型による静的検証能力の高い組版システムの開発
– SATYSFI: Typesetting System with a Type System –

1. 背景

書記言語を媒体として意思疎通を行なう社会に於いて，組版処理システムは不可欠なものである。組版処理システムには，大きく分けて GUI 上で版面がプレヴューされた状態で文書を記述していく WYSIWYG エディタ方式と，マークアップ言語によって文書を記述し言語処理系に入力として与えて最終的に PDF などの形式で版面を出力するマークアップ言語方式との 2 種類がある。どちらか一方が全面的に優れているというわけではなく利点と弱点のトレードオフがあると考えられる。マークアップ言語方式一般の利点は，傾向として

- 入力がテキストファイルなので差分管理が簡単
- コマンド定義機能により：
  - 複雑な自動処理が実現できる
  - 文書の内容を変更せずに後から体裁を柔軟に変更しやすい

といったことが挙げられ，これらの恩恵を活かして文書を作成したい場合がマークアップ言語の想定ユースケースということになる。一方で弱点としては

- ユーザが不適格なコードを入力として与えやすい

という点が挙げられ，処理系が不適格なコードに対してどのようにエラーを報告するかがユーザの執筆効率に大きく影響すると考えてよいだろう。ところが既存のマークアップ言語方式の組版システムは，特に TEX/LATEX を例にとると，組版処理中に不整合が生じたときに初めてエラーが報告されるので提示が遅く，内容も不親切と言わざるを得ない。そのため，ユーザは自分の書いた記述のどこに不適格な部分があるのかすぐには特定できず，円滑に執筆を進めることができない，という問題があった。またパッケージ開発者にとっても，コマンド定義を記述したコードの可読性が低く，変更や修正がしづらいという別の問題点の存在もあいまって，デバッグが困難を極めるという状況にあった。

2. 目的

本プロジェクトは，前節に於いて指摘した

- エラー報告機能が不親切で遅いため，文書執筆やパッケージ開発の効率が悪い
- コマンド定義を記述したコードの可読性が低く，変更や修正がしづらい

という既存のマークアップ方式の組版処理システムが抱えている問題が解消された，新しいマークアップ方式の組版処理システムを実装することを目的として実施されたものである。

3. 開発の内容

本プロジェクトでは，コマンド定義を記述したコードの可読性が高いため実装や修正がしやすく，またユーザが不適格な入力を与えたときのエラー報告が素早くわかりやすいという性質を備えた新たなマークアップ言語方式の組版処理システム SATYSFI を開発した。具体的な方法としては，まず“主にコマンド定義用の言語”と“主に文書を書くための言語”の 2 層に切り分けて言語を設計し，この言語に対する処理系を実装した。文書を書くための層は見た目上おおよそ LaTeX に近い設計にした一方で，コマンド定義用の層は OCaml や F# に近いいわゆる**函数型言語**らしい言語仕様をベースとし，これにより

- グローバルな状態を意識せず局所的な処理にだけ気を遣ってコマンド定義が読み書きできる
- 強力な型システムを搭載しやすい

といった，函数型言語に関して既に共有されていた知見を恩恵として一気に享受することで上記の性質を達成した。処理系は大きく分けて

- ユーザやパッケージ開発者による不適格な入力を検出し，エラーとして報告するフロントエンド（構文解析器・型検査器）
- フロントエンドでの検査を通った入力に組版処理を施して PDF を出力するバックエンド

からなっており，模式的に描くと図 1 のような構造をしている。組版処理の機能としては，以下に例として一部掲げるように，**合字**・**カーニング**・**自動ハイフネーション**などの施された欧文，（部分的に）『**日本語組版処理の要件**』に準拠した和文，数式，画像の挿入などができる程度に成熟している。なお本文書も SATYSFI で組まれているので，本文書の版面にあるものはすべて SATYSFI で出力できる。

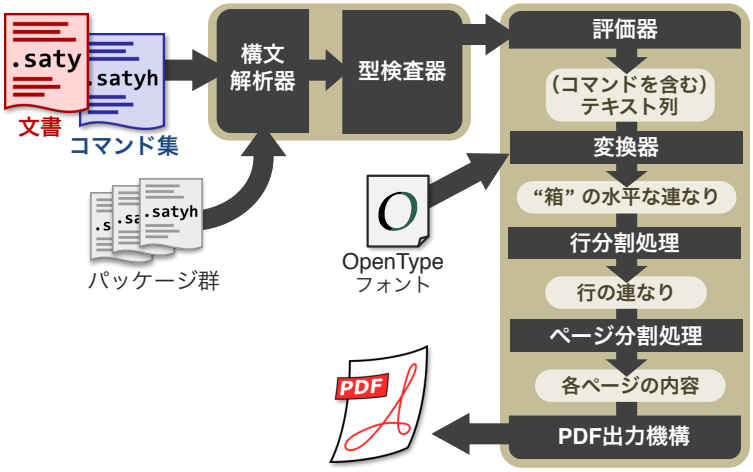

図 1 SATYSFI の内部構造の模式図

The quick brown fox jumps over the lazy dog. ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

$$\frac{\displaystyle\int_0^a x\,\mathrm{d}x \int_0^{\sqrt{a^2-x^2}} r\sqrt{x^2+r^2}\,\mathrm{d}r}{\displaystyle\int_0^a \mathrm{d}x \int_0^{\sqrt{a^2-x^2}} r\sqrt{x^2+r^2}\,\mathrm{d}r} = \frac{2a}{5}$$

4. 従来の技術（または機能）との相違

前節で既に述べたように，従来のマークアップ言語方式の組版処理システムとは

異なり，SATγSFi はユーザやパッケージ開発者の与えた入力に不適格な記述が含まれていれば組版処理を始めるよりも前に（＝静的に）エラーとして報告することができる。特に型エラーは以下のようなものである：

```
! [Type Error] at line 13, character 26 to line 16, character 1:
    this expression has type
      block-text,
    but is expected of type
      length.
```

この例は「13 行目 26 文字目から `block-text` 型の引数が与えられているが，本当は `length` 型の引数，つまり何らかの長さの指定が与えられることが期待される場所である」というエラーの報告で，コマンドについた型と引数の型との不整合が報告されているのである。

5. 期待される効果

本システムでは，ユーザが不適格な記述をコードに埋め込んでしまった場合も（構文エラーや）型エラーとして組版処理を始めるよりも前に素早くエラーが提示され，そのエラー報告内容も多くの場合直接の原因を指している傾向にあるため，従来システムに比べ効率的に文書を執筆することができる。またエンドユーザだけにとどまらず，従来システムでは実装したコマンドのデバッグや既存のコマンドの改造が難しいといったパッケージ開発者の抱えていた問題も，本システムに備えられた型検査の機構により大幅な解決が見込まれる。

6. 普及（または活用）の見通し

解説書・仕様書を整備し，積極的にプロモーションを行なう予定である。既にいくつか文書作成用のパッケージを整備しているが，今後も拡充していく所存である。

7. クリエータ名（所属）

諏訪 敬之（東京大学大学院情報理工学系研究科コンピュータ科学専攻）

（参考）関連 URL

SATγSFi リポジトリ： `https://github.com/gfngfn/SATySFi`