Translit

Распространенным примером искажающего преобразования является транслит (translit).

Translit - это сокращение от *transliteration* - процедуры записи текстов одного языка символами другого. Транслиты часто возникают при смешении племен с различной письменностью, как результат миграции, изгнания или колонизации, и просто как заимствования (кальки) из других языков (амбар, батон, вокзал, директор, жакет, кран, магазин, сарай, трактор, футбол, ярлык итд.)

В качестве культурно и исторически значимого примера транслита, можно назвать идиш - один из жаргонов европейских евреев (возникший после их изгнания из Испании в 1492 г.), в котором слова германского диалекта записывались буквами древнееврейского алфавита. Другим примером может служить воровское арго российского криминального мира ("феня"), в котором русскими буквами передавались многочисленные заимствования из латинского, древнееврейского, французского и венгерского (мент, параша, ксива, малина, порода, хипеш, бан, хаза итд.)

В теории информации транслит столь же удобен, как дрозофила в генетике. Едва ли не все основные понятия могут быть выведены из рассмотрения этой модели коммуникации.

Значимость и популярность транслитов усилилась с развитием (почтовых) средств связи с ограниченным (обычно, только латинским) алфавитом. Например, в GSM-стандарте мобильной связи символы национальных алфавитов могут передаваться только в двухбайтной кодировке (UCS2). Как следствие, кириллические SMS-сообщения обходятся вдвое дороже, чем в транслите.

Возможны ровно три варианта отображения (маппирования) символов одного языка на другой, в зависимости от соотношения мощности (размеров) их алфавитов.

1. Мощность исходного алфавита меньше мощности конечного.

Например, 27-ми буквам древнееврейского языка (22 + 5 "софитов") несложно найти кириллические (33 символа) эквиваленты, тем более, что начертания некоторых кириллических букв (не имевших греческих аналогов) именно оттуда и были заимствованы (например, "заин" -> "з", "шин" -> "ш").

2. Мощности алфавитов совпадают.

Решение тривиально.

3. Мощность исходного алфавита больше мощности конечного.

Например, 33 символа современной кириллицы необходимо отобразить в 26 символов латинского алфавита.

Классическое решение заключается в конструировании многосимвольных супербукв

(лигатур). Например, дифтонги английского языка "ae", "ph" и "th" используются для передачи отсутствующих в английском греческих звуков (букв). Кажется, почти во всех алфавитных языках используется удвоение согласной для передачи (звонкого) ударного звука ("суббота"). Некоторые лигатуры возникают для передачи оттенков звучания (неалфавитных звуков). Например, старославянское "io" ("іолка") или современное "йо" ("йод"), английское "oo" ("Oops!", "book", "moon").

При использовании супербукв (лигатур) возникает непростая проблема их идентификации.

Каким образом декодер должен решить, что означает прочитанный им из потока символ - ординарную букву используемого алфавита или первый символ многосимвольной лигатуры (супербуквы)?

Здесь, опять-таки, возможны три варианта.

1. Значение символа зависит от контекста - то есть от того, сколько и каких символов встретилось до и/или после данного. Этот вариант требует от декодера некоторого интеллекта и определенного объема буферной памяти, достаточной для хранения (грамматически корректной) фразы, используемой при решении. Практически, это означает встраивание в декодер словаря (спелчекера).

Аналогом этой ситуации в естественных языках являются омонимы. Например, невозможно указать значение таких, отдельно стоящих, слов как "ключ" или "коса" - требуется знание контекста.

Если на интернет-форуме вы читаете написанный транслитом текст, то это, как раз вышеописанный вариант. Знание контекста и (встроенный) словарь общей лексики позволяют вам "on-the-fly" декодировать сообщение.

Мне не попадались контекстно-зависимые алгоритмы декодеров транслита, но использование встроенного словаря - классика схем компрессии данных.

2. Назначение одного или нескольких символов алфавита в качестве <ESC>-символов, трактуемых, как имеющих специальное значение.

Классическое решение, уходящее корнями в начало письменности. Например, на дорогом пергаменте текст писался без промежутков (пробелов). Заглавные (прописные) буквы и абзацные отступы - изобретение позднейшего времени. Для облегчения чтения (декодирования) и опознания начала слов модифицировалось написание отдельных букв, в зависимости от того, в какой позиции слова они находились. В некоторых стилях арабской вязи буквы могут иметь до четырех различных вариантов начертания: отдельно стоящая буква, начальная буква слова, конечная буква слова и лигатура, заменяющая сочетание нескольких букв. В иврите пять согласных ("софиты") меняют начертание, если находятся в конечной позиции слова. В английском, по крайней мере, одна буква использовалась таким образом ("i" -> "y").

Обычно, <ESC>-символ помещается в начале маркируемой последовательности. По крайней мере, так, обычно, делается в схемах компрессии данных и сотнях других применений.

Однако, в транслите ("Twoletters", "Volapuyk" и "Universal 2") <ESC>-символы последовательно используются в качестве суффикса для маркируемого символа.

3. "Common context" - использование общей модели. "Высокие договаривающиеся стороны" - оба кодер и декодер используют фиксированный список супербукв (лигатур). Этот вариант выглядит повторением первого, но это не так - взамен знания полной грамматики языка в первом случае (для отбраковки недопустимых языком конструкций), теперь требуется только знание (простейшей) грамматики построения супербукв. Практически, это наиболее распространенный вариант.

Декодер должен использовать "жадный" ("greedy") алгоритм для конструирования супербукв, остаток текста рассматривается при этом как односимвольная транслитерация.

Технически (и именно так устроены многие программы транслитерации), декодирование может быть выполнено в две фазы (в точности инверсные кодированию):

1. Поиск/замена всех многосимвольных последовательностей (лигатур).
2. XLAT - byte-2-byte трансляция полученного на фазе 1 текста.

Быстрый поиск/замену предлагает большинство распространенных библиотек работы со строками и эти алгоритмы относительно просты и более эффективны, чем алгоритмы конструирования конечного автомата для разбора регулярных выражений (ReExp). Конечно, при этом теряется возможность обработки особых случаев (буква в начале, конце, середине слова итд).

В сети Интернет представлено, по крайней мере, с дюжину различных многосимвольных схем транслитерации, лучшая из которых, IMHO, ***Simplified***. Вариант Покровского (***EuroTex-92***) совершенно непригоден для чтения. ***LOC*** (*Library of Congress*) провальна во всех отношениях (на самом деле, приведена ее устоявшаяся аппроксимация, без акцентированных символов), ***ГОСТ 16876-71*** (которым никто никогда не пользуется) однозначно декодируется, но уродует квотированный текст, ***Telegram*** "съедает" буквы 'Ъ' и 'Ь', все остальные просто ужасны. Я слегка модифицировал ***Simplified*** и добавил этот вариант под названием "***Modern***". Я также сконструировал еще одну схему: ***Funny*** - смесь традиций транслита и хакерской графики. Обе схемы, в отличие от многих других, однозначно декодируемы.