OS に依存しないキーマッピングのカスタマイズシステム
- violets キーマッピングの記述に特化した言語 -

1　背景

キーボードは，キー入力によりコンピュータを操作するインタフェースである．しかし，キーボードは計算機の発明以前に開発されたタイプライターに起源を持ち，歴史的な経緯を経て現代的なコンピューティングのためのインタフェースとしてはいくかの問題をかかえている．キーの物理配置についていえば，役割が重複するキーが複数個あったり，使用頻度の高いキーがホームポジションから遠い位置に配置されていたり，合理性のない物理配置になっている．キーボードの挙動についていえば，キーの押下と引上と 2 つのキーイベントがあるにもかかわらずキー押下しかキー入力に利用してこなかったり，修飾キーとの同時押しに頼るばかりでキーの連続した入力に対して意味をもたせてこなかったり，ユーザの仕様場面に応じてキーボードの挙動を変化させることができなかったりしている．

このキーボードに関する問題に対するソフトウェア的なアプローチがキーマッピングソフトウェアを用いたキーマッピングのカスタマイズである．あるキー入力に対し特定のコンピュータの操作を対応付けることをキーマッピングと呼び，キーボードユーザが自身の嗜好に合わせた柔軟なキーマッピングができるようになれば，ユーザの生産性を向上させることが期待できる．しかし，キーマッピングは，複雑なイベント，フラグ管理が発生しやすく，キーマッピングに対するモデル化がキーマッピングソフトウェアごと大きく異なっており，OS を横断してキーマッピングを記述する言語がなかった．そのため，キーマッピングソフトウェアにロックインされ乗り換えが難しかったり，一度定義したキーマッピングを再配布しやすい形で提供し，複雑な条件分岐が発生する特定キーイベントの並びに対して機能を割り当てることが困難であったりした．このようなキーマッピング記述言語がばらばらであることがキーマッピングの普及の障害となり，環境をまたいでユーザがナレッジを共有を阻みキーマッピングの認知は一般ユーザのみならず開発者の間でも高いとはいえなかった．

2　目的

本プロジェクトでは，OS に依存しないユーザフレンドリーなキーマッピングのカスタマイズシステムとして，既存のキーマッピングツールの設定ファイルを生成することが可能なキーマッピングの記述に特化した言語を開発する．別個に実装されてきたキーマッピングノウハウを統一的な記法で提供し，キーマッピング設定が再編集，マージがしやすい環境をつくる．ユーザが本システムを利用することにより，OS をまたいだキーマッピングのカスタマイズ設定を実現し，快適なタイピング環境の構築をサポートする．キーマッピングを導入するための障壁を取り払うことでユーザの裾野を広げ，環境ごとに分断されていたキーマッピングのノウハウを統合し，キーマッピングの普及と発展を目指す．

3　開発の内容

本プロジェクトで開発した violets の全体概要を以下に示す（図 1）.

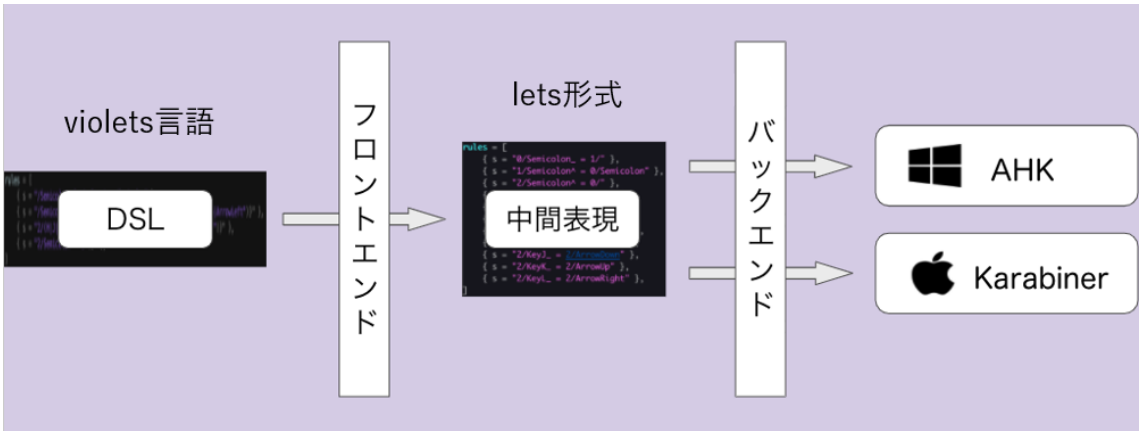

図 1: プロジェクト概要

- lets 形式: OS に依存にしない中間表現
- バックエンド: キーマッピング中間表現から各キーマッピングソフトウェアの設定ファイルを生成する機構
- フロントエンド: violets 言語で書かれたキーマッピング定義を中間表現へ変換する機構
- violets 言語: ユーザがキーマッピング定義に使う独自の DSL

図 2 は，ワンショットモディファイアの状態遷移図を表したもので，

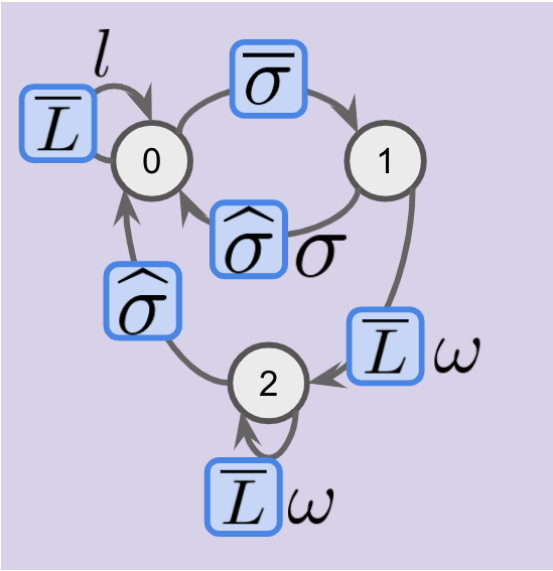

図 2: ワンショットモディファイアの状態遷移図

- セミコロンキーを押し下げて引き上げると，セミコロンを入力
- セミコロンキーを押しながらL キーを押すと，カーソルを右へ進ませる入力

という動作を示している．
状態遷移図の記号は

- $\overline{\sigma}$ がセミコロンキーの押下
- $\widehat{\sigma}$ がセミコロンキーの引上
- $\overline{L}$ がL キーの押下
- ‘;‘がテキスト ‘;‘の出力
- $\rightarrow$ がカーソル右移動

を表す．
これに対応する violets の記述が次の通りである．

```
name = "lcolon"
edition = 202002
rules = [
    { s = "0/Semicolon_ = 1/" },
    { s = "1/Semicolon^ = 0/‘;‘" },
    { s = "2/Semicolon^ = 0/" },
    { s = "1/KeyL_ = 2/ArrowRight" },
    { s = "2/KeyL_ = 2/ArrowRight" },
]
```

この記述の意味するところは次の通り．

- 1 行目で図 2 の状態 0 からセミコロンキーの押下で状態 1 へ遷移する
- 2 行目で図 2 の状態 1 からセミコロンキーの引上で状態 0 へ遷移し，テキスト ‘;‘を出力する
- 3 行目で図 2 の状態 2 からセミコロンキーの引上で状態 0 へ遷移する
- 4 行目で図 2 の状態 1 からL キーの押下で状態 2 へ遷移し，カーソルを右へ移動する
- 5 行目で図 2 の状態 1 からL キーの押下で状態 2 へ遷移し，カーソルを右へ移動する

このようにして，状態遷移を用いてキーマッピングを定義する．このキーマッピング設定を元に，AHK（AutoHotKey）, Karabiner（Karabiner-Elements）をターゲットとしてそれぞれの設定ファイルを出力する．
この他に次のようなキーマッピング記述言語として次のような機能が実装されている．

- 操作中のアプリに応じたキーマッピング
- シェル実行
- キーマッピングの優先度
- グルーピングとキャプション
- キーマッピングパターンの再利用
- モジュールのサポート

4　従来の技術との相違

キーマッピング記述言語として AHK，Karabiner と比較をすると，キーマッピングの持つ複雑な条件分岐やイベント，フラグ管理を非常にシンプルな構文にまとめた点が本システムの特徴である．それにより，カスタマイズ性とモジュール性を両立させた，再利用，再配布しやすい形式でキーマッピングを定義することができる．キーマッピングを表現する式として比較的わかりやすく，モノリシックにキーマッピングを定義することなく，限定的ながら宣言的なキーマッピング定義が可能になった．

5　期待される効果

ユーザが本システムを利用することにより，OS をまたいだキーマッピングのカスタマイズ設定を実現し，快適なタイピング環境の構築をサポートされる．キーマッピング設定が再編集，マージしやすくなったため，環境ごとに分断されていたキーマッピングのノウハウの共有が加速されキーマッピングの普及と発展が期待される．

6　普及の見通し

本言語のグルーピング表記とモジュール表記により，状態遷移を書き下すための記述量は減っているはずである．ここからさらに正規表現の量指定子の導入を検討するなどをして，ユーザが直接記述しなければならない状態遷移の負担を軽減する，よりユーザフレンドリーな記法の実現に努める．

7　クリエータ名（所属）

- 福田優太朗（電気通信大学大学院）

(参考) 関連 URL

- ティザーサイト：`https://violets.io/`