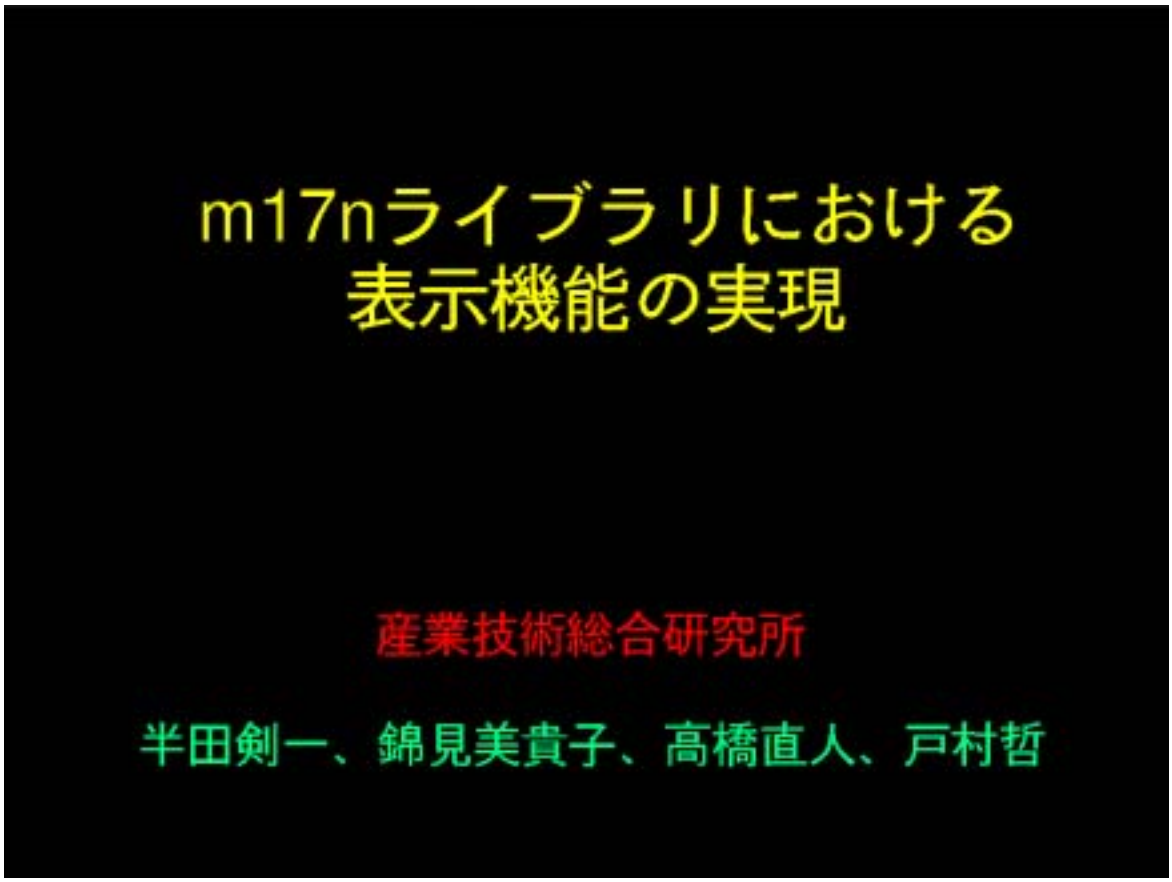
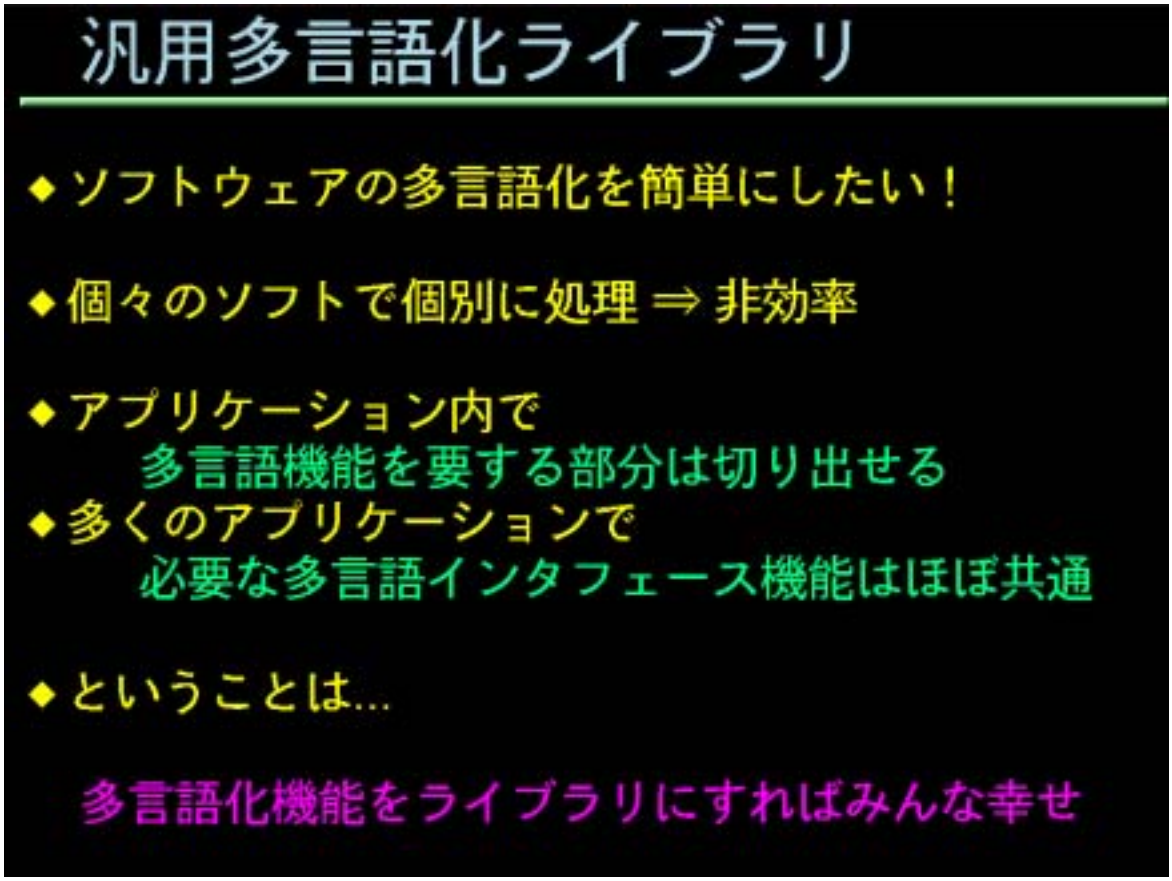
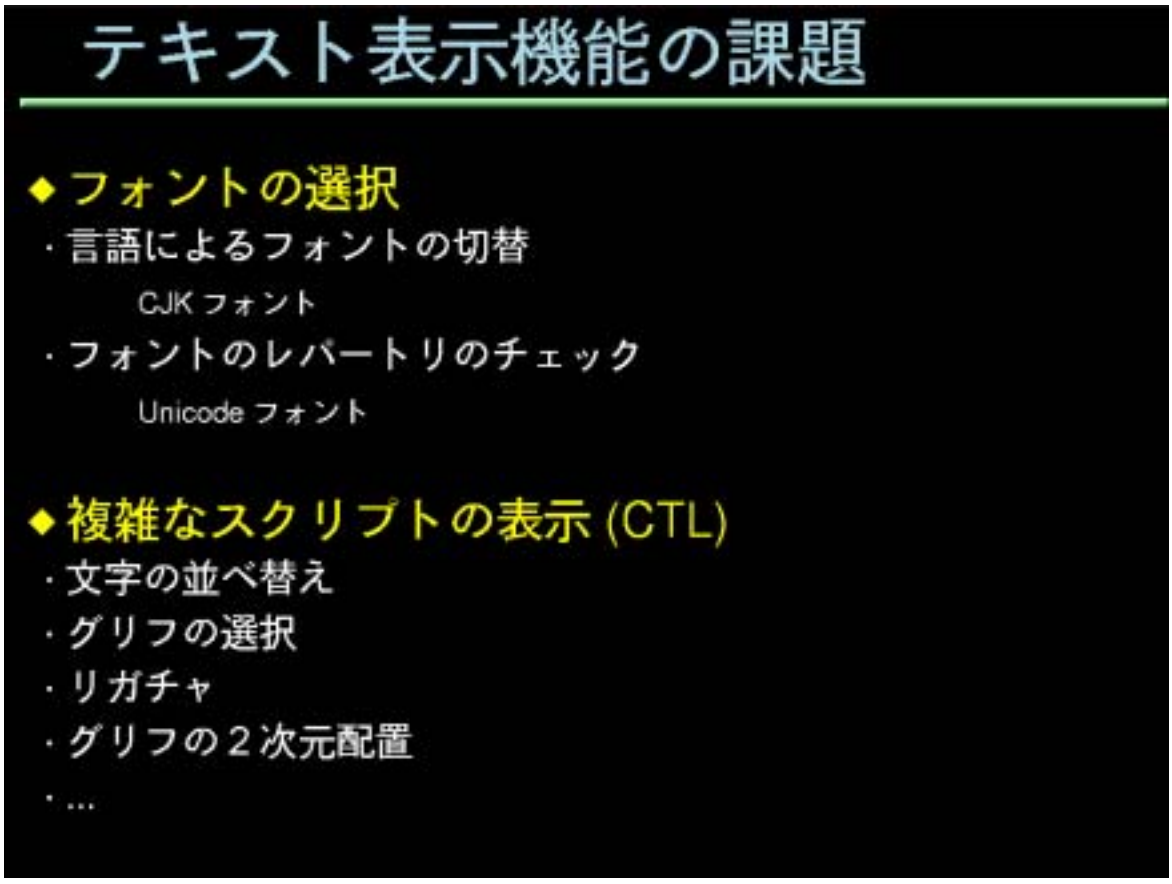
slide #1



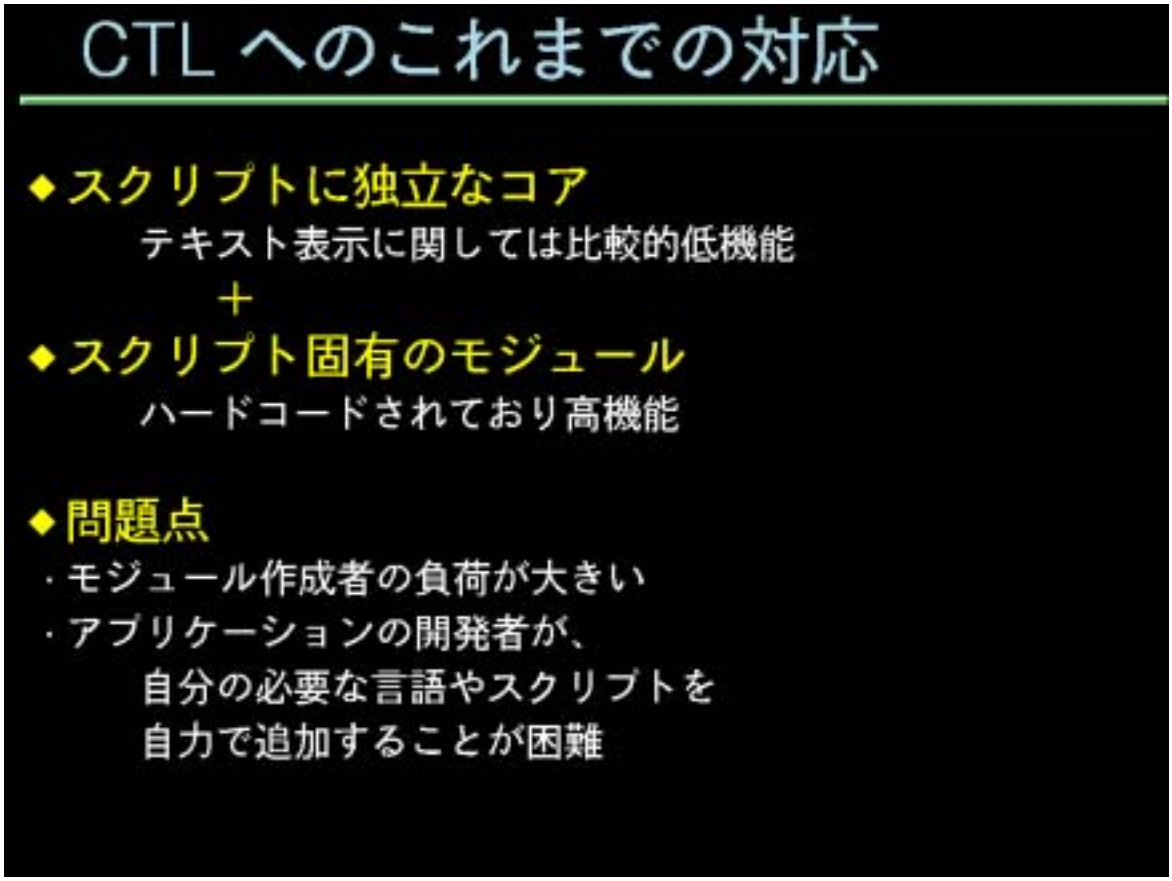
slide #2



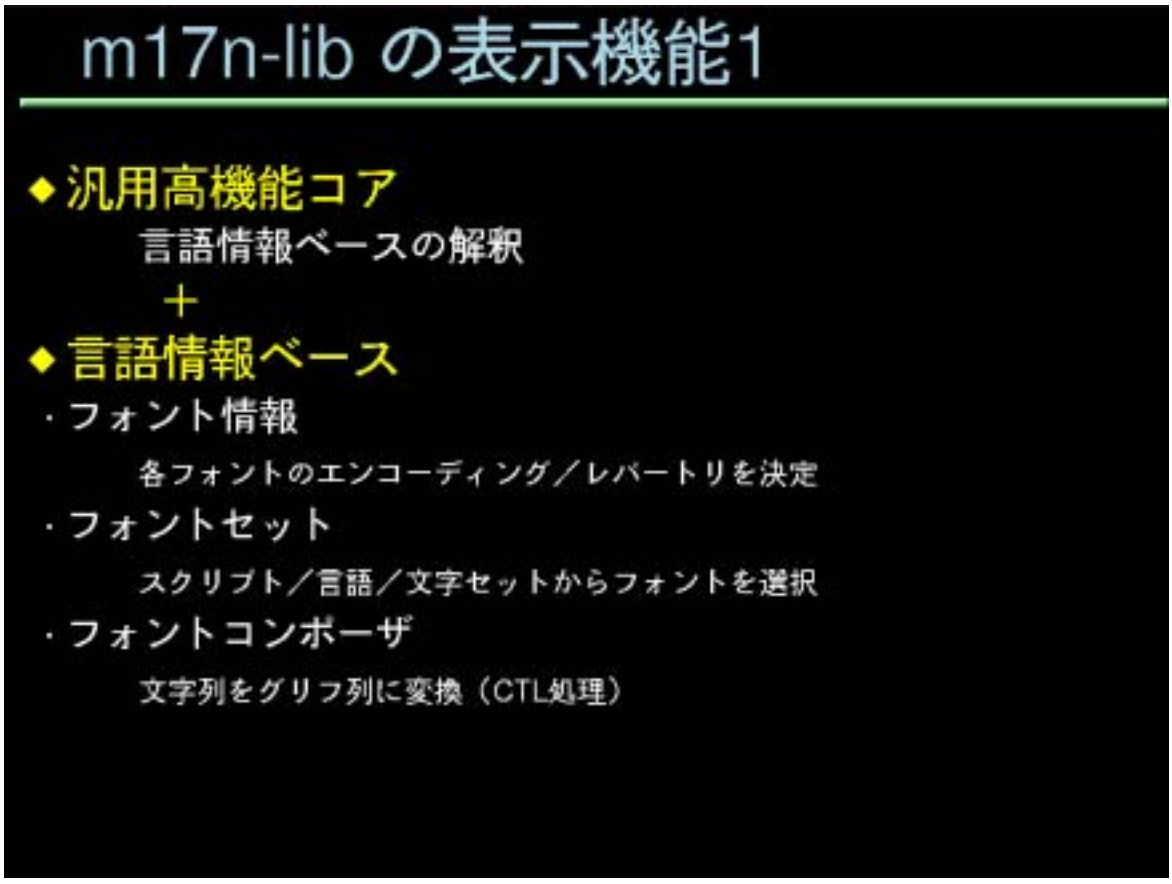
slide #3



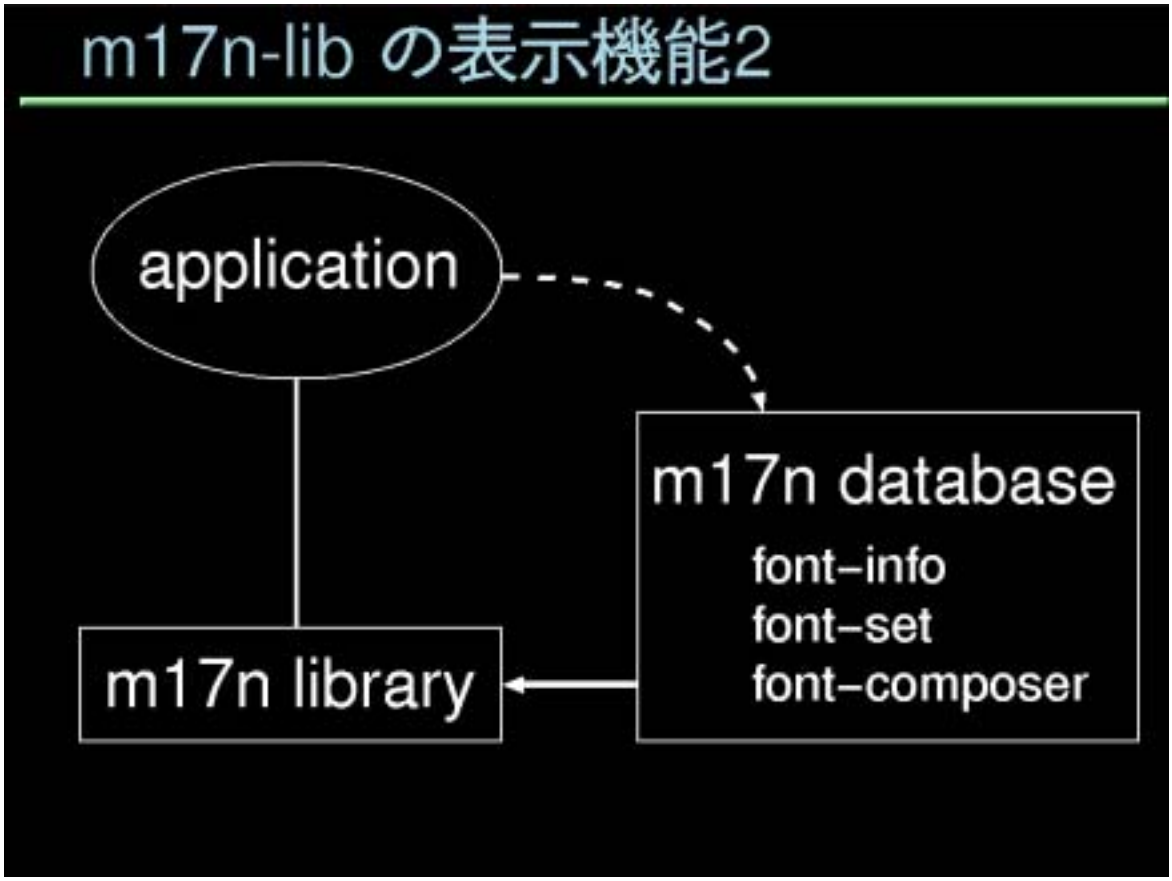
slide #4



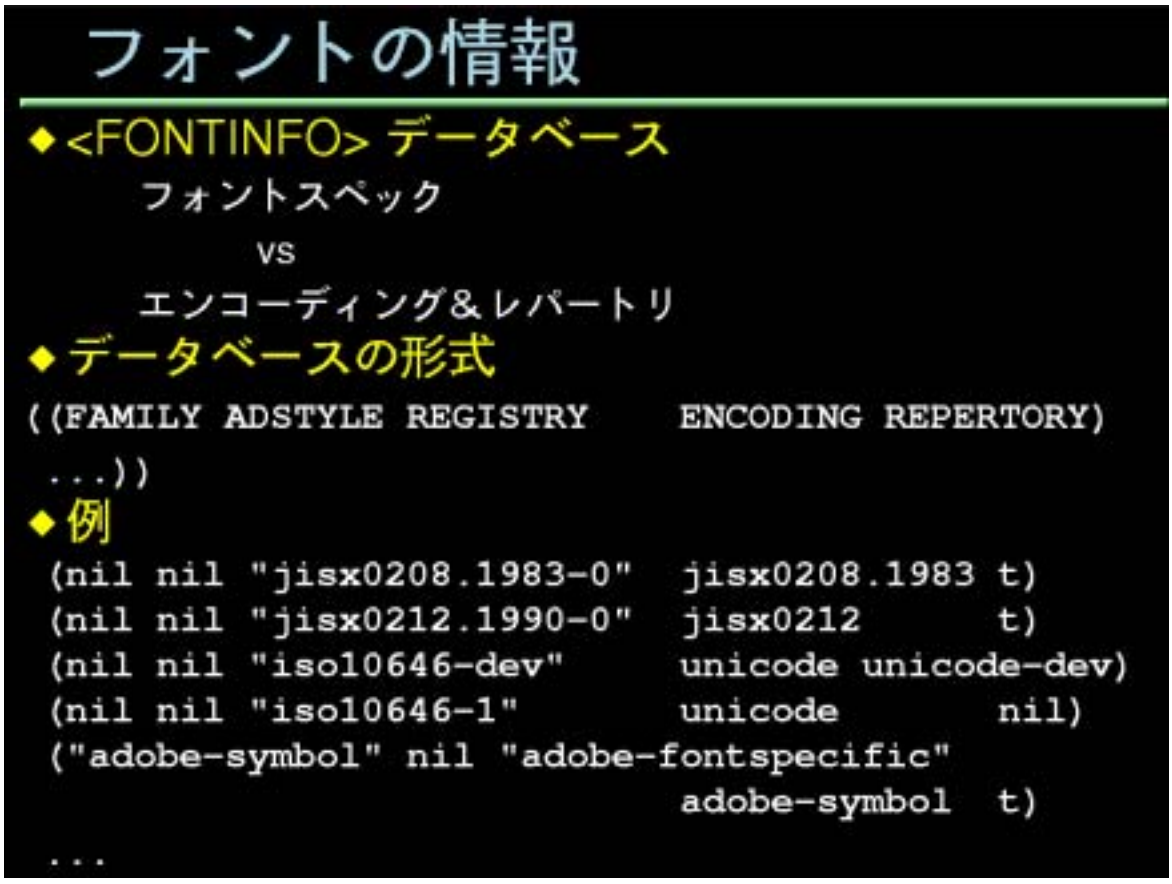
slide #5



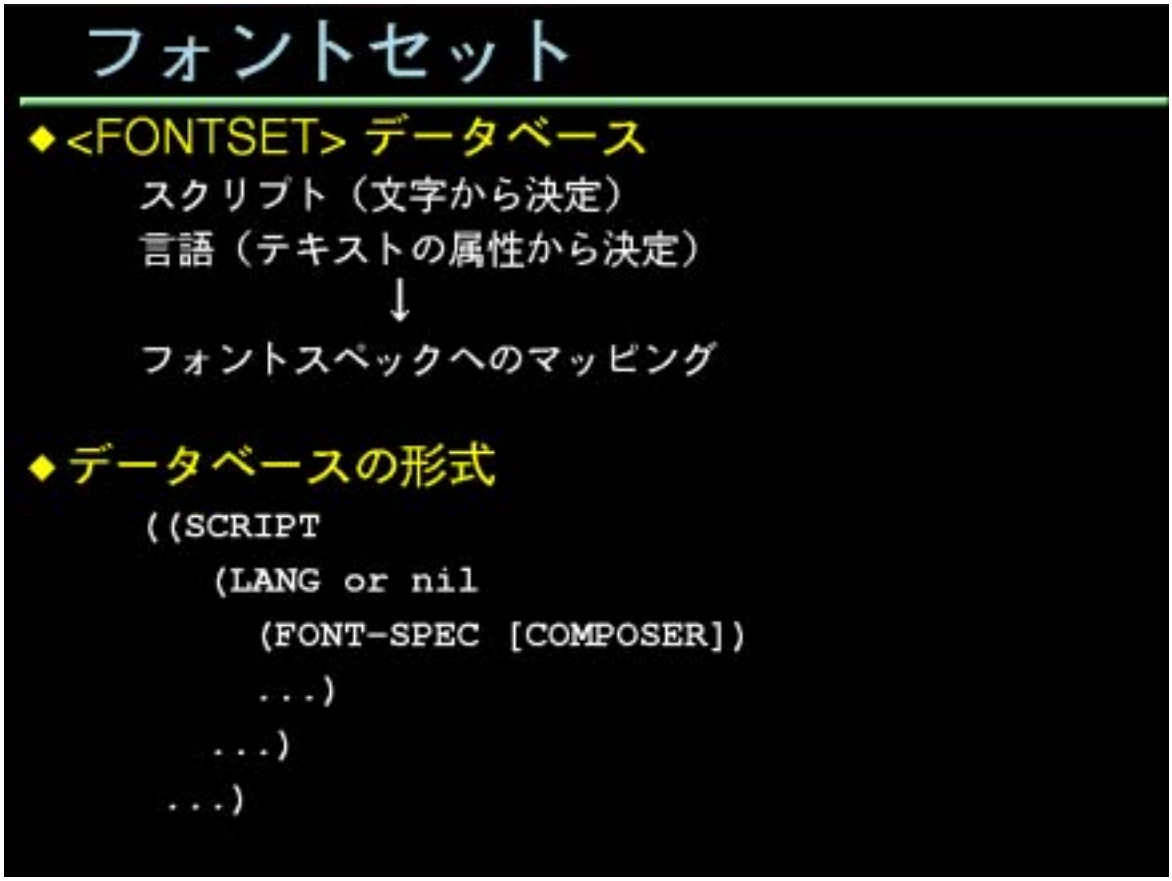
slide #6



slide #7



slide #8



slide #9

フォントセットの例

```
(greek
  (nil (nil nil "iso10646-1")
       (nil nil "iso8859-7")))
(han
  (ja  (nil nil "jisx0208.1983-0")
       (nil nil "jisx0212.1990-0"))
  (zh  (nil nil "gb2312.1980-0")
       (nil nil "big5.eten-0")))
(devanagari
  (nil (nil nil "devanagari-cdac" cdac-dev)
       (nil nil "iso10646-dev"    pango-dev)))
```

slide #10

フォントコンポーザ

◆ タスク

CTL 処理: ह f न ् ड ी ⇒ हिन्डी

◆ 機能

- 書記素(grapheme)クラスタの切り出し
- 文字の並べ替え
- グリフの選択
- リガチャ処理
- グリフの２次元的配置

◆ 特徴

- 多段パスで文字列をグリフ列に変換
- 各パスの変換はルールベース
- 正規表現による文脈依存処理

slide #11

フォントコンポーザの構成

◆構成要素

・コードのカテゴリ定義

入力コードをカテゴリ（A-Za-z）に分類

・生成ルール

入力コード列から出力コード列を生成

(PATTERN COMMAND ...)

◆多段パス

第１ステージのカテゴリの定義

第１ステージの生成ルール（文字列→中間コード列）

...

最終ステージのカテゴリの定義（必要ならば）

最終ステージの生成ルール（中間コード列→グリフ列）

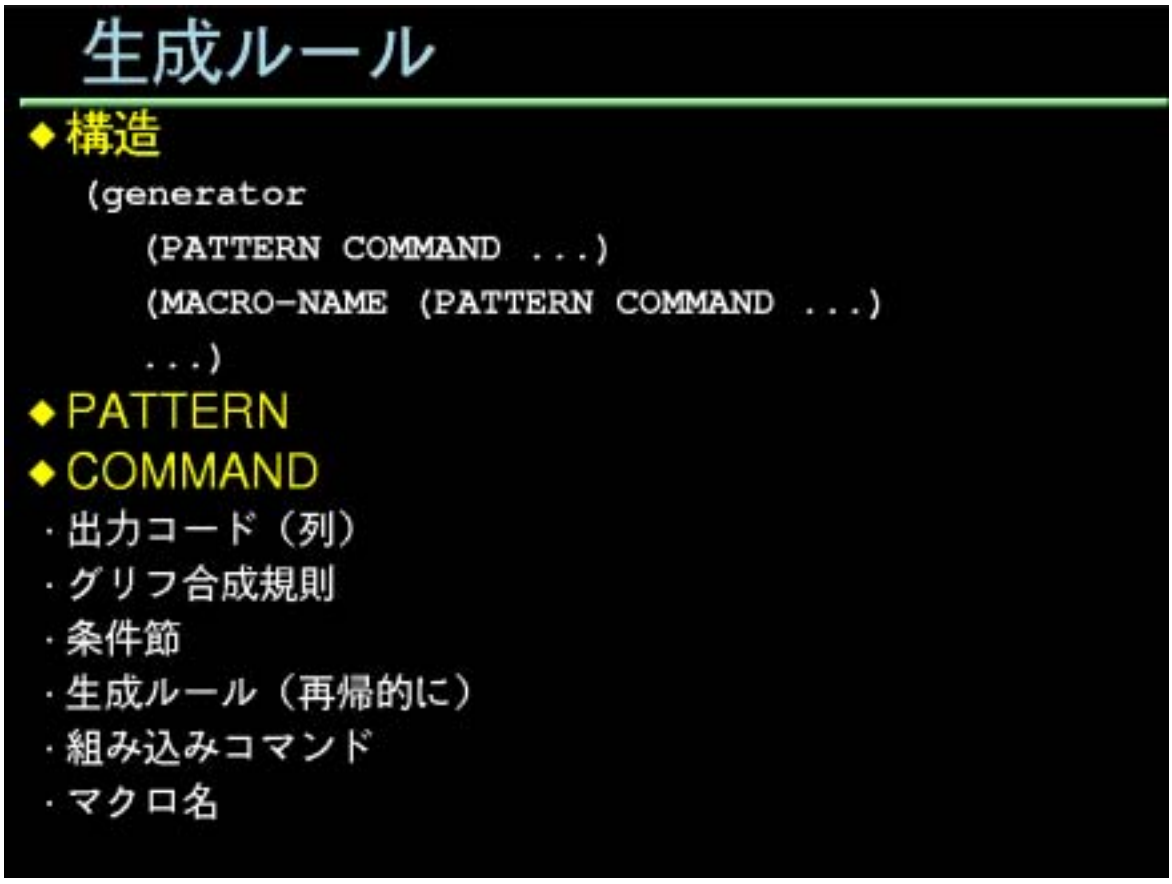
slide #12

コードカテゴリ

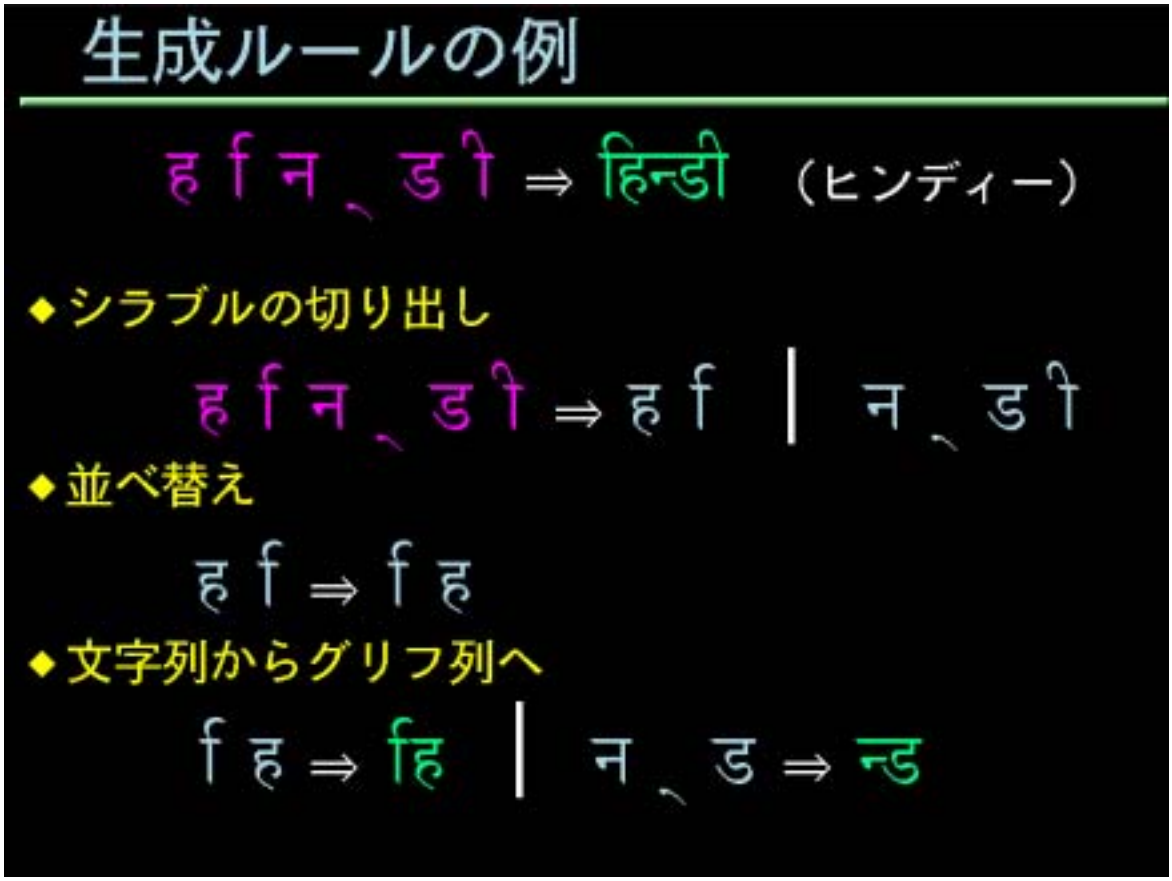
◆デバナガリのカテゴリ定義例

```
(category
 ;; A: ANUSVARA or CANDRABINDU
 ;; C: CONSONANT (except for R)
 ;; R: LETTER RA
 ;; V: VOWEL INDEPENDENT
 ;; E: ELSE
 ...
 (0x0901 0x0902  ?A) ; CANDRABINDU, ANUSVARA
 (0x0903         ?E) ; SIGN VISARGA
 (0x0905 0x0914  ?V) ; A .. AU
 (0x0915 0x0939  ?C) ; KA .. HA
 (0x0930       ?R) ; RA
 ...)
```

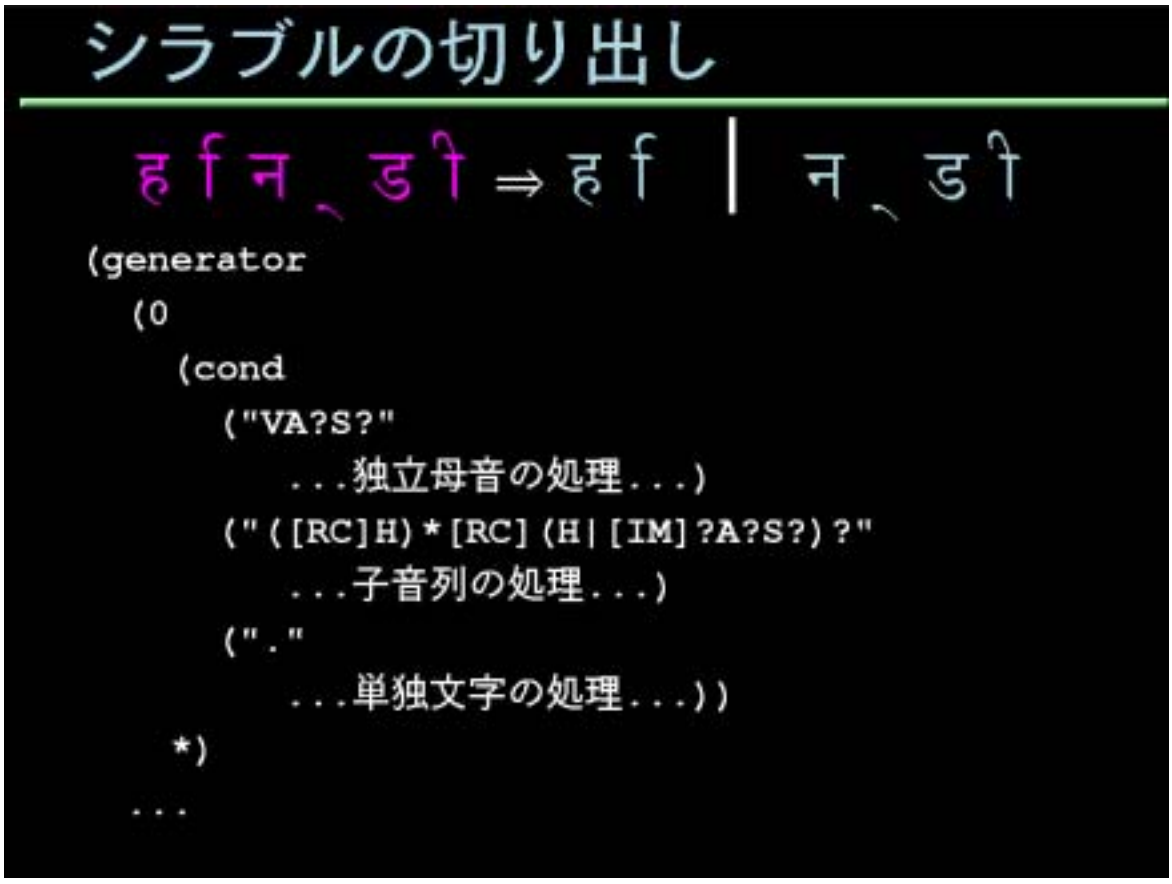
slide #13



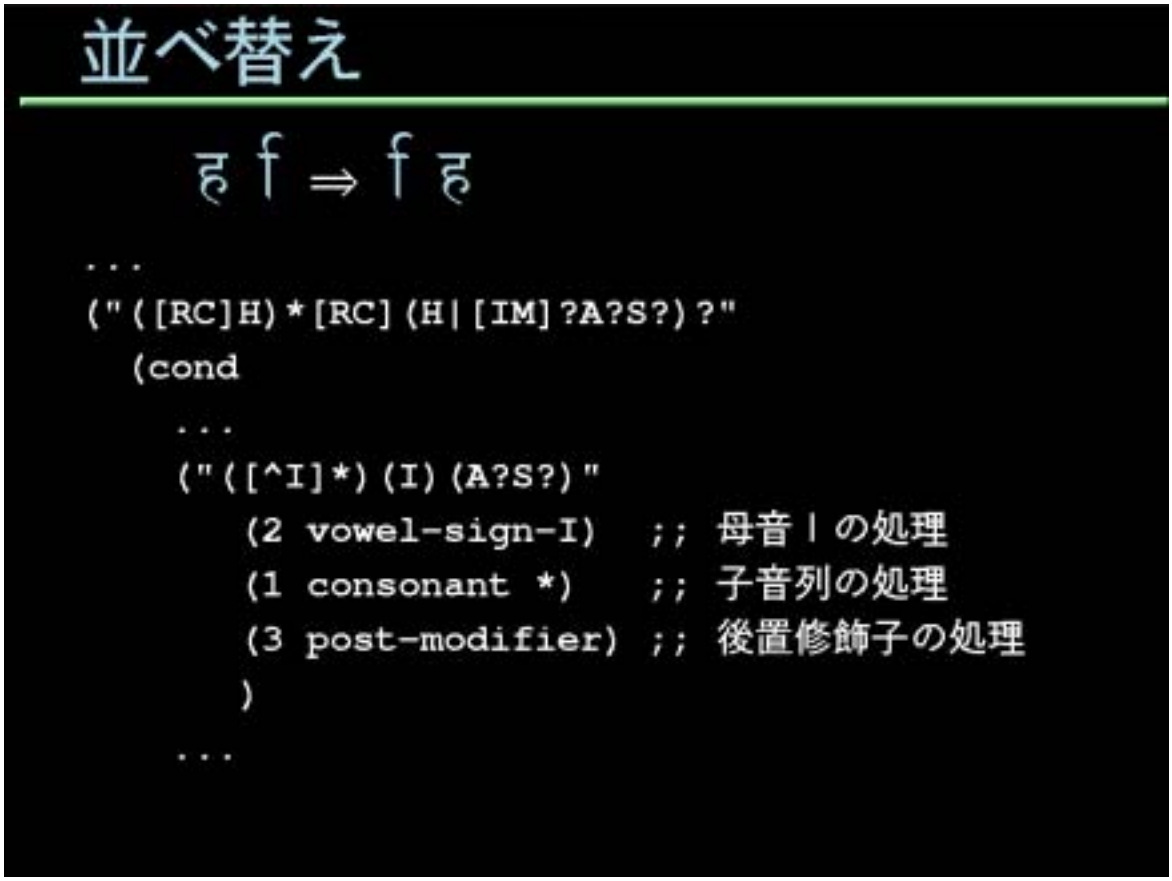
slide #14



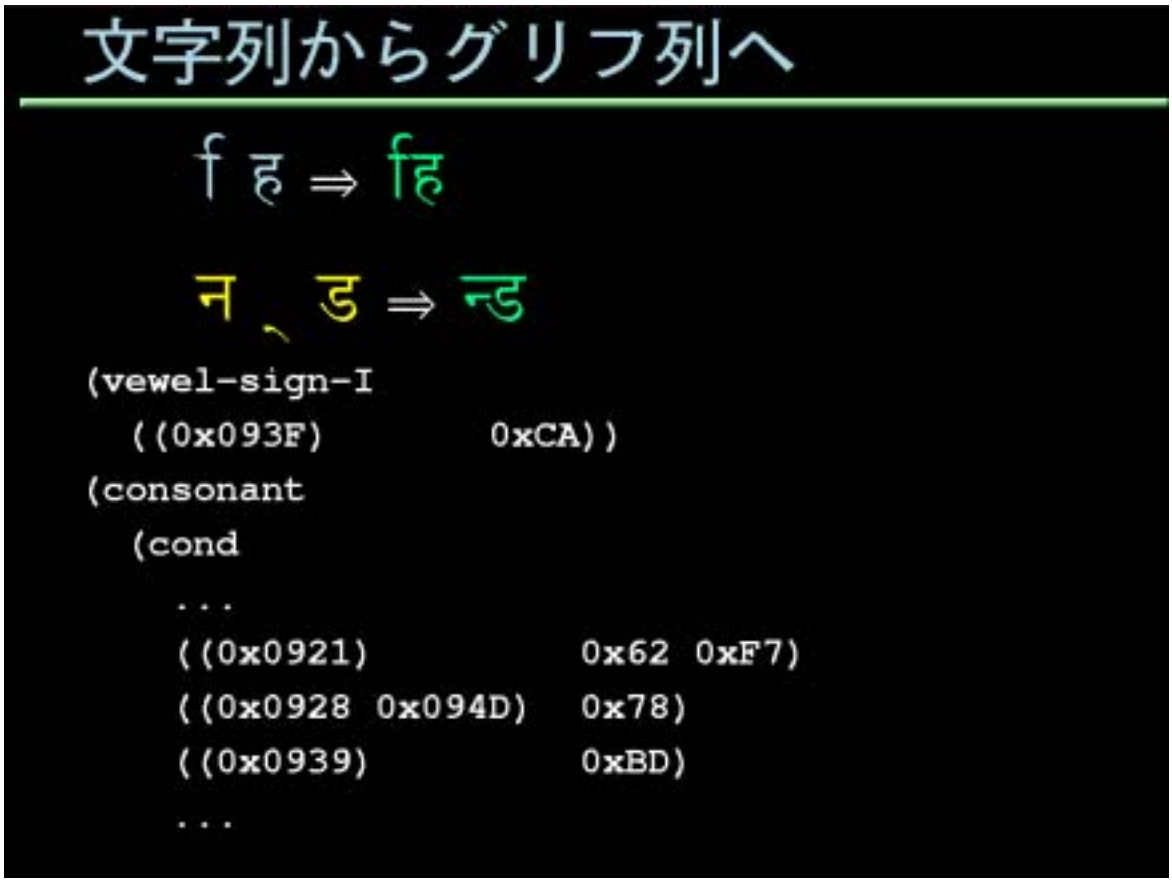
slide #15



slide #16



slide #17



slide #18

slide #19

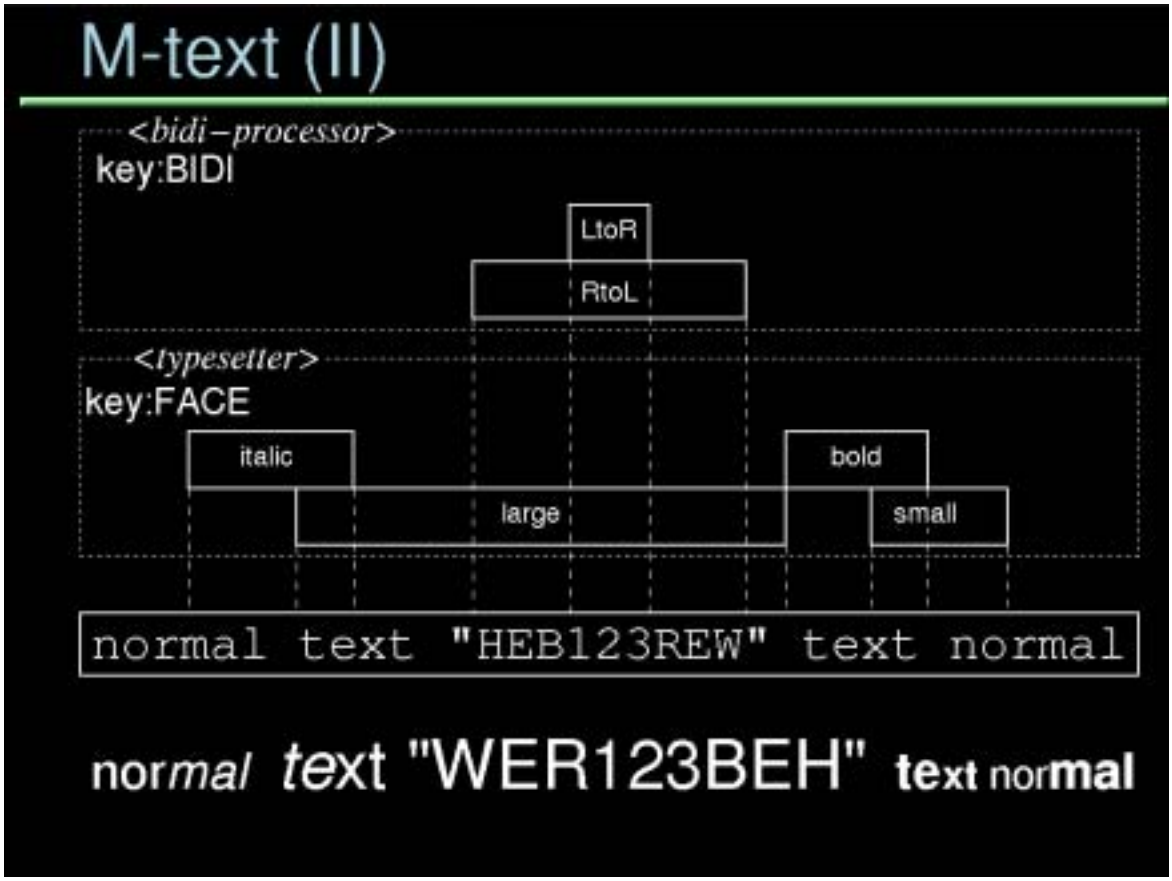
M-Text (I)

◆多言語のテキストを表わす構造体
属性付き文字列オブジェクト

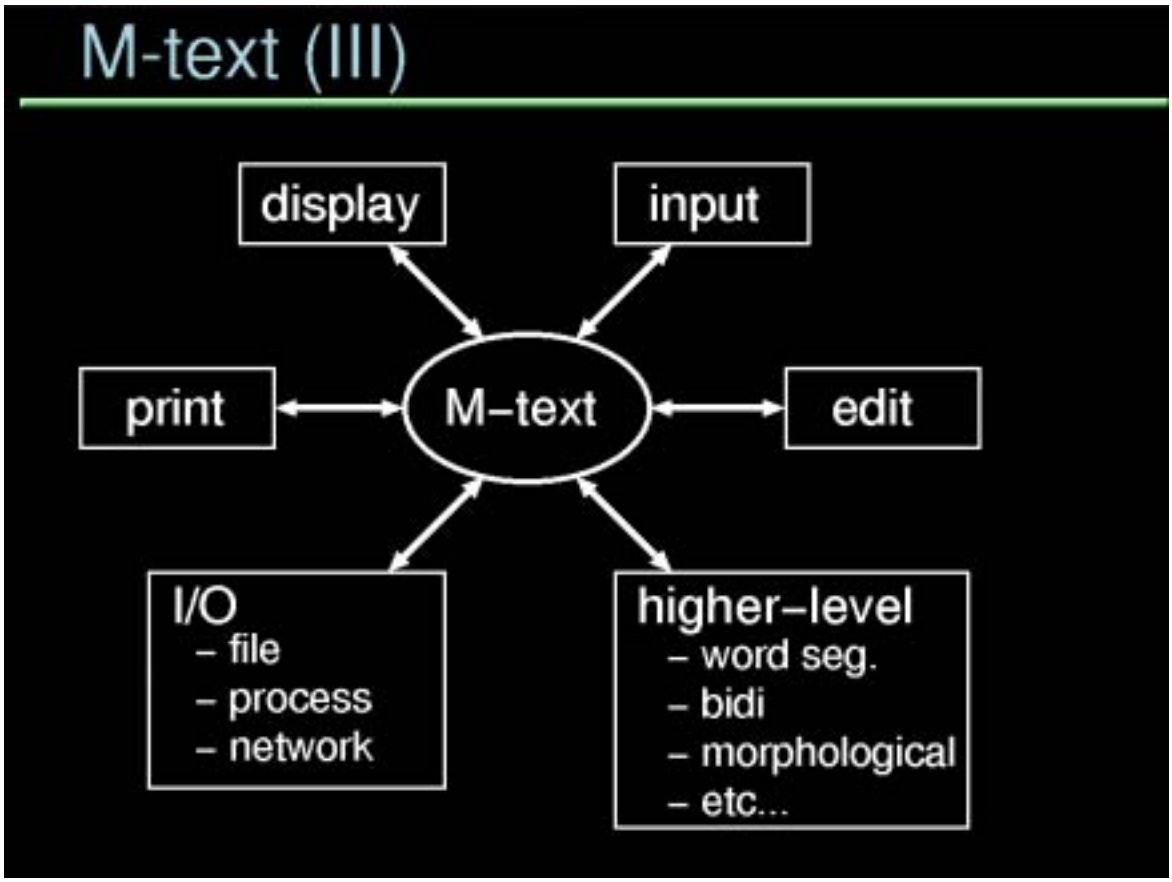
◆テキストプロパティ
属性-属性値のペア
M-text の任意の部分に
任意の属性を
幾層にも付加できる
ex. 言語、書字方向、タイプフェース

slide #20

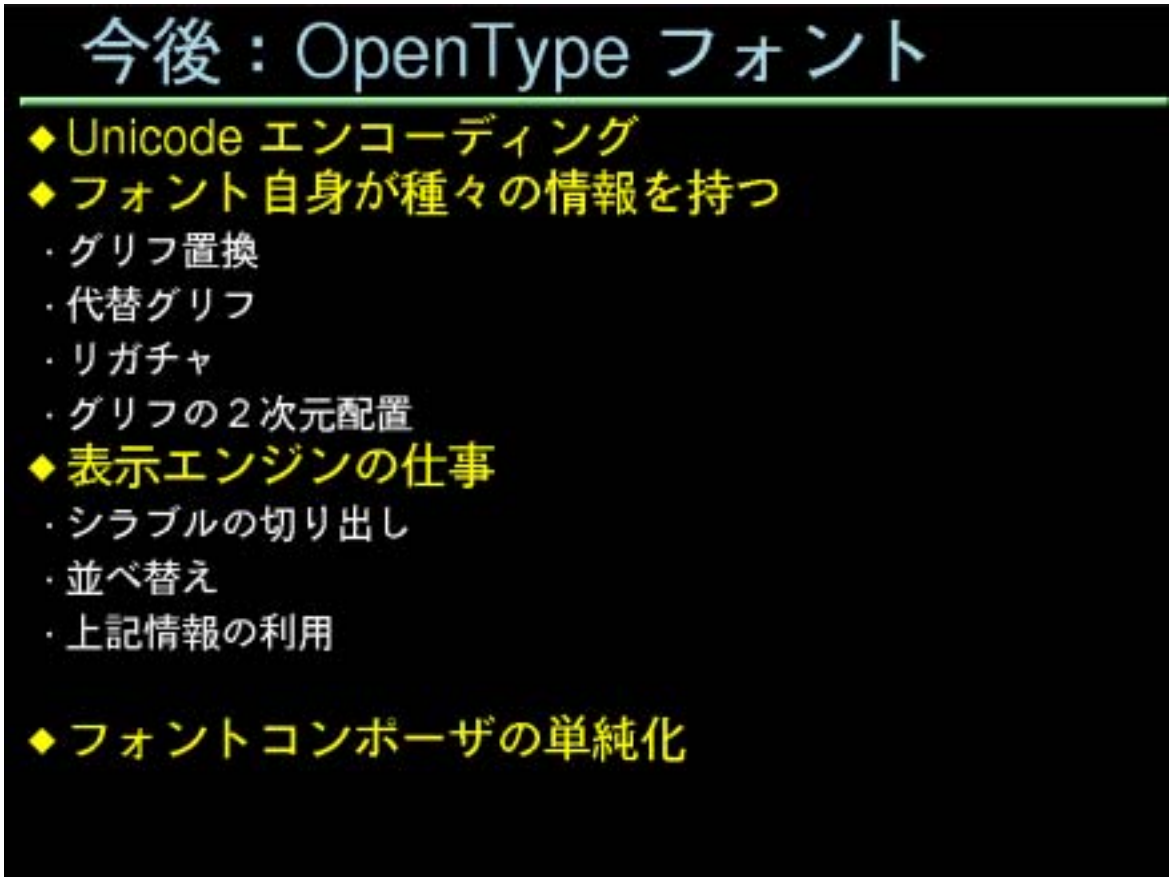
M-text (II)



slide #21



slide #22



slide #23

今後：GNU Emacs 等との関連

- m17nlib は Emacs とは独立したライブラリ
  - Emacs 開発の経験に基づく
  - 表示機能は全く新規に開発
- 現在 Unicode ベースの Emacs も開発中
  - m17nlib と同等の表示エンジンを搭載予定
  - 言語情報ベースも共有

slide #24

今後：GNOME

- 現状
- 将来