非標準的な方法で TeX を利用するプログラミング

鈴木信吾
ichimal@takopen.cs.uec.ac.jp
電通大 情報工学専攻

概要

- 図表的文書作成ツール AFAL-S の実装の詳細の報告
  - TEX や emacs の資源をいかに利用したか
  - AFAL-S は TEX を非標準的な方法で利用する

背景 (1) AFAL

- 図表的文書を記述するための枠組
  科研費申請書類の類

- 図表的文書の構成要素を再帰的に記述する
  - 「パネル状の領域上に文書の構成要素を張り付ける」
- 構成要素の大きさ調整を大幅に自動化

背景 (2) AFAL-S

- S 式で図表的文書を記述させる AFAL の一実装
  - UTiTEX ライブラリとして実装
    - UTiTEX は汎用 TEX プリプロセッサ
      - TEX コード中に lisp 関数呼び出しを書けるようにした
      - emacs lisp で実装

- TEX 文書を出力する

背景 (3) なぜTEXか

- 「お手軽」だと思った
  考えが甘かった

- 普及している

背景 (4) なぜemacsか

- UTiTEX
- 多バイト文字処理
- free

どこがリソコンか

- 「バックエンドや実装言語の選択は重要です」
  - TEX や emacs は半ば自己完結した世界を持つ
    - 強力なツールであることの証明
    - 非標準的なことをするのは面倒
  - TEX を非標準的な方法で使った
    TEX と emacs の資源の限界を強く意識する羽目に

事例 (1) TEXから情報を取得する

- AFAL-S はTEX処理系から動的に情報を取得
  - 細かい粒度のデータが大量に必要
  - 大きい粒度のデータも必要
- 取得方法

  (1) ログファイルに情報を出力

  (2) 任意のファイルに情報を出力

  (3) TEXの標準出力に情報を出力

方法 (1) ログファイル経由

- 長所
  - `\showbox` できる
    人間向けログの一種を出力
- 短所
  - ファイル経由はよくない
    - emacs のバッファを経由しなければならない
  - `\immediate` が効かない
    - 情報を即時取得できない
  - 雑多なログに情報が埋もれる

方法 (2) 任意のファイル経由

- 長所
  - 必要な情報だけを即時取得できる
- 短所
  - ファイル経由はよくない
    - emacs のバッファを経由しなければならない
    - 細粒度の情報が大量に必要
  - `\showbox` できない

方法 (3) 標準出力経由

- 長所
  - ログファイルとほぼ同等の情報を得られる
  - ほぼ即時に情報を得られる

- 短所
  - emacs のバッファを経由しなければならない
    - TEX の出力とその走査が非同期になる

partial typesetter

- UTiTEX のプリミティブ
  emacs のバッファ上で TEX 処理系を動かす機能
  - 部分的な UTiTEX コードを組版できる
    プリプロセス結果を `process-send-string` 関数で一行ずつ TEX 処理系に入力
  - 返値は以下の二つ
    - プリプロセス結果を納めたバッファ
    - TEX 処理系が動いているバッファ

AFAL-Sの方式

(1) partial typesetterでTEXに入力を与える

(2) TEX処理系が動いているバッファを取得

(3) バッファを走査

(4) 文字列処理

問題点

- 必要な情報がゴミログに埋もれる
- emacs のバッファを経由しなければならない
  - TEX の出力とその走査が非同期になる

解決策

- データに目印
  - 一意なヘッダとフッダではさんだ
- ポーリング
- 「おかしい」場合はヘッダ探索からやり直し

事例 (2) 文章の実際の横幅を調べる

- AFAL には文章の実際の横幅が必要
  大きさやレイアウトの調節に利用する
  - 実際の横幅 = 見た目の横幅

- TEX の問題

  任意の文章の実際の横幅を調べる標準的方法が無い

  - TEX は「まず横幅ありき」
    - 指定横幅で綺麗に見えるように組版
    - 余った部分は glue でパディング

解決策

(1) ユーザ入力を制限

- 単一パラグラフに制限

  LaTeX の`\shortstack`マクロ
- 各行を明示的に`\hbox`で囲む

(2) 人間向けログを動的に解析

- `\showbox` プリミティブを利用
  - 組版結果の詳細を人間向けに表示
    - 大量の文字列処理を要する
    - 動的なユーザ入力を要求してくる
    - フォーマットが怪しい
- ユーザ入力はきっと恐らく多分無制限

\showboxによるログの構成

\setbox3\vbox{私の本名は

鈴木信吾です。}

\showboxdepth=1

\showbox3

⇓

```
> \box3=
\vbox(19.77588+1.38855)x469.75499, yoko direction
.\hbox(7.77588+1.38855)x469.75499, glue set 401.64418fil []
.\glue(\parskip) 0.0 plus 1.0
.\glue(\baselineskip) 2.83557
.\hbox(7.77588+1.38855)x469.75499, glue set 392.02202fil []
```

ログ解析

```
.\hbox(7.77588+1.38855)x469.75499, glue set 401.64418fil []
```

高さ　深さ　横幅　パディング

- 全行について…
  (1) 横幅を調べる
  (2) パディングを調べる
  (3) (1) - (2) で行の実際の横幅を求める
  - 結果的に文章の横幅がわかる

事例 (3) 文章の大きさを調整する

文章の「表示領域の最低限度の大きさ」は取得できたが…

- 再構成が必要になる
  - 表示領域の大きさを調整
  - レイアウトを調整
    左詰めで仮組み

問題点

- 大きさの変更を TEX に任せられない
  - 改行位置が変わるかもしれない

AFAL-Sでの実装方針

- 仮組みした文章を分解・再構成する
  - `\vsplit` を使う
    組版した「文章」を上から一行ずつ分解できる

- 問題点
  - `\vsplit` の仕様
    - 「上からこの範囲に含まれる行」
    - glue などを無視 or 無効化する

解決策

- \showbox でログ解析
  - 各行の縦幅を調べる
  - ログから一部情報を再構築し挿入する

    penalty, glue, special, ...

再構築手順

(1) \vsplit で行に分解

(2) \vbox に\unvbox

(3) (2) を\hbox でラップ

(4) (3) を全行分まとめて\vbox to でラップ

その他の問題

- 数値の取扱い

  TEX は内部的には全て scaled point (sp) で整数演算しているが、出力は全て point (pt) による小数表示
  - emacs 上での数値の取扱いが面倒
    - 単位換算と有効桁
- 「高さ」の概念の違い
- コンパイルできない?
  - 二重マクロがコンパイラに通らない
  - 実装のマズさ

    動的変数を使い過ぎ

まとめ

TEX と emacs を組み合わせれば高度なことができるが…

- お互いの資源の限界を把握しなければならない
  - TEX で出来ることはなるべく TEX にやらせるべき
    TEX — emacs 間のやりとりは少ない方が良い
    - 実行効率
    - 数値演算の精度
  - TEX の挙動を予測・制御しなければならない
    「バッファ上でプロセスが動く」のも善し悪し