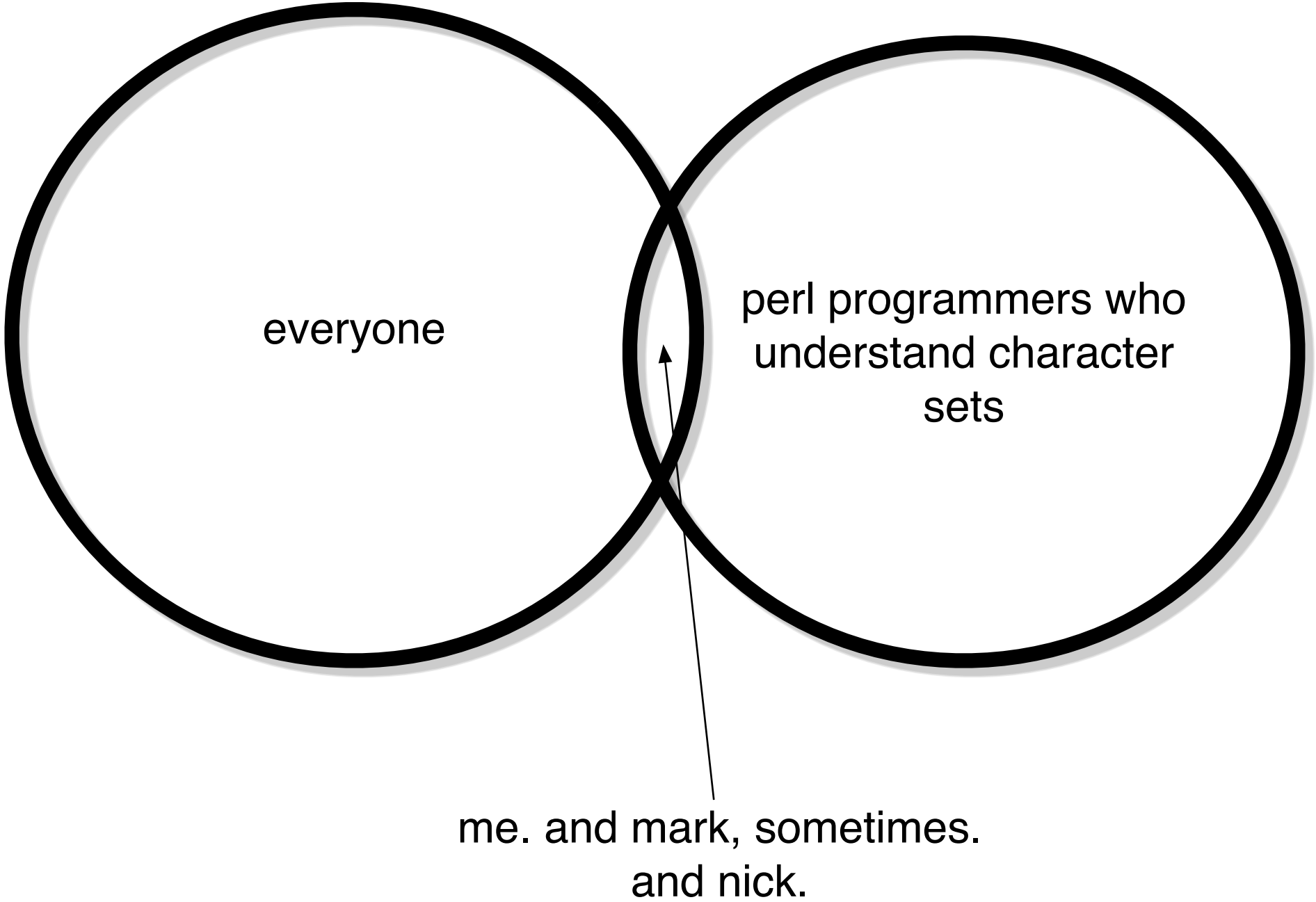
"why perl â¤ utf-8"

also (bonus!)

"why OmniGraffle is not a replacement for Powerpoint"


everyone
perl programmers who understand character sets
me. and mark, sometimes.
and nick.

10010110

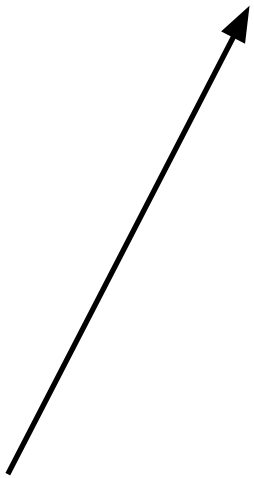
byte

character

a sequence of bytes

010101 01101100 1101011 00110100

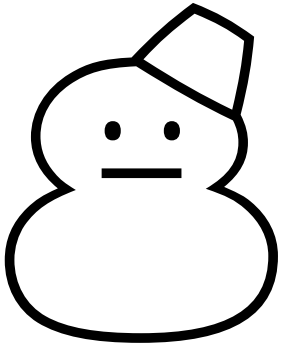
a sequence of characters



a sequence of characters

0010101 01101100 1101011 001101

$\neq$

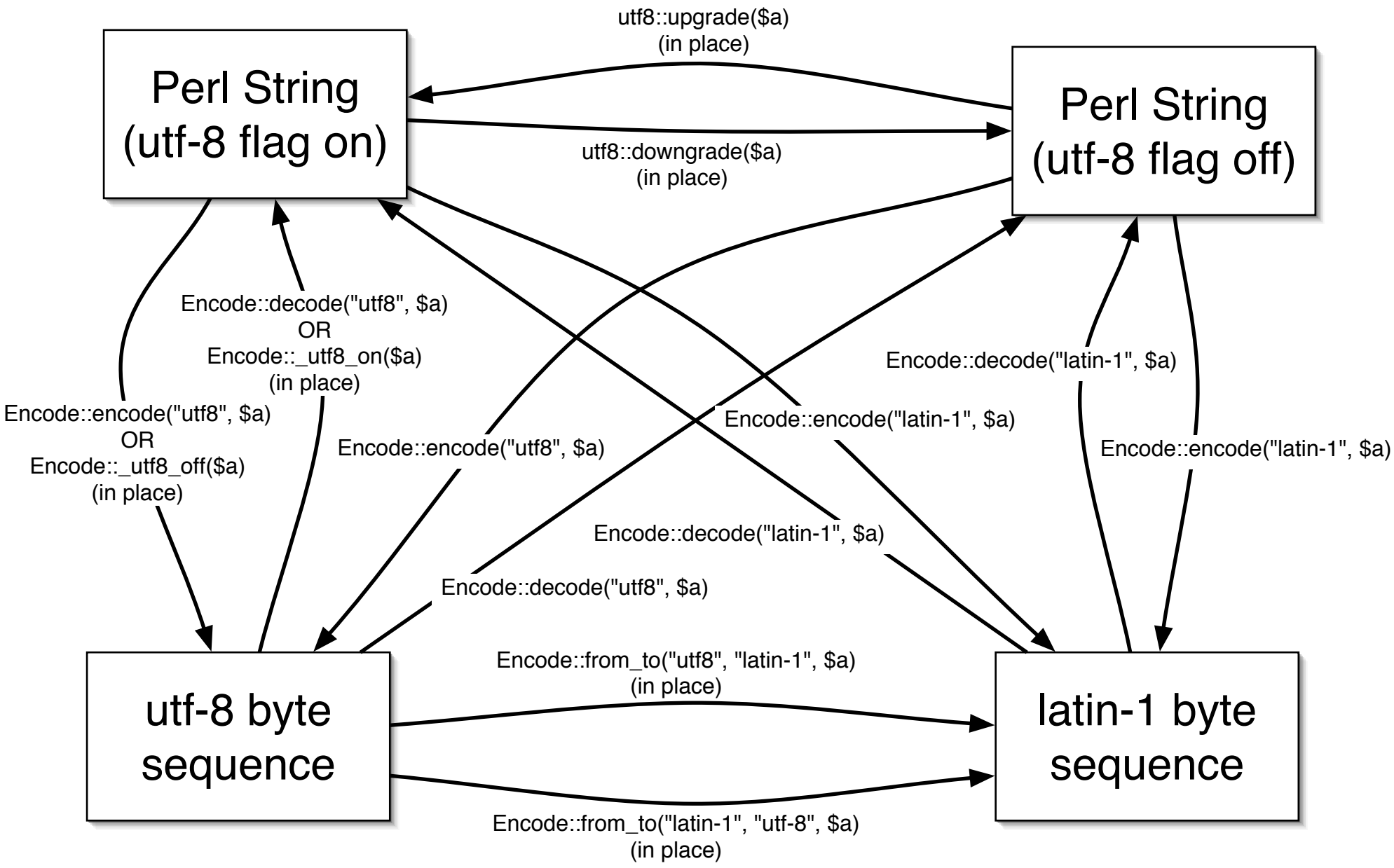


11100010 10011000 10000011

+

"that's utf-8"

⇨


Perl String
(utf-8 flag on)
Perl String
(utf-8 flag off)
utf-8 byte
sequence
latin-1 byte
sequence
utf8::upgrade($a)
(in place)
utf8::downgrade($a)
(in place)
Encode::decode("utf8", $a)
OR
Encode::_utf8_on($a)
(in place)
Encode::encode("utf8", $a)
OR
Encode::_utf8_off($a)
(in place)
Encode::encode("utf8", $a)
Encode::decode("utf8", $a)
Encode::decode("latin-1", $a)
Encode::encode("latin-1", $a)
Encode::decode("latin-1", $a)
Encode::encode("latin-1", $a)
Encode::from_to("utf8", "latin-1", $a)
(in place)
Encode::from_to("latin-1", "utf-8", $a)
(in place)

latin-1 byte sequence

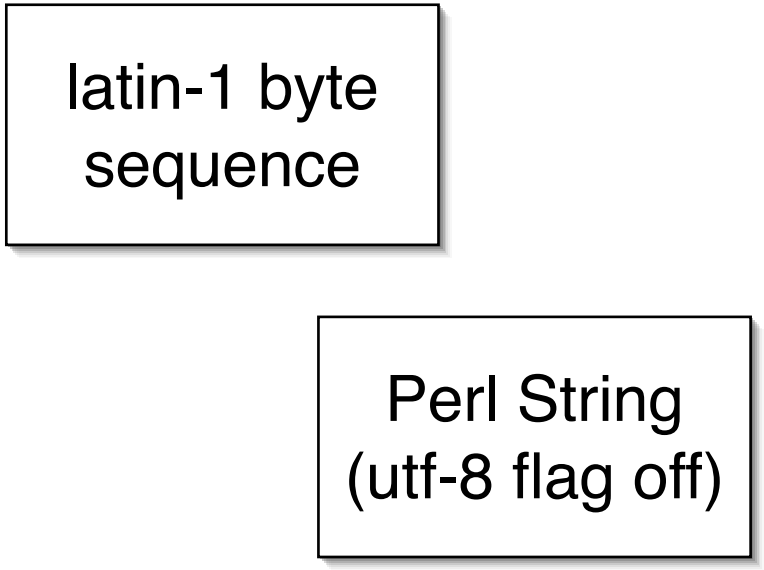
bytes = code points = characters

everything Just Works

Perl String
(utf-8 flag off)

bytes = code points = characters

everything Just Works

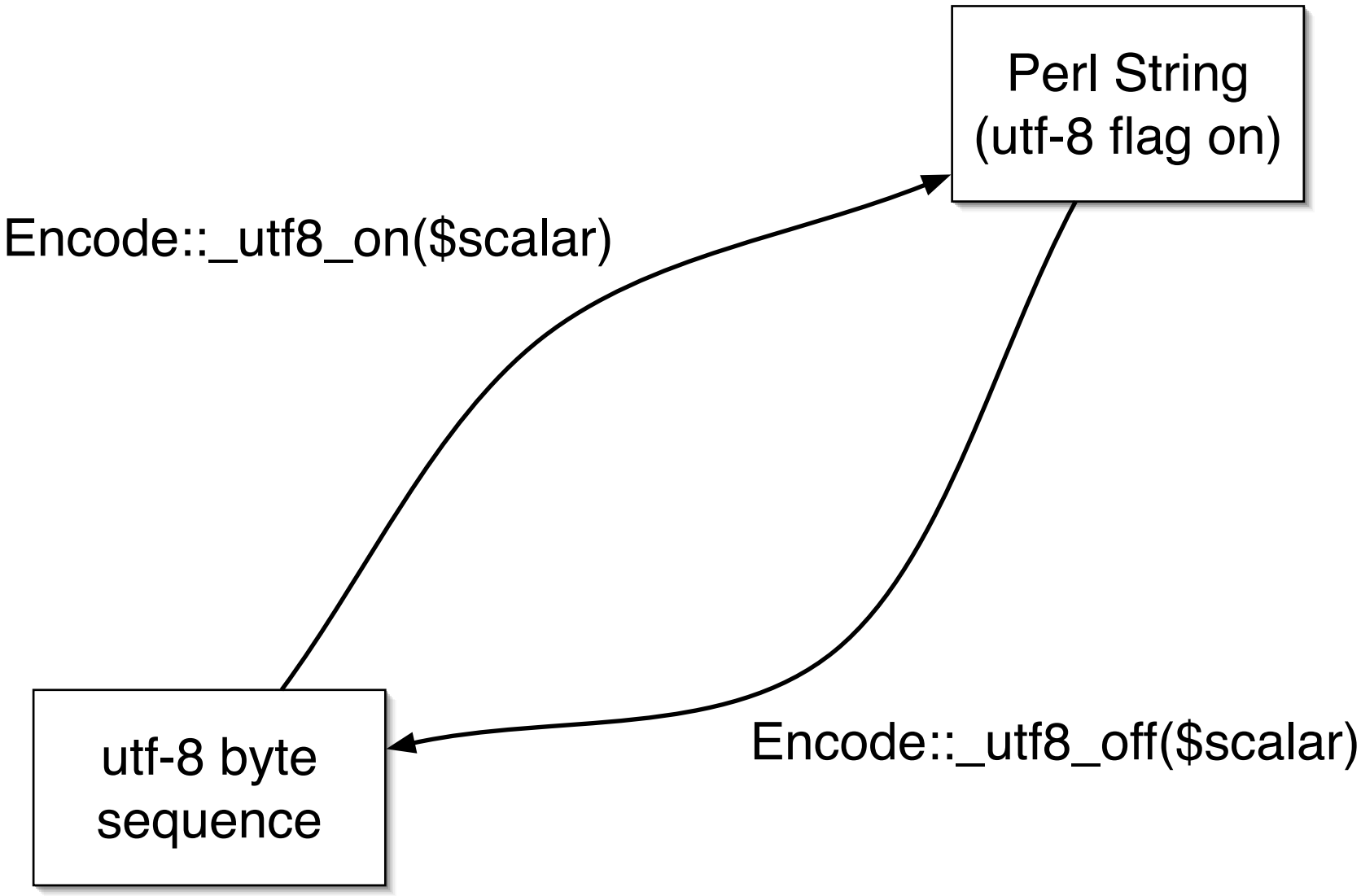

latin-1 byte sequence
Perl String (utf-8 flag off)

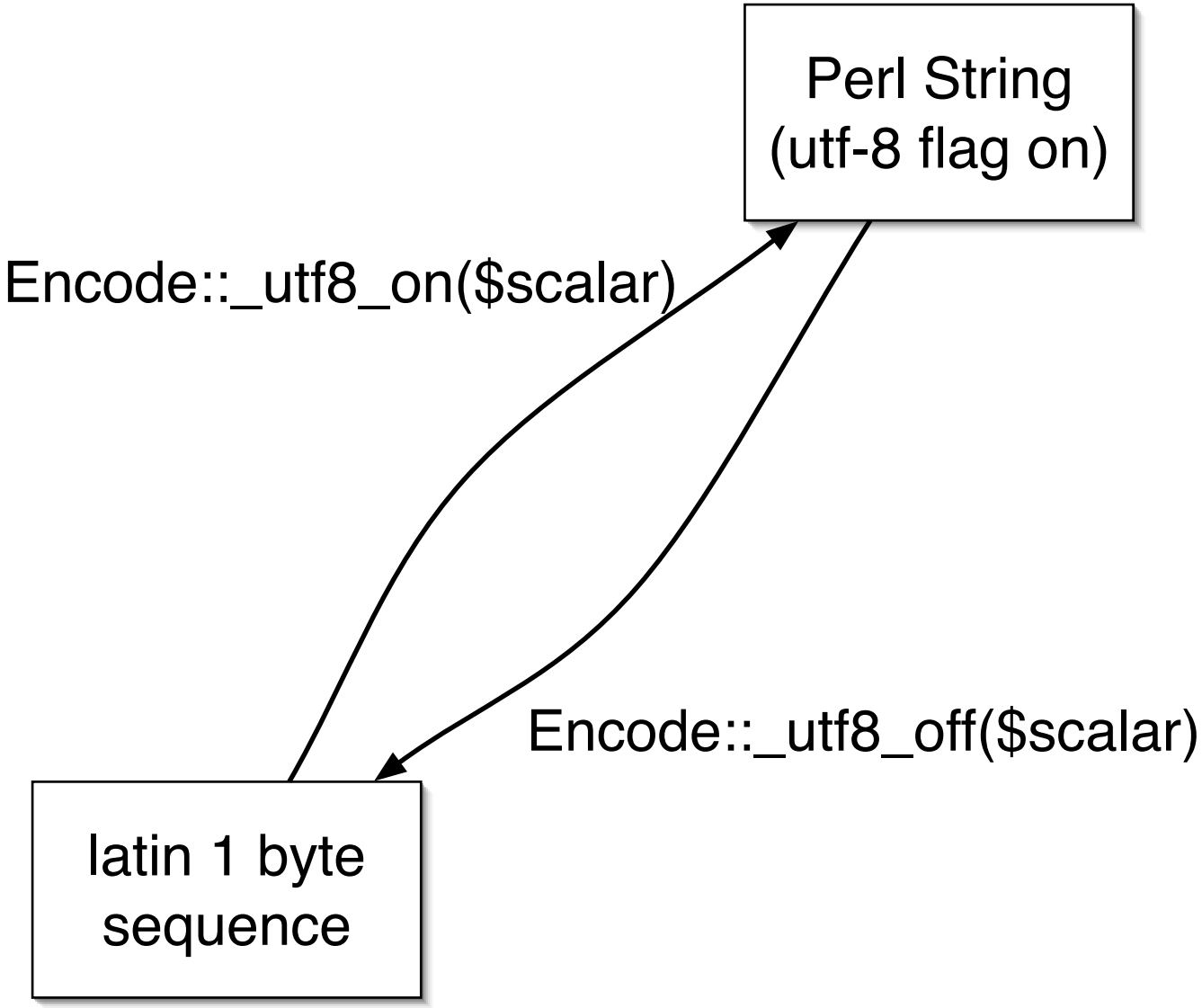
utf-8 byte sequence

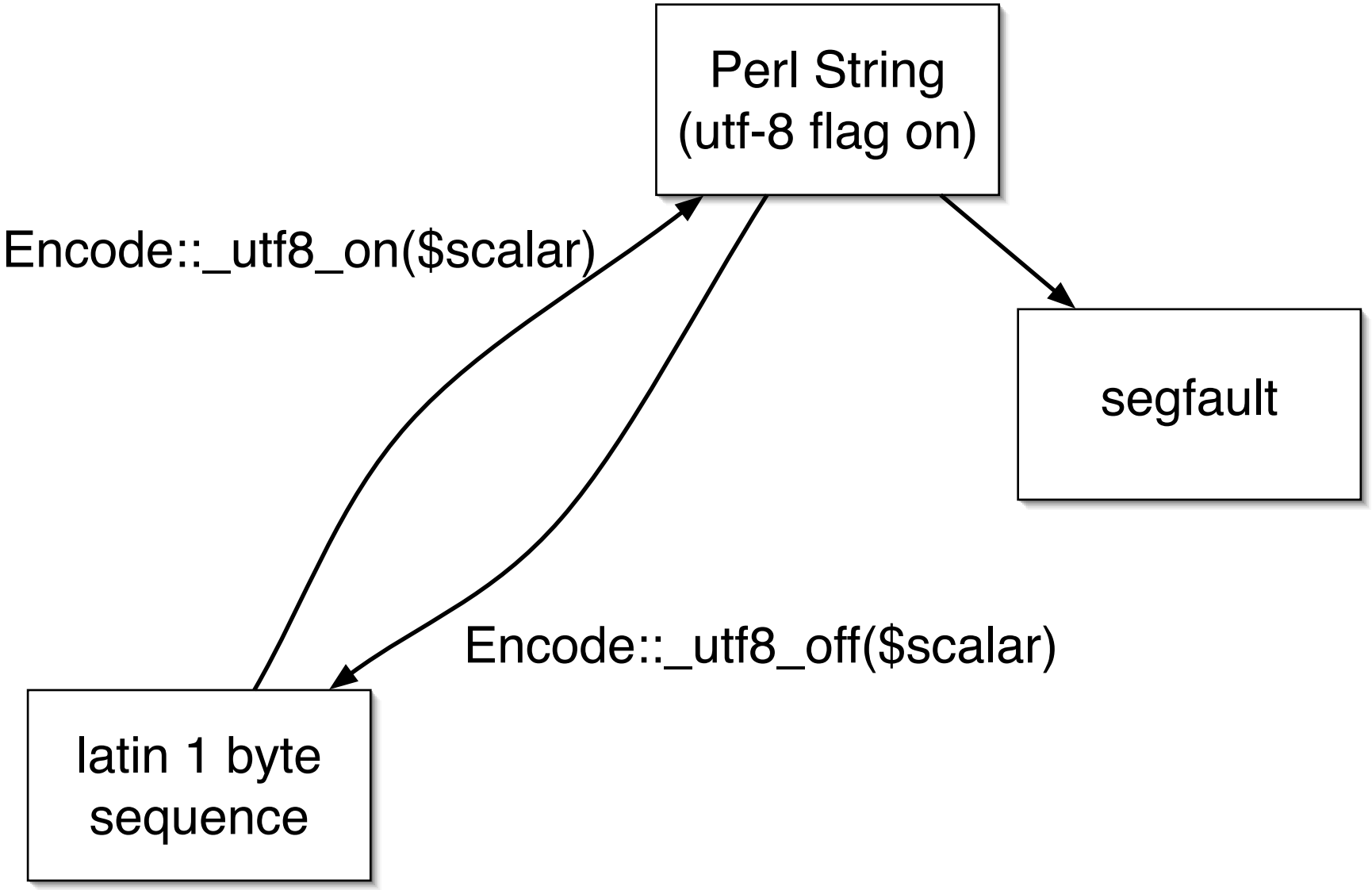
This is a sequence of bytes

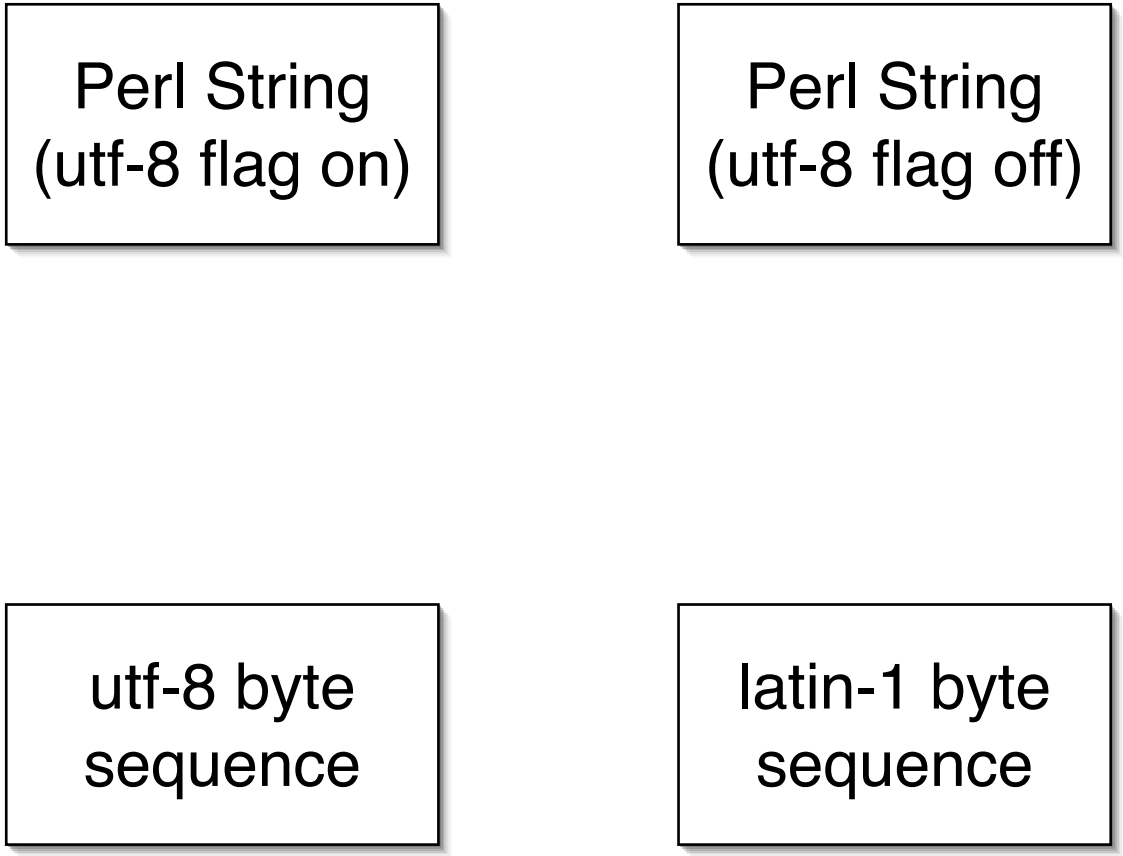
Perl String
(utf-8 flag on)

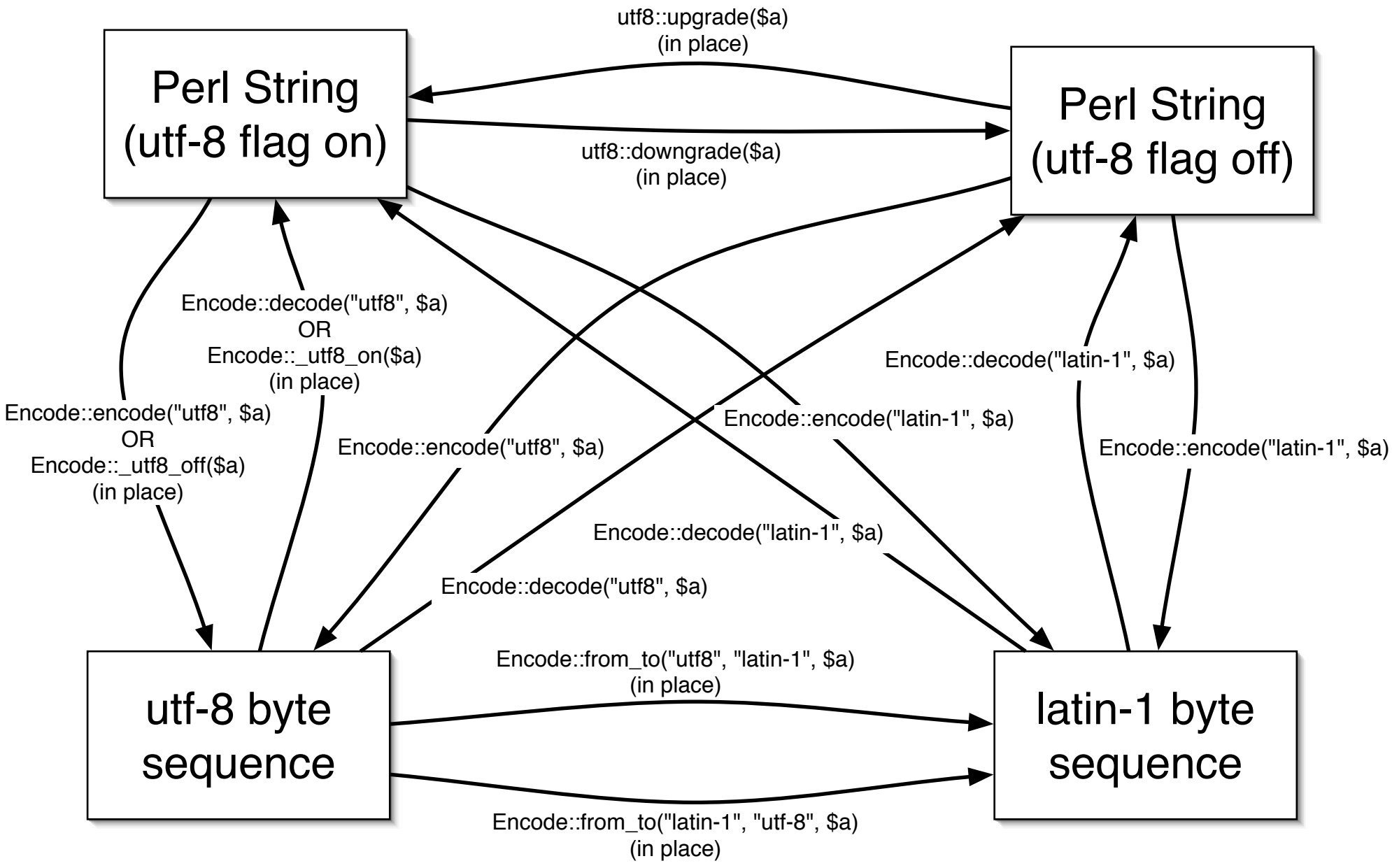
This is a sequence of characters

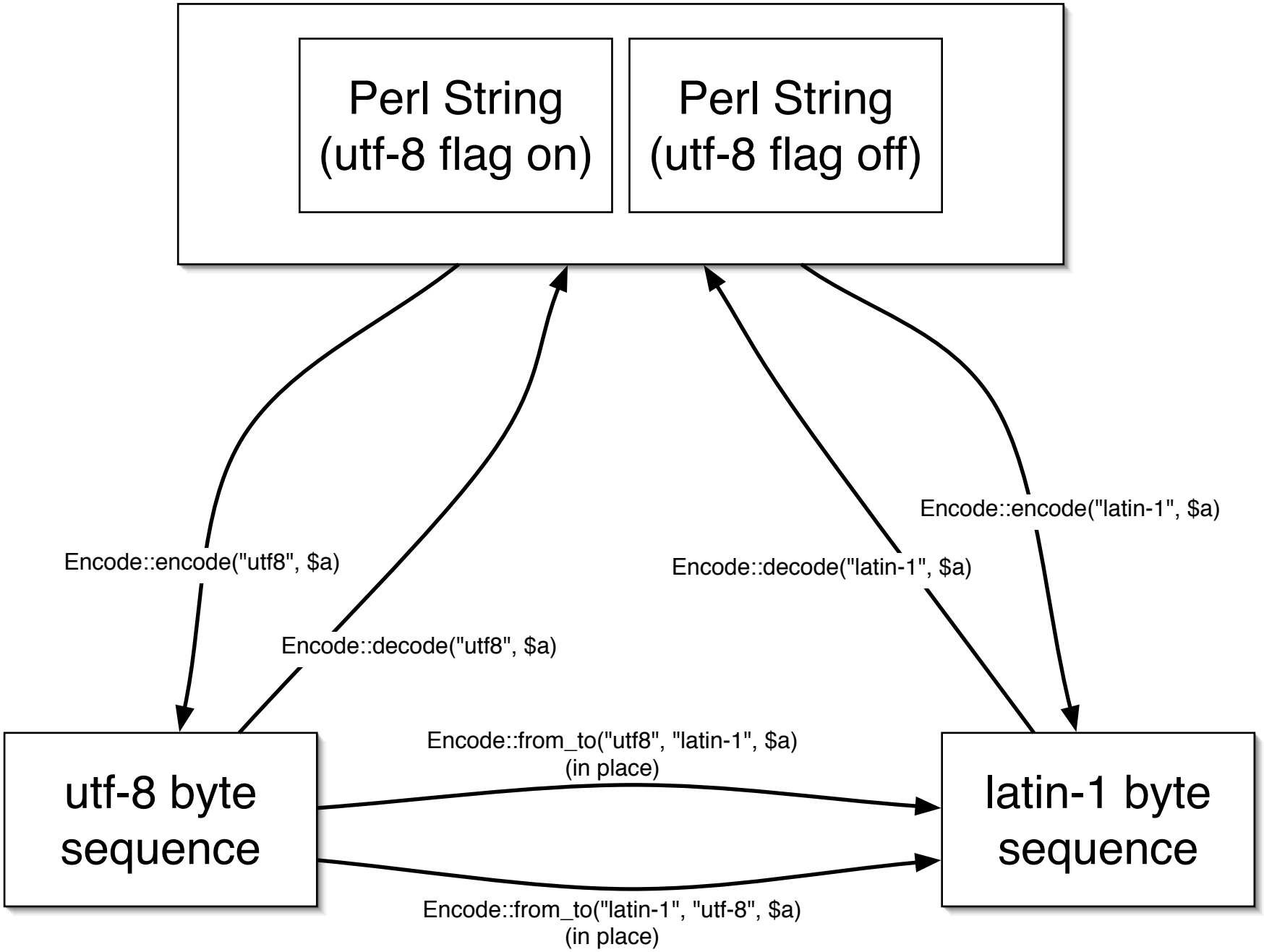

Perl String
(utf-8 flag on)
Encode::_utf8_on($scalar)
utf-8 byte
sequence
Encode::_utf8_off($scalar)

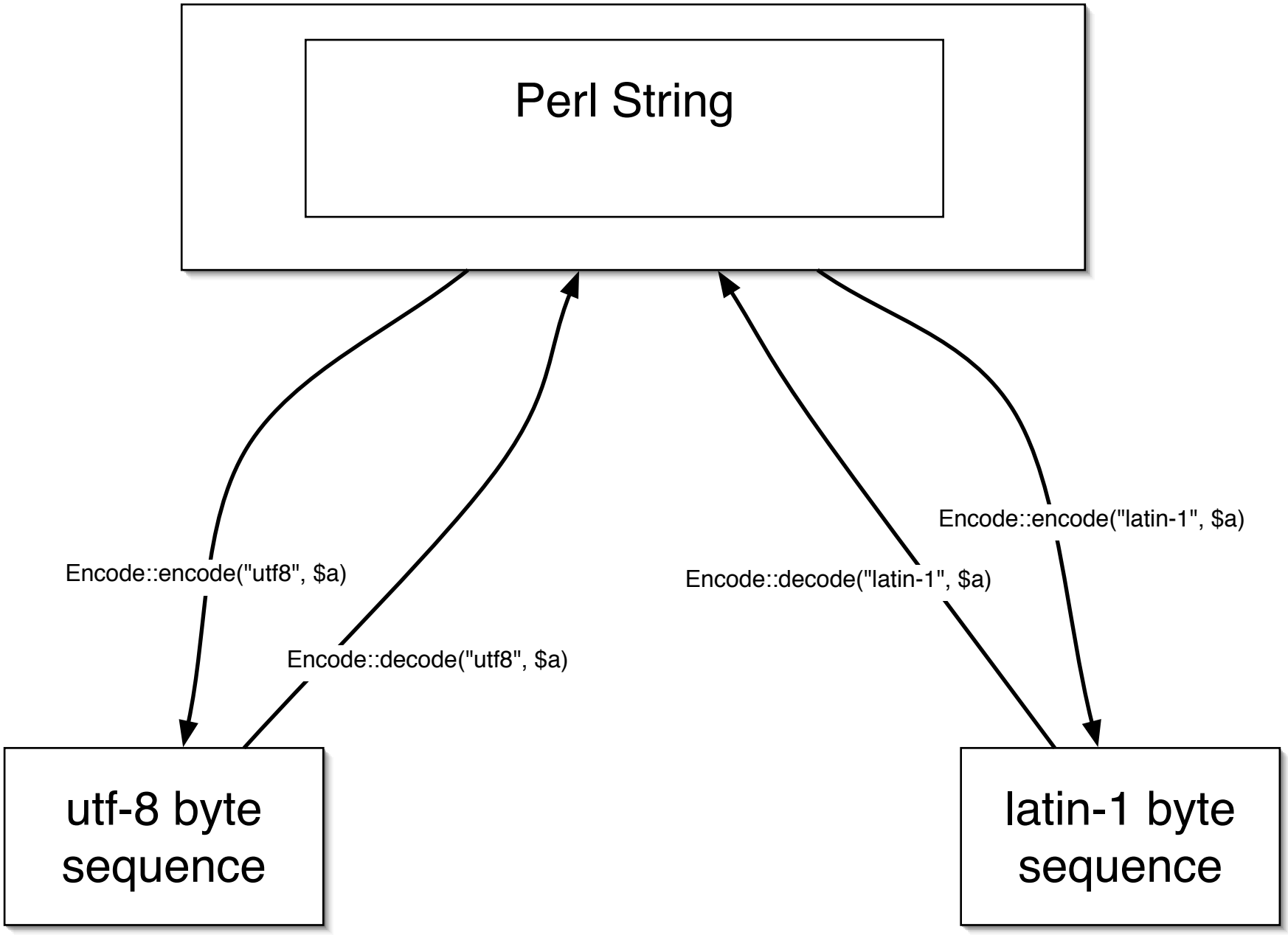

Perl String
(utf-8 flag on)
Encode::_utf8_on($scalar)
Encode::_utf8_off($scalar)
latin 1 byte
sequence


Perl String
(utf-8 flag on)
Encode::_utf8_on($scalar)
segfault
Encode::_utf8_off($scalar)
latin 1 byte
sequence


Perl String
(utf-8 flag on)
Perl String
(utf-8 flag off)
utf-8 byte
sequence
latin-1 byte
sequence


Perl String
(utf-8 flag on)
Perl String
(utf-8 flag off)
utf-8 byte
sequence
latin-1 byte
sequence
utf8::upgrade($a)
(in place)
utf8::downgrade($a)
(in place)
Encode::decode("utf8", $a)
OR
Encode::_utf8_on($a)
(in place)
Encode::encode("utf8", $a)
OR
Encode::_utf8_off($a)
(in place)
Encode::encode("utf8", $a)
Encode::decode("latin-1", $a)
Encode::encode("latin-1", $a)
Encode::decode("latin-1", $a)
Encode::encode("latin-1", $a)
Encode::decode("latin-1", $a)
Encode::decode("utf8", $a)
Encode::from_to("utf8", "latin-1", $a)
(in place)
Encode::from_to("latin-1", "utf-8", $a)
(in place)


Perl String
(utf-8 flag on)
Perl String
(utf-8 flag off)
Encode::encode("utf8", $a)
Encode::decode("utf8", $a)
Encode::decode("latin-1", $a)
Encode::encode("latin-1", $a)
utf-8 byte
sequence
latin-1 byte
sequence
Encode::from_to("utf8", "latin-1", $a)
(in place)
Encode::from_to("latin-1", "utf-8", $a)
(in place)


Perl String
Encode::encode("utf8", $a)
Encode::decode("utf8", $a)
Encode::decode("latin-1", $a)
Encode::encode("latin-1", $a)
utf-8 byte sequence
latin-1 byte sequence

```
$bytes = Encode::encode( 'encoding', $chars )
```

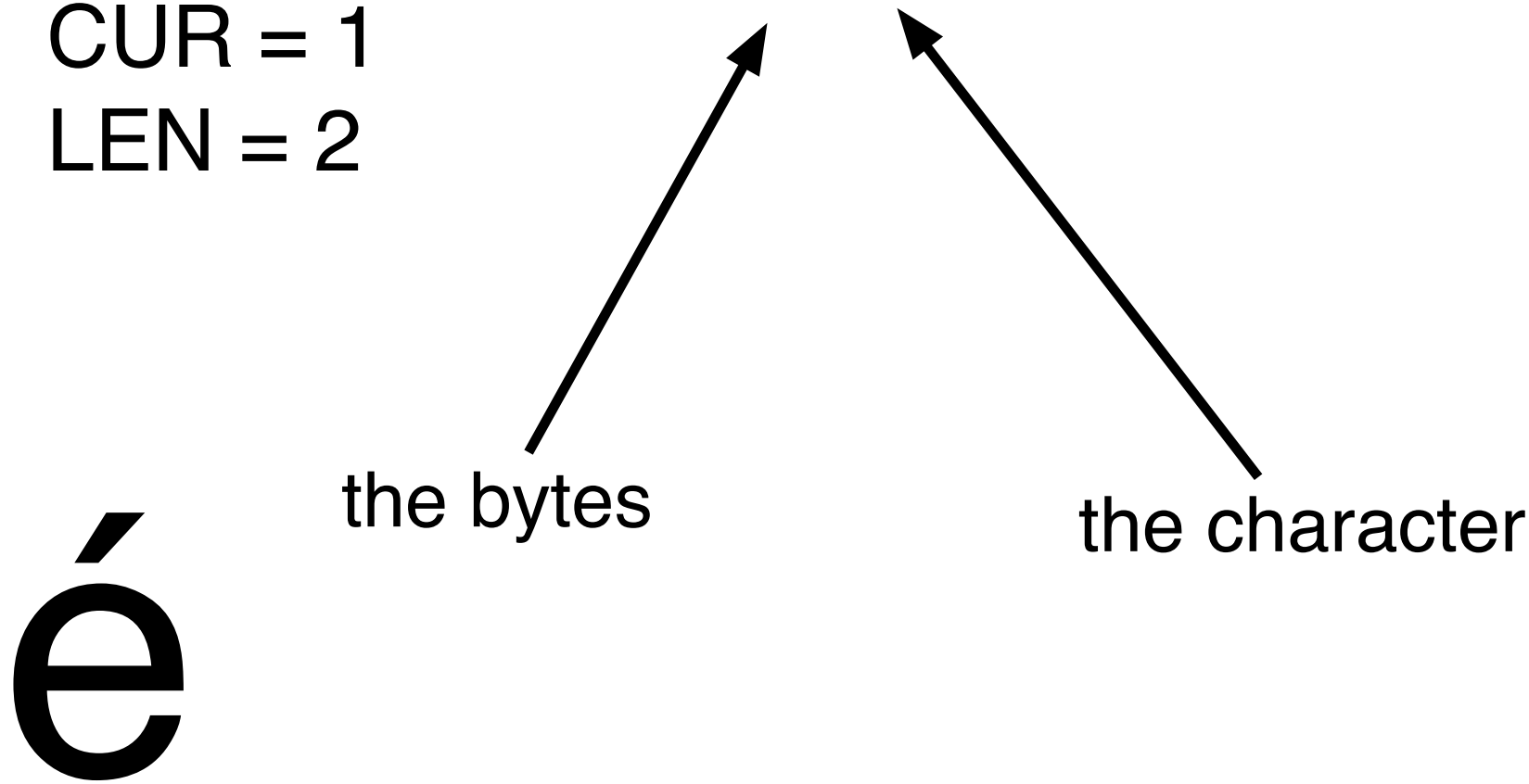
```
$chars = Encode::decode( 'encoding', $bytes )
```

use Devel::Peek

not very nice

XS

SV = PV(0x8131020) at 0x811d234
REFCNT = 1
FLAGS = (POK,READONLY,pPOK)
PV = 0x812a9c8 "\351"\0
CUR = 1
LEN = 2



```
SV = PV(0x811d470) at 0x8127c38
  REFCNT = 1
  FLAGS = (POK,pPOK,UTF8)
  PV = 0x8122ee8 "\303\251"\0 [UTF8 "\x{e9}"]
  CUR = 2
  LEN = 3
```

the bytes

the character

é

DBD::mysql

2 approaches

right

Encode::encode

Encode::decode

fast

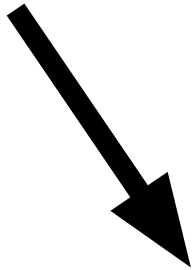
Encode::_utf8_on

the real correct approach

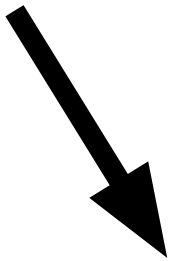
DBD::Pg

XML

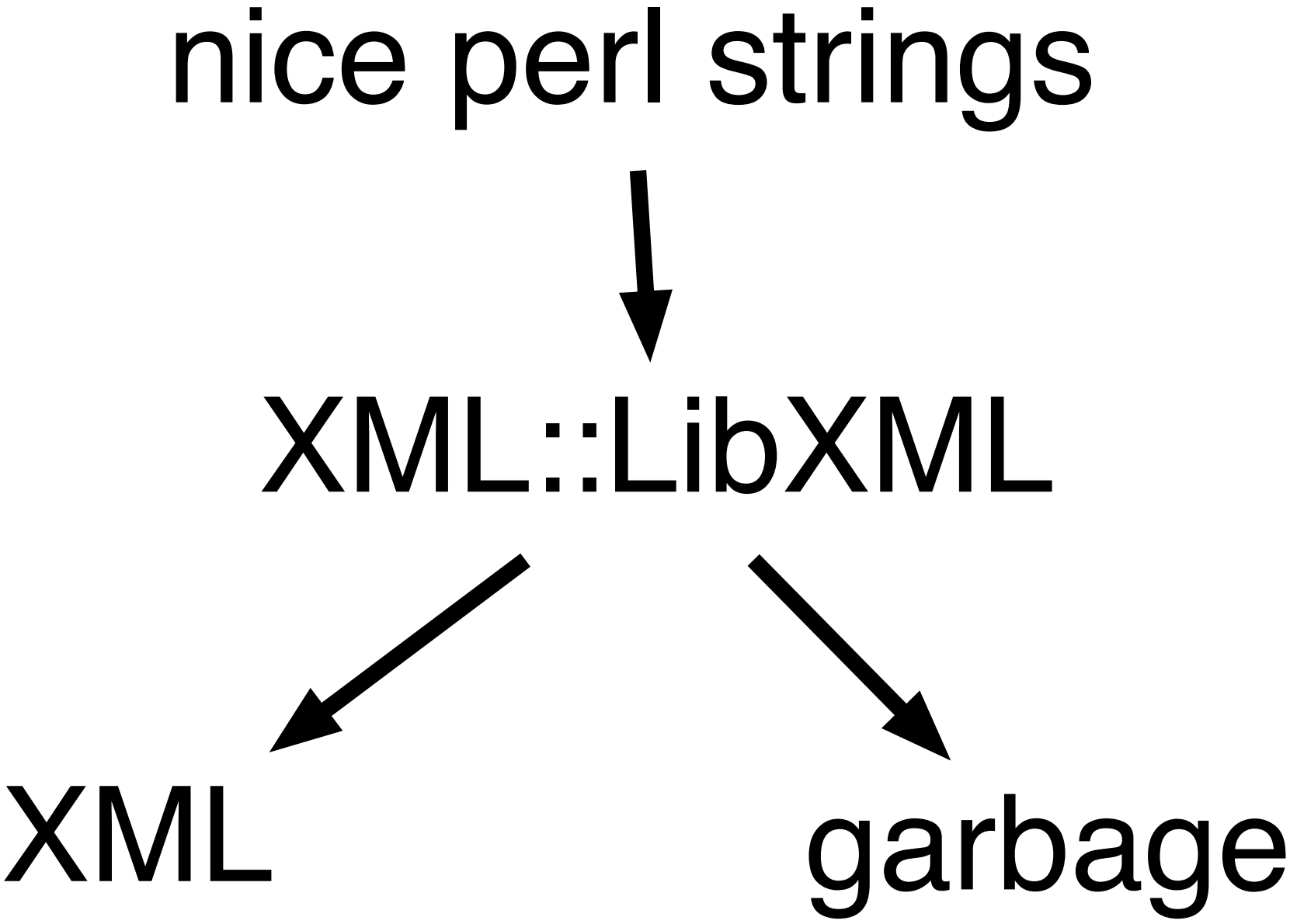
XML



XML::LibXML



nice perl strings


nice perl strings
XML::LibXML
XML
garbage

use java

there are very expensive
courses you can go to