

# SATySF<sub>I</sub>-formatter

usagrada

---

---

## 目次

---

---

1. formatter の install 方法 .....	2
2. formatter の使い方 .....	2
3. 開発者の方 .....	2
3.1. Pull Request .....	3
3.2. issue .....	3
4. SATySF <sub>I</sub> -formatter の実装 .....	4
4.1. indent の管理 .....	4
4.2. そのまま出力する場所 .....	4
5. document の更新 .....	5
6. format .....	5

SATySF<sub>I</sub> 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。<sup>\*1 \*2</sup>

---

## 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。<sup>\*3</sup>

```
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 -o $output
```

output を指定しなかった場合、コマンドラインの標準出力に結果が表示されます。

---

## 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。

---

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 SATySF<sub>I</sub> の文法に習熟している訳ではないため、parser を元に復元するという手法によりフォーマットを実現しており、容易にビルドが失敗するようになります。(22/3/11 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。

```
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けてから、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。<sup>\*4</sup>

### 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 }
```

---

<sup>4</sup> そもそもまだ全部のテストケースが通らない .....

```
>  
"#;  
    test_tmpl(text, expect);  
}
```

---

## 4. SATySF<sub>I</sub>-formatter の実装

---

### 4.1. indent の管理

SATySF<sub>I</sub>-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。

\* block\_text \* cmd\_text\_arg \* record \* type\_record \* 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
  - tuple
  - 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

---

---

基本の indent は 4 です。(そのうち引数で管理ができるようにします)