A Note on the Relationship between Different Types of Correction Queries[⋆]

Cristina Tîrnăucă

Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
cristina.bibire@estudiants.urv.es
http://www.grlmc.com

Abstract. The adult-child interaction which takes place during the child's language acquisition process has been the inspiration for Angluin's teacher-learner model [1], the forerunner of today's active learning field. But the initial types of queries have some drawbacks: equivalence queries are both unrealistic and computationally costly; membership queries, on the other hand, are not informative enough, not being able to capture the feedback received by the child when he or she makes mistakes. This is why a new type of query (called correction query), weaker than the first one and more informative than the second, appeared. While in the case of natural languages it is well understood what correcting means, in formal language theory different objects may require different types of corrections. Therefore, several types of correction queries have been introduced so far. In this paper we investigate the relations existing between different models of correction queries, as well as their connection to other well-known Gold-style and query learning models. The study comprises results obtained in the general case when time complexity issues are ignored, and in the restricted case when efficiency constraints are imposed.

Keywords: query learning, Gold-style learning, correction query.

1 Introduction

The way children learn their mother language is an amazing process. They receive examples of words in the vocabulary and sentences in that language, and after some transitory period - in which they still make mistakes and are corrected by adults - they are suddenly able to express themselves fluently and errorless.

It has been argued that the formal model that best describes the child-adult interaction within the process of child acquiring his or her native language is the *query learning model* [1]. The most investigated types of queries, and in the

[⋆] This work was possible thanks to the FPU Fellowships AP2004-6968 from the Spanish Ministry of Education and Science.

same time the first introduced, are *membership queries* (MQs) and *equivalence queries* (EQs).

There are quite a few reasons though for which people working in grammatical inference, and especially in active learning, have been trying to find effective query learning algorithms that avoid the use of EQs. First of all, EQs are computationally costly. Secondly, they are quite unnatural for a real-life setting: no child would ever ask his mother if the current hypothesis represents the correct grammar of the language. Finally, it might happen that the teacher does not even have a grammar for the target language - take, for example, the case of native speakers that did not ever studied grammar.

On the other hand, when answering a MQ in the negative way, no other information is provided by the teacher. Inspired by the way adults guide the process of children's language acquisition by correcting them when necessary, the authors of [2] propose a modified version of MQs, called *correction query* (CQ), that incorporates this idea. More precisely, the difference consists in the fact that for strings not belonging to the target language, the teacher must provide the learner with a *correction*.

Whereas in a real-life setting correcting an ungrammatical utterance is done, most of the times, by replacing the error with a correct (sequence of) word(s), if the target is a formal language, then one needs to adapt the type of correction to best suit the particularities of that language class. And indeed, since their introduction, several types of corrections have been proposed in order to learn different objects: prefix correction queries [2] and length bounded correction queries [3] for deterministic finite automata, edit distance based correction queries for balls of strings [4], regular expressions and pattern languages [5], the nearest positive example [6] for sets of positive integers, and structural correction queries for regular tree languages [7].

Intuitively, this new type of query can be placed somewhere between MQs and EQs. But what exactly can we learn with CQs, and what can be done in polynomial time? These are the questions whose answers constitute the object of the present paper. The first steps in this research direction have been done already: Tîrnăucă and Kobayashi [8] investigate the relations existing between the model of learning with prefix correction queries (PCQs) and other well-known Gold-style and query learning models when time complexity issues are neglected (they show, among other things, that learning with PCQs is strictly less powerful than learning with EQs, and more powerful than the model of learning with MQs). Moreover, this study is continued in [9] by imposing efficiency constraints.

In this paper we focus on the identification of formal languages ranging over indexable classes of non-empty recursive languages as target concepts when the learner is allowed to ask one of the other two types of CQs: length bounded correction queries (LBCQs) and edit distance correction queries (EDCQs).

The article is organized as follows. Preliminary notions are presented in Section 2. In the third section we show that when we neglect time complexity issues, learning with LBCQs or EDCQs is basically the same as learning with MQs. In Section 4 we concentrate on polynomial time algorithms and we present the relations between different models of learning with CQs. We conclude with further remarks and future work topics in Section 5.

2 Preliminaries

Let Σ be a finite set of symbols called *alphabet*, and Σ^* the set of strings over Σ. A *language* L over Σ is a subset of Σ^*. The elements of L are called *strings*. Let u, v, w be strings in Σ^* and $|w|$ be the length of the string w. λ is a special string called the *empty string* and has length 0. We denote by uv the concatenation of the strings u and v. If $w = uv$ for some u, v in Σ^*, then u is a *prefix* of w and v is a *suffix* of w. By $Tail_L(u)$ we denote the set $\{v \mid uv \in L\}$.

Assume that Σ is a totally ordered set, and let $\prec_L$ be the lexicographical order on Σ^*. Then, the *lex-length order* $\prec$ on Σ^* is defined by: $u \prec v$ if either $|u| < |v|$, or else $|u| = |v|$ and $u \prec_L v$. In other words, strings are compared first according to length and then lexicographically. In the rest of the paper strings are always compared with respect to the lex-length order.

The edit distance between two strings w and w', denoted $d(w, w')$ in the sequel, is the minimum number of *edit operations* needed to transform w into w' and can be computed in $O(|w| \cdot |w'|)$ time by dynamic programming [10]. The edit operations are either (1) *deletion*: $w = uav$ and $w' = uv$, or (2) *insertion*: $w = uv$ and $w' = uav$, or (3) *substitution*: $w = uav$ and $w' = ubv$, where $u, v \in \Sigma^*, a, b \in \Sigma$ and $a \neq b$.

2.1 Learning Models

An *indexed family* (or *indexable class*) Let $\mathcal{C} = (L_i)_{i \geq 1}$ is a recursive enumeration of non-empty languages such that membership in L_i is uniformly decidable for all $i \geq 1$, i.e., there is a computable function that, for any $w \in \Sigma^*$ and $i \geq 1$, returns 1 if $w \in L_i$, and 0 otherwise.

Gold-Style Learning. An *inductive inference machine* (IIM) $\mathcal{A}lg$ is an algorithmic device that reads longer and longer initial segments σ of a text (informant), and outputs numbers as its hypotheses. Given a text (an informant) σ for a language $L \in \mathcal{C}$, *$\mathcal{A}lg$ learns L from σ* if the sequence of hypotheses output by $\mathcal{A}lg$, when fed σ, stabilizes on a number i with $L_i = L$. We say that *$\mathcal{A}lg$ learns $\mathcal{C}$ from text (informant)* if it identifies each $L \in \mathcal{C}$ from every corresponding text (informant). A slightly modified version of the learning in the limit model is the so-called model of *conservative learning* (see [12,13] for more details). A conservative IIM is only allowed to change its mind in case its actual guess contradicts the data seen so far.

We denote by *LimTxt* (*LimInf*) the collection of all indexable classes $\mathcal{C}$ for which there is an IIM $\mathcal{A}lg$ such that $\mathcal{A}lg$ identifies $\mathcal{C}$ from text (informant). One can similarly define *ConsvTxt* and *ConsvInf* for which the inference machines should be conservative IIMs.

Although an IIM is allowed to change its mind finitely many times before returning its final and correct hypothesis, in general it is not decidable whether or not it has already output its final hypothesis. In case that for a given indexable class $\mathcal{C}$, there exists an IIM $\mathcal{A}lg$ such that given any language $L \in \mathcal{C}$ and any text (or informant) for L, the first hypothesis i output by $\mathcal{A}lg$ is already correct (i.e., $L_i = L$), we say that $\mathcal{A}lg$ *finitely identifies* $\mathcal{C}$ (see [14]). The corresponding models *FinTxt* and *FinInf* are defined as above.

Query Learning. In this model a learner has access to an oracle or a teacher that truthfully answers queries of a specified kind. Conform [15], a *query learner* $\mathcal{A}lg$ is an algorithmic device that, depending on the reply of the previous queries, either computes a new query, or returns a hypothesis and halts. More formally, let $\mathcal{C}$ be an indexable class and L an arbitrary language in $\mathcal{C}$. The query learner $\mathcal{A}lg$ *learns L using some type of queries* if it eventually halts and its only hypothesis, say i, correctly describes L (i.e., $L_i = L$). So, $\mathcal{A}lg$ returns its unique and correct guess i after only finitely many queries. Moreover, $\mathcal{A}lg$ *learns the class* $\mathcal{C}$ *using some type of queries* if it learns every language of that class using queries of the specified type.

Apart from the well-known membership and equivalence queries, in this paper we investigate three types of correction queries. Let L be a language over the alphabet Σ and w a string in Σ^*.

- **Prefix correction queries** (Becerra, Dediu, Tîrnăucă [2])
  The *prefix correcting string of w with respect to L*, denoted $C_L(w)$, is
  $$C_L(w) = \begin{cases} \min\{v \mid v \in \mathit{Tail}_L(w)\}, & \text{if } \mathit{Tail}_L(w) \neq \emptyset \\ \Theta, & \text{otherwise.} \end{cases}$$
  Hence, C_L is a function from Σ^* to $\Sigma^* \cup \{\Theta\}$. Note that $C_L(w)$ is λ if and only if $w \in L$, and $C_L(w)$ equals Θ if and only if w is not the prefix of any of the strings in L.
- **Length bounded correction queries** (Tîrnăucă [3])
  Let us fix an integer l. The *l-bounded correction of w with respect to L*, denoted $C_L^l(w)$, is
  $$C_L^l(w) = \{v \in \mathit{Tail}_L(w) \mid |v| \leq l\}.$$
  So, C_L^l is a function from Σ^* to $\mathcal{P}(\Sigma^*)$. Note that $\lambda \in C_L^l(w)$ if and only if $w \in L$.
- **Edit distance correction queries** (Becerra et al. [4])
  The *edit distance correction of w with respect to L*, denoted $\textsc{Edc}_L(w)$, is
  $$\textsc{Edc}_L(w) = \begin{cases} \text{Yes}, & \text{if } w \in L \\ \text{one string of } \{w' \in L \mid d(w, w') \text{ is minimum}\}, & \text{if } w \notin L. \end{cases}$$

The collection of all indexable classes $\mathcal{C}$ for which there is a query learner $\mathcal{A}lg$ such that $\mathcal{A}lg$ learns $\mathcal{C}$ using MQs is denoted by *MemQ*. *EquQ*, *PCorQ*, *lBCorQ* and *EditCorQ* are defined similarly for the models of learning with EQs, PCQs, l-bounded correction queries (lBCQs) and EDCQs, respectively.

There is a strong relation between query learning models and Gold-style learning models. The following strict hierarchy holds [15]:

$$\mathit{FinTxt} \subset \mathit{FinInf} = \mathit{MemQ} \subset \mathit{ConsvTxt} \subset \mathit{LimTxt} \subset \mathit{LimInf} = \mathit{EquQ}.$$

3 Learning with Correction Queries

In [8], necessary and sufficient conditions for a language class to be learnable with PCQs are given, facilitating a comparison between the model of learning with PCQs and other well-known learning models. The results can be summarized as follows:

– *MemQ* is strictly included in *PCorQ*,
– *PCorQ* and *ConsvTxt* are incomparable, and
– *PCorQ* is strictly included in *LimTxt*.

We continue this study for LBCQs and EDCQs.

3.1 Learning with Length Bounded Correction Queries

Note that in the case of DFAs, returning the answer to a PCQ can be easily done in polynomial time. However, when the target concept ranges over arbitrary recursive languages, the answer to a PCQ might be very long or not even computable (given a recursive language L and a string w, one cannot decide, in general, if w is a prefix of a string in L). A possible solution to avoid very long (or infinite) searches is to restrict the search space to only short enough suffixes. So, let us consider the model in which the learner must identify the target language after asking a finite number of lBCQs for a fixed integer $l \geq 0$.

Since any 0BCQ can be simulated by a MQ and the other way around, it is clear that when $l = 0$, learning with lBCQs is equivalent to learning with MQs. It is though less straightforward that the same property holds for an arbitrary l. We show in the sequel that for any l, a language class is learnable with MQs if and only if it is learnable with lBCQs.

Theorem 1. *For any $l \geq 0$, lBCorQ = MemQ.*

Proof. Since for any language L and any string w in Σ^*, if we know the answer to $C_L^l(w)$ we also know if the string w is in L or not, it is clear that *lBCorQ* includes *MemQ*. Hence, we have to show only that *lBCorQ* $\subseteq$ *MemQ*.

Let us consider a language class $\mathcal{C}$ in *lBCorQ*, and let $\mathcal{A}lg$ be an l-bounded correction query learner for $\mathcal{C}$. We modify $\mathcal{A}lg$ such that instead of submitting an lBCQ for a given string w, to submit MQs for all the strings wu with $u \in \Sigma^{\leq l}$. The learner will use this information to construct the answer for $C_L^l(w)$ (recall that $C_L^l(w) = \{u \in \Sigma^{\leq l} \mid wu \in L\}$). Clearly, this modified version of $\mathcal{A}lg$ is a query learner algorithm that learns $\mathcal{C}$ using MQs. □

This theorem is basically saying that having an oracle that can return at once the answers for more than one MQ (one lBCQ contains the answer for $1 + |\Sigma| + \ldots + |\Sigma|^l$ MQs) does not increase the learnability power of the model (that is, the learning with MQs model). The result was somehow expected if we recall that time complexity issues are neglected in our analysis. Moreover, this allows us to talk about the model of learning with LBCQs in general, without specifying a given length bound. Therefore, we can talk about *LBCorQ*, the collection of all language classes $\mathcal{C}$ for which there exists an $l \geq 0$ and a query learner $\mathcal{A}lg$ such that $\mathcal{A}lg$ learns $\mathcal{C}$ using a finite number of l-bounded correction queries.

3.2 Learning with Edit Distance Correction Queries

Let us now investigate the model of learning with EDCQs. It is clear that any oracle answering EDCQs would implicitly give us the answer for the corresponding MQ, so *EditCorQ* trivially includes *MemQ*. In fact, the two learning models are equivalent:

Theorem 2. *EditCorQ = MemQ.*

Proof. Let us first show that having an MQ oracle allows us to compute the answer to an EDCQ using a finite number of MQs. Indeed, given a non-empty recursive language L and a string w in Σ^*, the following algorithm computes the value of $\textsc{Edc}_L(w)$ by asking only MQs.

Algorithm 1. An algorithm that computes $\textsc{Edc}_L(w)$ with an MQ oracle

```
1: input: $L, w$
2: ask the oracle if $w$ is in $L$
3: if the answer is Yes then
4:    output Yes
5: else
6:    while TRUE do
7:       $i := 1$
8:       for all $u$ such that $d(w,u) = i$ do
9:          ask the oracle if $u$ is in $L$
10:         if the answer is Yes then
11:            output $u$ and halt
12:         end if
13:      end for
14:      $i := i + 1$
15:   end while
16: end if
```

Clearly, Algorithm 1 terminates by outputting a string $u \in L$ such that there is no $v \in L$ with $d(w,v) < d(w,u)$. Note that for a given $w \in \Sigma^*$ and $i \in \mathbb{N}$ there are only a finite number of strings $v \in \Sigma^*$ such that $d(w,v) = i$, and there exists an algorithm who can generate all these strings (remember that we are not concerned with the complexity of the resulting algorithm - the only requirement is to return the answer after finite steps).

Now, if we take $\mathcal{C}$ to be a language class in *EditCorQ*, then there exists an algorithm *Alg* that learns $\mathcal{C}$ using EDCQs. *Alg* can be modified to use the MQ oracle instead of the EDCQ oracle to get the necessary answers as described above. We obtained an algorithm that learns $\mathcal{C}$ using MQs only, so *EditCorQ* $\subseteq$ *MemQ* which concludes our proof. □

3.3 The Global Picture

A complete picture displaying the relations between all discussed versions of query learning and Gold-style learning is obtained (see Figure 1).

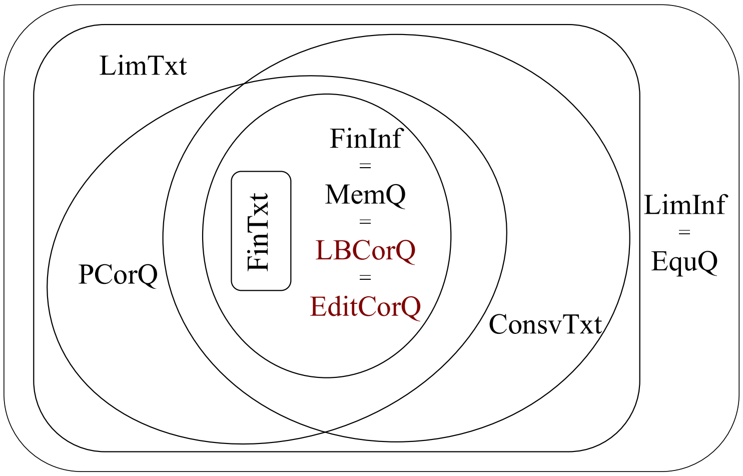

Fig. 1. A hierarchy of Gold-style and query learning models

As we have already mentioned, the results in this section are all about learning with queries where we do not take into account time complexity issues. So, what happens if we restrict to polynomial time learning? We will answer this question in the next section.

4 Polynomial Time Learning with Correction Queries

Although from a theoretical point of view it is important to know which language classes are inferable in finite time steps (see the proof of Proposition 3 for a relevant example), what matters in practice is the efficiency of the algorithms. In this section we investigate the relations between different types of CQs when complexity issues are taken into consideration. We will see that there are situations when, although two query types are equally powerful in the general case, the equality is not preserved under efficiency constraints. So far we know that learning with MQs is a strictly weaker model than learning with PCQs for both finite time algorithms [8] and polynomial ones [9]. We show by an example why one can not automatically generalize a result obtained in the general case to the restricted model of polynomial time learning.

Let us first recall that learning with EQs is strictly more powerful than learning with PCQs when ignoring time complexity: *PCorQ* is strictly included in *LimTxt* [8], and *LimTxt* is strictly included in *EquQ* [15]. Nevertheless, there exists a class of languages, namely the zero-reversible languages, that is polynomially learnable with PCQs [9] and not identifiable in polynomial time with EQs [16].

For a better comprehension of our results, let us introduce some notations. Let $\mathcal{C} = (L_i)_{i\geq 1}$ be an indexable class. We denote by *PolMemQ* the collection of all indexable classes $\mathcal{C}$ for which there exists a polynomial $p(\cdot)$ and an algorithm $\mathcal{A}lg$

that learns any language L in $\mathcal{C}$ in time $\mathcal{O}(p(\mathit{size}(L)))$ by asking a finite number of MQs. Similarly, *PolPCorQ*, *PollBCorQ* and *PolEditCorQ* are defined for the models of learning with PCQs, lBCQs and EDCQs, respectively.

4.1 Polynomial Time Learning with LBCQs

Let us first recall that there is basically no difference between a 0-BCQ and an MQ, so *Pol0BCorQ* = *PolMemQ*. Now, if we take l to be a fixed positive integer, then the following property holds.

Proposition 1. *Pol(l-1)BCorQ = PollBCorQ for any* $l \geq 1$.

Proof. Since one can easily extract the answer to an $(l-1)$BCQ from the corresponding lBCQ, it is clear that *Pol(l-1)BCorQ* is included in *PollBCorQ*. Let us now show that the other inclusion holds as well. Let $\mathcal{C}$ be an indexed family of languages in *PollBCorQ* and $\mathcal{A}lg$ a polynomial time algorithm that learns $\mathcal{C}$ with lBCQs. Note that for any language L over Σ, any $w \in \Sigma^*$ and any $l \geq 1$,

$$\begin{aligned} C_L^l(w) &= \{u \in \Sigma^{\leq l} \mid wu \in L\} \\ &= \{u \in \Sigma^{\leq l-1} \mid wu \in L\} \cup \{au \mid a \in \Sigma, u \in \Sigma^{l-1} \text{ and } wau \in L\} \\ &= C_L^{l-1}(w) \cup \{au \mid a \in \Sigma, u \in C_L^{l-1}(wa)\} \\ &= C_L^{l-1}(w) \cup \textstyle\bigcup_{a\in\Sigma} aC_L^{l-1}(wa). \end{aligned}$$

So, one can modify $\mathcal{A}lg$ such that instead of asking an lBCQ for the string w, to ask a finite number of $(l-1)$BCQs ($|\Sigma|+1$ queries to be precise) for the strings wa with a in $\{\lambda\} \cup \Sigma$. Clearly, the modified algorithm is still polynomial. Hence, *Pol(l-1)BCorQ* equals *PollBCorQ*.

We draw the conclusion that *PollBCorQ* = *PolMemQ* for any $l \geq 0$, and hence, we can talk about the model of polynomial learning with LBCQs in general, without specifying the length bound (we denote by *PolLBCorQ* the collection of all language classes $\mathcal{C}$ for which there exists an $l \geq 0$ such that $\mathcal{C}$ is in *PollBCorQ*). So, having the possibility to get answers for more than one MQ at once does not add any more learning power, even if we impose time restrictions.

4.2 Polynomial Time Learning with EDCQs

We continue the analysis done in Section 3.2 about the power of learning with EDCQs, this time by taking into account time complexity issues. We have seen that what happens in the general model does not necessary carry on to the polynomially bounded model. Let us recall the already known results:

- *MemQ* $\subsetneq$ *PCorQ* [8] and *PolMemQ* $\subsetneq$ *PolPCorQ* [9],
- *PCorQ* $\subsetneq$ *EquQ* [8] but *PolPCorQ* $\not\subseteq$ *PolEquQ* (see page 8 above, lines 5-8),
- *MemQ* = *LBCorQ* and *PolMemQ* = *PolLBCorQ*,
- *EditCorQ* = *MemQ*.

We show that in the case of EDCQs, the equality is not preserved.

Proposition 2. *PolMemQ* $\subsetneq$ *PolEditCorQ*.

Proof. Recall that $\text{EDC}_L(w)$ is Yes if and only if w belongs to L. Assume that $\mathcal{C}$ is a language class in *PolMemQ* and let *Alg* be a polynomial time algorithm that learns every L of $\mathcal{C}$ after asking a finite number of MQs. Obviously, the number of MQs asked while *Alg* is running with input L is also bounded by a polynomial, let us say $p(n)$, where n is the size of the target language L. If we modify *Alg* so that instead of asking the oracle a MQ for the string w, to ask an EDCQ for the same string, we obtain another algorithm Alg' which learns $\mathcal{C}$ with EDCQs (it just uses the information received from asking EDCQs to determine whether or not the given string is in the target language). The only thing left to be shown is that Alg' is still polynomial. But this is clear if we notice that Alg' performs at most $p(n)$ more operations than *Alg* (for each queried string w it compares $\text{EDC}_L(w)$ with Yes). Since Alg' is a polynomial time algorithm that learns $\mathcal{C}$ using EDCQs, we obtain that $\mathcal{C}$ is in *PolEditCorQ*.

Moreover, if $\mathcal{S} = (L_w)_{w\in\Sigma^*}$ is the class of singleton languages $L_w = \{w\}$ over the alphabet Σ, then $\mathcal{S}$ can be used as a separating language class:

– $\mathcal{S} \notin$ *PolMemQ* since any algorithm that learns $\mathcal{S}$ using MQs might need to ask $|\Sigma| + |\Sigma|^2 + \ldots + |\Sigma|^{|w|}$ MQs in the worst case when running on input language L_w;
– $\mathcal{S} \in$ *PolEditCorQ* since there exists a very simple EDCQ algorithm for this class. Indeed, it is enough to ask one EDCQ for an arbitrarily chosen string w. The algorithm just outputs w, if the oracle's answer is Yes, and w' if the answer returned by the oracle is the string w'. The algorithm described above is clearly polynomial in the size of the target language. □

So learning with EDCQs is strictly more powerful than leaning with MQs when we restrict to efficient algorithms.

4.3 The Global Picture

In the end of the previous section we exhibited a complete picture of the relations existing between several models of learning with CQs and other learning models. We have seen that when we neglect time complexity issues learning with LBCQs and EDCQs is basically the same as learning with MQs, whereas PCQs are the only ones adding some power to the model.

When we restrict to polynomial time algorithms, things are changing. And although having an LBCQs oracle does still not improve on the learnability power with respect to the MQ learning model, an EDCQ oracle or a PCQ oracle does. It is less clear what relation is between learning with EDCQs and learning with PCQs when we restrict to efficient algorithms.

Let us first notice that the class of singleton languages is in *PolPCorQ* $\cap$ *PolEditCorQ*. Moreover, we argue that there are languages polynomial time learnable with PCQs for which there is no efficient EDCQ algorithm.

Proposition 3. *PolPCorQ*$\setminus$*PolEditCorQ* $\neq \emptyset$.

Proof. From [9] we know that the class of k-reversible languages is in *PolPCorQ* and not in *MemQ*. But *MemQ* $=$ *EditCorQ* by Theorem 2 (recall we mentioned that sometimes theoretical results like this one might be useful), so k-reversible languages are not identifiable in finite time steps with EDCQs, and hence, they can not be polynomial time learnable with EDCQs either. □

To complete the picture, we would like to be able to say if there are language classes polynomial time learnable with EDCQs for which there is no efficient PCQ algorithm. We conjecture that the two classes *PolEditCorQ* and *PolPCorQ* are incomparable (see Figure 2).

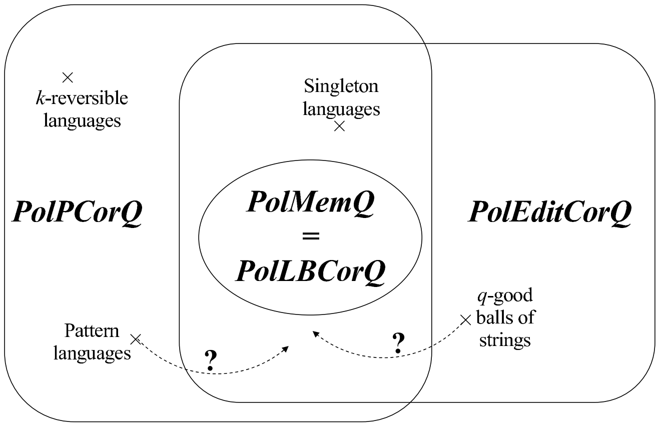

Fig. 2. Different types of correction queries

Our candidate for showing that $PolEditCorQ \setminus PolPCorQ \neq \emptyset$ is a subset of the class $\mathcal{B}_\Sigma = (B_r(w))_{w\in\Sigma^*,r\in\mathbb{R}}$, where $B_r(w) = \{v \in \Sigma^* \mid d(v,w) \leq r\}$ is a *ball of center w and radius r*. The authors of [4] show that for the so-called *q-good balls* (the balls $B_r(w)$ for which there exists a polynomial $q(\cdot)$ such that the radius r is no longer than $q(|w|)$), there exists an EDCQ algorithm which runs in polynomial time in the size of (the representation of) the language. So, what is left to be shown is that PCQs are not very useful in the process of leaning this particular class.

A slightly different type of EDCQ is used in [5] for learning the class of pattern languages: if the queried string is not in the target language, then the oracle returns the positive example with a smallest distance from the queried string and previously not used in the learning process. Moreover, preference is given to correcting strings of the same length, if any, and among those having the same length, the smallest one with respect to the lexicographical order is returned. Kinber describes in [5] an efficient algorithm that learns any pattern language with this modified type of EDCQ. We strongly believe that this requirement (i.e., that the oracle must not return as a correction any of the strings which appeared before) is actually mandatory, that is, there is no algorithm which can learn the class of pattern languages with regular EDCQs. We leave this as an open problem.

Acknowledgements

Special thanks to Colin de la Higuera for valuable advices, and to the anonymous reviewers for their remarks and suggestions.

References

1. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation 75(2), 87–106 (1987)
2. Becerra-Bonache, L., Dediu, A.H., Tîrnăucă, C.: Learning DFA from correction and equivalence queries. In: Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E. (eds.) ICGI 2006. LNCS (LNAI), vol. 4201, pp. 281–292. Springer, Heidelberg (2006)
3. Tîrnăucă, C.: Learning reversible languages from correction queries only, `http://grlmc-dfilrom.urv.cat/grlmc/PersonalPages/cristina/publications.htm`
4. Becerra-Bonache, L., de la Higuera, C., Janodet, J.C., Tantini, F.: Learning balls of strings with correction queries. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 18–29. Springer, Heidelberg (2007)
5. Kinber, E.: On learning regular expressions and patterns via membership and correction queries (manuscript, 2008)
6. Jain, S., Kinber, E.B.: One-shot learners using negative counterexamples and nearest positive examples. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 257–271. Springer, Heidelberg (2007)
7. Tîrnăucă, C.I., Tîrnăucă, C.: Learning regular tree languages from correction and equivalence queries. Journal of Automata, Languages and Combinatorics 12(4), 501–524 (2007)
8. Tîrnăucă, C., Kobayashi, S.: A characterization of the language classes learnable with correction queries. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 398–407. Springer, Heidelberg (2007)
9. Tîrnăucă, C., Knuutila, T.: Polynomial time algorithms for learning k-reversible languages and pattern languages with correction queries. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 272–284. Springer, Heidelberg (2007)
10. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. Journal of ACM 21(1), 168–173 (1974)
11. Angluin, D.: Inference of reversible languages. Journal of the ACM 29(3), 741–765 (1982)
12. Zeugmann, T., Lange, S., Kapur, S.: Characterizations of monotonic and dual monotonic language learning. Information and Computation 120(2), 155–173 (1995)
13. Zeugmann, T.: Inductive inference and language learning. In: Cai, J.-Y., Cooper, S.B., Li, A. (eds.) TAMC 2006. LNCS, vol. 3959, pp. 464–473. Springer, Heidelberg (2006)
14. Gold, E.M.: Language identification in the limit. Information and Control 10(5), 447–474 (1967)
15. Lange, S., Zilles, S.: Formal language identification: query learning vs. Gold-style learning. Information Processing Letters 91(6), 285–292 (2004)
16. Angluin, D.: Negative results for equivalence queries. Machine Learning 5(2), 121–150 (1990)