

SATySF_I-formatter

usagrada

目次

1. formatter の install 方法	2
2. formatter の使い方	2
3. 開発者の方	3
3.1. Pull Request	3
3.2. issue	3
4. SATySF _I -formatter の実装	4
4.1. indent の管理	4
4.2. そのまま出力する場所	5
5. document の更新	6
6. format option	6
6.1. インデント	6

SATySF_I を使うに当たって、formatter が無いのが不便だったので、format をするためのツールを作りました。^{*1 *2}

1. formatter の install 方法

以下のどちらかの方法で、入れることができます。ターミナルに以下のコマンドを打ち込んでください。^{*3}

```
cargo install --force --git https://github.com/usagrada/satysfi-formatter. ↵  
↳ git --branch main
```

```
git clone -b main https://github.com/usagrada/satysfi-formatter.git  
cd satysfi-formatter  
cargo install --path .
```

2. formatter の使い方

```
satysfi-fmt $input --write
```

-w または --write を指定することで、input で入力したファイルに書き込むことができます。以下の --output のオプションが指定されている場合、--output が優先されます。

```
satysfi-fmt $input -o $output
```

-o または --output でファイルを指定して出力できます。

1 このドキュメントは format のテストも兼ねて書いています。

2 愚直な実装をしているため、期待するフォーマットが得られない場合が存在します。その場合は、Issue や Pull Request いただくと幸いです。(23/1/27 現在)

3 --force は無くても入りますが、既にインストールしている場合、最新のデータにアップデートするために同じコマンドを使用できます。

3. 開発者の方

release build でない場合、src/visualize.rs にある関数が呼び出されるようになっており、ファイルの構造を確認できるようになってます。

```
cargo run -- $input
```

lib.rs の format を開始地点とし、コードから satysfi-parser で CST 化し、文字列に戻して結合しています。現状では、かなり愚直な実装をしている + 一部のみしか対応していない (コメントが消去される、改行入れて欲しいのに消える etc.) ため、修正等があれば、Pull Request や Issue をお願いいたします。Issue でいただく場合、期待するフォーマットのテストをいただけるとスムーズに対応ができると思います。その際、実際にそれがコンパイル可能である必要はありません。

3.1. Pull Request

実装した部分のテストケースを書いていただいてから、プルリクエストをいただけると幸いです。その際、src/tests 以下でしたら何処に書いていただいても構いません。

3.2. issue

以下にサンプル (src/tests/common.rs test1 と同じ) を載せておきます。r#" の内部に書かれたテキストはスペースや改行を含め全てそのまま出力されるため、スペース数改行数等の違いにより、テストが容易に落ちます。

```
#[test]
fn test1() {
    // format 前のテキスト
    let text = r#"@import: hello
@require: local

document(|title = {hello}|)'<+p{hello world}>"#;

    // 期待されるテキスト
    let expect = r#"@import: hello
```

```
@require: local

document(|title = {hello}|)'<
  +p { hello world }
>
"#;
  test_tmpl(text, expect);
}
```

4. SATySF_I-formatter の実装

4.1. indent の管理

SATySF_I-formatter では以下の場所でインデントの管理を行っています。

インデントが以下の場所では深くなります。

- record
- cmd_text_arg
- type_record
- block_text
- horizontal_list
- list
- type_block_cmd
- type_inline_cmd
- math_cmd
- match_expr
- let_rec_matcharm
- let_rec_inner
- sig_stmt

- `struct_stmt`

`let-*` 文については 次の行が `let-*` 文以外るとき、インデントが 1 つ深くなります。

4.2. そのまま出力する場所

- 変数名
 - `pkgname`
 - `arg`
 - `typename`
 - `var`
 - `var_ptn`
 - `modvar`
 - `mod_cmd_name`
 - `module_name`
 - `variant_name`
 - `cmd_name_ptn`
 - `block_cmd_name`
 - `math_cmd_name`
 - `math_cmd_expr_arg`
 - `pattern`
- これ以上分解できないもの
 - `horizontal_escaped_char`
 - `const_string`
- `prefix`
 - `unary_prefix`
- TODO(今後実装予定)
 - `expr_with_mod`
 - `math_cmd`
 - `math_text`

5. document の更新

`draft.saty` を更新して、以下のコマンドを叩くと `doc.pdf` を更新します。

```
cargo make build-doc
```

その際、`cargo-make` というパッケージが必要なので、インストールしていない方は以下のコマンドでインストールしてください。

```
cargo install --force cargo-make
```

6. format option

6.1. インデント

基本の indent は 4 です。引数に `-i number` を書くことでインデントのスペースを管理できます。

```
satysfi-fmt -i 2 doc/draft.saty -o doc/doc.saty
```