the head of

M never moves to the right of $p_M(n)$. We also assume that M never attempts a left-ward move when it is on the initial cell on the tape.

We first give the "closed-world" relations, along with their intended semantics:

We have some relations that code the states and transition function of a machin:

- NonFinalState(x) stores all the non-final states.
- FinalState(x) stores the final states of the machine.
- TransHead1(x, y), which stores the states x and y such that $\delta(x, 1) = (\alpha, \beta, y)$.
- TransHead0(x, y), storing the states x and y such that $\delta(x, 0) = (\alpha, \beta, y)$.
- MoveLeft1(x), storing states x such that $\delta(x, 1) = (\leftarrow, \alpha, \beta)$.
- MoveLeft0(x), storing states x such that $\delta(x, 0) = (\leftarrow, \alpha, \beta)$.
- MoveRight1(x), storing states x such that $\delta(x, 1) = (\rightarrow, \alpha, y)$.
- MoveRight0(x), storing states x such that $\delta(x, 0) = (\rightarrow, \alpha, y)$.
- Tran0to1(x), storing states x such that $\delta(x, 0) = (\alpha, 1, y)$.
- Tran0to0(x), storing states x such that $\delta(x, 0) = (\alpha, 0, y)$.
- Tran1to0(x), storing the states x such that $\delta(x, 1) = (\alpha, 0, y)$.
- Tran1to1(x), storing the states x such that $\delta(x, 1) = (\alpha, 1, y)$.

We also have closed-world relations that code the tape:

- CellSucc(x, y), stores the successor relation among identifiers for each cell of the tape; if the input has size n, such identifiers can be taken to be a number between 1 and n. The pair (x, y) is in CellSucc iff y is the index of the cell which is the successor of the cell of x.
- Cell$^{\neq}(x, y)$ stores the pair of indices of values such that x and y are different.
- FirstCell(x) stores the first cell.

Finally we have closed-world relations concerning the first two configurations:

- Conf0$(x, y, z_1, q_1, z_2, q_2, z_3, z_4)$ where x is the index of the initial configuration, y is the index of the successor configuration of x, z_1 is the position of the head in x, q_1 is the state of machine in x, z_2 is the position of the head in y, q_2 is the state of machine in y, z_3, z_4 are the indices of two consecutive cells.
- InitIsOne(x, y) where x is the index of one of the first two configurations, y is the index of the cell.
- InitIsZero(x, y) where x is the index of one of the first two configurations, y is the index of the cell.

We now give the open-world relations (i.e. those that are "hidden" in the terminology of [BBPtC16]), along with their intended semantics:

- Config$(x, y, z_1, q_1, z_2, q_2, z_3, z_4)$, whose intended semantics is the same as Conf0 but for arbitrary configurations. That is, x is the index of a configuration, y is the index of the successor configuration of x, z_1 is the position of the head in x, q_1 is the state of machine in x, z_2 is the position of the head in y, q_2 is the state of machine in y, z_3, z_4 are the indices of two consecutive cells.
- IsOne(x, y) whose intended semantics is the same as InitIsOne but for arbitrary configurations: x is the index of any configuration, y is the index of the cell.

- $\mathsf{IsZero}(x, y)$ where x is the index of any configuration, y is the index of the cell.
- $\mathsf{Good}()$, which represents that the machine halts.

The goal will be to arrange the constraints Σ and query Q so that a counterexample to certainty of Q corresponds to a halting computation. The difficulty will be to enforce that in a counterexample instance $\mathcal{D}'$, Config represents a correct transition relation. We cannot use non-guarded constraints to enforce properties of pairs of configurations; neither can we can discuss pairs of Config atoms in the query, which must be atomic.

We now give the set of constraints Σ. are the following:

- An *initial configuration constraint* relating the visible relations storing initial configuration information to the invisible relations storing generic configuration information:

$$\forall x, y, z_1, q_1, z_2, q_2, z_3, z_4\ \mathsf{Conf0}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \\ \rightarrow \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4)$$

- A *cell successor constraint* ensuring that each cell in a configuration has a successor cell:

$$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \longrightarrow \exists z_5\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_4, z_5) \wedge \mathsf{CellSucc}(z_4, z_5)$$

- A set of *transition existence constraints*, enforcing that the machine makes progress.
  - A constraint enforcing the creation of a new configuration with some correctness conditions related to the transition, in the case where the label of the cell on which the head lies is equal to 1 and the transition moves the head to the left.

$$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4, \\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{FirstCell}(z_3) \wedge \mathsf{IsOne}(y, z_2) \wedge \mathsf{NonFinalState}(q_2) \wedge \mathsf{MoveLeft1}(q_2) \longrightarrow \\ \exists w, z_5, q_5\ \mathsf{Config}(y, w, z_2, q_2, z_5, q_5, z_3, z_4) \wedge \mathsf{TransHead1}(q_2, q_5) \wedge \mathsf{CellSucc}(z_5, z_2)$$

  - A constraint enforcing creation of a new configuration with some correctness conditions concerning the transition in the case where the label of the cell is as in the previou case, but the transition moves the head to the right.

$$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4, \\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{FirstCell}(z_3) \wedge \mathsf{IsOne}(y, z_2) \wedge \mathsf{NonFinalState}(q_2) \wedge \mathsf{MoveRight1}(q_2) \longrightarrow \\ \exists w, z_5, q_5\ \mathsf{Config}(y, w, z_2, q_2, z_5, q_5, z_3, z_4) \wedge \mathsf{TransHead1}(q_2, q_5) \wedge \mathsf{CellSucc}(z_2, z_5)$$

  - A constraint enforcing the creation of a new configuration as above but where the label of the cell contaiing the head is equal to 0 and the transition moves the head to the right.

$$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4, \\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{FirstCell}(z_3) \wedge \mathsf{IsZero}(y, z_2) \wedge \mathsf{NonFinalState}(q_2) \wedge \mathsf{MoveRight0}(q_2) \longrightarrow \\ \exists w, z_5, q_5\ \mathsf{Config}(y, w, z_2, q_2, z_5, q_5, z_3, z_4) \wedge \mathsf{TransHead0}(q_2, q_5) \wedge \mathsf{CellSucc}(z_2, z_5)$$

  - A constraint enforcing the creation of a new configuration in the case where the label of the cell containing the head is equal to 0 and the transition moves the head to the left.

$$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4, \\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{FirstCell}(z_3) \wedge \mathsf{IsZero}(y, z_2) \wedge \mathsf{NonFinalState}(q_2) \wedge \mathsf{MoveLeft0}(q_2) \longrightarrow \\ \exists w, z_5, q_5\ \mathsf{Config}(y, w, z_2, q_2, z_5, q_5, z_3, z_4) \wedge \mathsf{TransHead1}(q_2, q_5) \wedge \mathsf{CellSucc}(z_5, z_2)$$

- A collection of *transition correctness constraints*, ensuring that in every transition of the machine the values assigned to cells are correct.
  - A constraint enforcing that the label of z_1 is correct after a transition where the head position is z_1, the previous label is 1 and the transition changed it to 0

    $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_3, q_2, z_1, z_4) \wedge \mathsf{IsOne}(x, z_1) \wedge \mathsf{Tran1to0}(q_1) \longrightarrow \mathsf{IsZero}(y, z_1)$$

  - A constraint enforcing that the label of z_1 is correct after a transition in which the head position is z_1, the previous label is 1 and the transition changed it to 1

    $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{IsOne}(x, z_1) \wedge \mathsf{Tran1to1}(q_1) \longrightarrow \mathsf{IsOne}(y, z_1)$$

  - A constraint enforcing that the label of z_1 is correct after a transition where the head position is z_1, the previous label is 0 and the transition changed it to 0

    $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{IsOne}(x, z_1) \wedge \mathsf{Tran0to0}(q_1) \longrightarrow \mathsf{IsZero}(y, z_1)$$

  - A constraint enforcing that the label of z_1 is correct after a transition where the head position is z_1, the previous label is 0 and the transition changed it to 1

    $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{IsOne}(x, z_1) \wedge \mathsf{Tran0to1}(q_1) \longrightarrow \mathsf{IsOne}(y, z_1)$$

  - A constraint enforcing that the label of z_3 is still 1 after any transition where z_3 was not the head position in the previous configuration, e.g different than z_1.

    $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{IsOne}(x, z_3) \wedge \mathsf{Cell}^{\neq}(z_1, z_3) \longrightarrow \mathsf{IsOne}(y, z_3)$$

  - A constraint enforcing that the label of z_3 is still 0 after any transition in which z_3 was not the head position in the previous configuration.

    $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4,\ \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{IsZero}(x, z_3) \wedge \mathsf{Cell}^{\neq}(z_1, z_3) \longrightarrow \mathsf{IsZero}(y, z_3)$$

  Notice that it is in the above constraints that we are taking advantage of the fact that Config atoms code configuration information redundantly with every pair of cell positions z_3, z_4; this allows the constraint above to be guarded.

- A *unique value constraint* ensuring that a configuration does not give any cell two distinct labels.

  $$\forall x, y\ \mathsf{IsOne}(x, y) \wedge \mathsf{IsZero}(x, y) \longrightarrow \mathsf{Bad}$$

- A *final state constraint* stating that if the configuration reaches a final state the predicate Good holds (which will ensure that the query holds).

  $$\forall x, y, q_1, q_2, z_1, z_2, z_3, z_4, \mathsf{Config}(x, y, z_1, q_1, z_2, q_2, z_3, z_4) \wedge \mathsf{FinalState}(q_2) \longrightarrow \mathsf{Good}()$$

The query Q is the atomic query Good().

3 Correctness

We now show that there is a reduction from the halting problem for a deterministic PSPACE machine M on an input w to the certain answer problem for $\mathcal{O}, C, \Sigma, \mathcal{D}$ where $\mathcal{O}, C, \Sigma$ are defined above.

Given a PSPACE machine M with polynomial space bound p_M and input w, our reduction produces $\mathsf{InstanceOf}(M, w)$, the instance for the closed-world predicates formed by filling in the predicates in the "obvious way". E.g. $\mathsf{CellSucc}$ stores the successor relation among the $p_M(|w|)$ cells used by M, $\mathsf{FinalState}$ storing the final states of M, and so forth. The only quirk in our representation will be that the successor of the last cell is itself. Note that computing $\mathsf{Conf0}, \mathsf{InitIsOne}, \mathsf{InitIsZero}$ requires determining the first two configurations in the unique run of the machine, but this can clearly be done in polynomial time. All of the open-world relations are empty.

We claim that this gives the desired reduction.

In one direction, suppose that the machine M does not halt on w. Coding the witness run in the obvious way, we get a counterexample to the closed- and open- world query answering problem for Good on $\mathsf{InstanceOf}(M)$. The self-loop condition on the last cell in our representation is needed in order to ensure that we can add the necessary successor facts for the last cell of the tape in the transition correctness constraint.

In the other direction, suppose that M does halt on w. We will show that $Q = \mathsf{Good}$ is certain under the hybrid closed- and open- world semantics for $\mathsf{InstanceOf}(M, w)$. Suppose by way of contradiction there is a counterexample instance $\mathcal{D}'$. We inductively create an infinite sequence of tape configurations $\mathsf{Config}_1 \ldots$ which witness that M does not halt on w. Our inductive invariant on $\mathsf{Config}_1 \ldots \mathsf{Config}_n$ is that the configurations obey the transition function of M and the final two configurations Config_{n-1} and Config_n will be associated with configuration ids x_{n-1} and x_n (respectively) from $\mathcal{D}'$ such that:

- the second-to-last configuration has head at cell c_{n-1} in state q_{n-1} for c_{n-1}, q_{n-1} such that there is a a fact $\mathsf{Config}(x_{n-1}, x_n, c_{n-1}, q_{n-1}, \ldots)$ in $\mathcal{D}'$.
- the second-to-last configuration has cell c associated with value 1 if and only if $\mathcal{D}'$ has facts of the form $\mathsf{Config}(x_{n-1} \ldots)$ and $\mathsf{IsOne}(x_{n-1}, c)$, and similarly for value 0.
- the last configuration has head at cell c_n in state q_n for c_n, q_n such that there is a fact $\mathsf{Config}(x_{n-1}, x_n, c_{n-1}, q_{n-1}, c_n, q_n \ldots)$
- the last configuration has cell c associated with value 1 if and only if $\mathcal{D}'$ has facts $\mathsf{Config}(x_{n-1}, x_n \ldots)$ and $\mathsf{IsOne}(x_n, c)$, and similarly for value 0.

The base case is handled by the initial configuration constraint.

In the induction step, we assume we have $\mathsf{Config}_1 \ldots \mathsf{Config}_n$, and identifiers $x_1 \ldots x_n$ for which the hypothesis holds; thus in particular there is a fact $\mathsf{IsOne}(x_n, c_n)$ or $\mathsf{IsZero}(x_n, c_n)$ in $\mathcal{D}'$. Further we know that q_n is not a final state, since otherwise the final state constraint would have fired, making the query Q hold in $\mathcal{D}'$.

We do a case analysis based on whether $\mathsf{IsOne}(x_n, c_n)$ or $\mathsf{IsZero}(x_n, c_n)$ holds, and what kind of transition occurs on q_n in M on the corresponding value. We consider only the case where $\mathsf{IsOne}(x_n, c_n)$ holds and the transition of M on q_n with 1 is to the left. The other 3 cases are analogous.

In this case we have the fact $\mathsf{NonFinalState}(q_n)$ and $\mathsf{MoveLeft1}(q_n)$ in $\mathcal{D}'$, since these facts held in $\mathcal{D}$ and these involved closed-world relations. There is also some z_3 such that $\mathsf{FirstCell}(z_3)$ holds, since this held in $\mathcal{D}$ and involved a closed-world relation. The second part of the induction hypothesis implies that for some z_4 we had a fact

$$\mathsf{Config}(x_{n-1}, x_n, c_{n-1}, q_{n-1}, c_n, q_n, z_3, z_4)$$

in $\mathcal{D}'$.

The first transition existence constraint now guarantees that for some $x_{n+1}, c_{n+1}, q_{n+1}$ $\mathsf{Config}(x_n, x_{n+1}, c_n, q_n, c_{n+1}, q_{n+1}, z_3, z_4) \wedge \mathsf{TransHead1}(q_n, q_{n+1}) \wedge \mathsf{CellSucc}(c_{n+1}, c_n)$ holds in $\mathcal{D}'$.

We consider the configuration Config_{n+1} formed as follows:

- the configuration has head at cell c_{n+1} in state q_{n+1}
- the configuration has cell c associated with value 1 if and only if $\mathsf{IsOne}(x_{n+1}, c)$ holds in $\mathcal{D}'$ and similarly for value 0

We claim that Config_{n+1} is a valid configuration, and that extending the sequence to $\mathsf{Config}_1 \dots \mathsf{Config}_{n+1}$ and $x_1 \dots x_{n+1}$ preserves the inductive invariant.

We first consider the properties required for the second-to-last element of the sequence. The first property we need to verify is that Config_n has head at cell c_n in state q_n for c_n, q_n such that there is a a fact $\mathsf{Config}(x_n, x_{n+1}, c_n, q_n, \dots)$ in $\mathcal{D}'$. This is immediate from the construction of x_{n+1}. The second property is that Config_n has cell c associated with value 1 if and only if there are facts $\mathsf{Config}(x_n \dots)$ and $\mathsf{IsOne}(x_n, c)$ in $\mathcal{D}'$, and similarly for value 0. Note that we have a fact $\mathsf{Config}(x_n \dots)$ and the facts $\mathsf{IsOne}(x_n, c)$ and $\mathsf{IsZero}(x_n, c)$ corresponding to Config_n by induction.

We now show that above we have defined a valid configuration Config_{n+1}, and one that obeys the transition function. We need first to show that for each cell c, exactly one of $\mathsf{IsOne}(x_{n+1}, c), \mathsf{IsZero}(x_{n+1}, c)$ holds in $\mathcal{D}'$. The fact that both of these cannot hold follows from the unique value constraint and the fact that Bad does not hold in $\mathcal{D}$, and hence (since Bad is closed-world) does not hold in $\mathcal{D}'$. To see that one of them holds in $\mathcal{D}'$, we do case analysis based on where c is relative to c_n, the symbol of c in the configuration associated with x_n, and the kind of transition there is going from q_n to q_{n+1}. We consider the subcase where c is not c_n and the value at c in the configuration associated with x_n is 1. The first fact implies $\mathsf{Cell}^{\neq}(c_n, c)$ holds in $\mathcal{D}$; hence (since the predicate is closed-world) it holds in $\mathcal{D}'$. The second fact and the induction hypothesis imply that $\mathsf{IsOne}(x_n, c)$ holds in $\mathcal{D}'$. Note that $\mathsf{CellSucc}$ is closed-world, and hence in $\mathcal{D}'$ there is a chain of $\mathsf{CellSucc}$ facts leading from z_3 to c. Inductively applying the cell successor constraint, we see that $\mathsf{Config}(x_n, x_{n+1}, c_n, q_n, c_{n+1}, q_{n+1}, c, z_4)$ holds in $\mathcal{D}'$ for some z_4. By the transition correctness constraints, since $\mathsf{IsOne}(x_n, c)$ and $\mathsf{Cell}^{\neq}(c_2, c)$ holds in $\mathcal{D}'$, we have that $\mathsf{IsOne}(x_{n+1}, c)$ holds in $\mathcal{D}'$. Notice that this argument shows also that the value at c is the one required by the transition function.

Note that since $\mathsf{IsOne}(x_n, c_n)$ and $\mathsf{Config}(x_{n-1}, x_n, c_{n-1}, q_{n-1}, c_n, q_n, \dots)$ hold in $\mathcal{D}'$ the inductive hypothesis guarantees that Config_n has the tape head at c_n, control state q_n, and a 1 under the head. Since $\mathsf{TransHead1}(q_n, q_{n+1})$ holds in $\mathcal{D}'$ and $\mathsf{TransHead1}$ is closed-world, $\mathsf{TransHead1}(q_n, q_{n+1})$ holds in $\mathcal{D}$, and thus q_{n+1} is the correct successor state of q_n under the transition function.

Finally, we note that the resulting run can never reach a halting state, since otherwise the final state constraint would imply that the query Q is true on $\mathcal{D}'$, contrary to the fact that $\mathcal{D}'$ is a counterexample instance.

References

[AD98] Serge Abiteboul and Oliver Duschka. Complexity of answering queries using materialized views. In *PODS*, 1998.

[BBPtC16] Michael Benedikt, Pierre Bourhis, Gabriele Puppis, and Balder ten Cate. Querying visible and invisible information. In *LICS*, 2016.

[LSW13] C. Lutz, I. Seylan, and F. Wolter. Ontology-based data access with closed predicates is inherently intractable(sometimes). In *IJCAI*, 2013.

[LSW15] Carsten Lutz, Inanç Seylan, and Frank Wolter. Ontology-mediated queries with closed predicates. In *IJCAI*, 2015.