

XPath パッケージ

The XPath Contributors

1. 基本的なコマンド

1.1. パスの組み立て

`satysfi-xpath` は、端的に言えばパスを表す独自の型である `XPath.t` 及び未完パスを表す `XPath.pre` を扱うためのライブラリです。これらは `SATySFI` における `path` 及び `pre-path` に対応します。

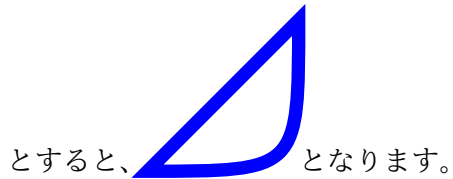
`satysfi-xpath` は `SATySFI` が持つパス操作コマンドと互換性のあるコマンドをサポートしており、`open XPath` とすることにより、`SATySFI` の提供するパス操作コマンドを置き換えることもできます (推奨はしない)。

具体的には、以下のコマンドをサポートします。

```
XPath.unite-path : XPath.t -> XPath.t -> XPath.t
XPath.shift-path : point -> XPath.t -> XPath.t
XPath.linear-transform-path : float -> float -> float -> float -> XPath.t
-> XPath.t
XPath.get-path-bbox : XPath.t -> point * point
XPath.start-path : point -> XPath.pre
XPath.line-to : point -> XPath.pre -> XPath.pre
XPath.bezier-to : point -> point -> point -> XPath.pre -> XPath.pre
XPath.terminate-path : XPath.pre -> XPath.t
XPath.close-with-line : XPath.pre -> XPath.t
XPath.close-with-bezier : point -> point -> XPath.pre -> XPath.t
XPath.stroke : length -> color -> XPath.t -> graphics
XPath.fill : color -> XPath.t -> graphics
XPath.dashed-stroke : length -> length * length * length -> color ->
XPath.t -> graphics
```

1.2. 例

```
XPath.(  
  start-path (0pt, 0pt)  
  |> line-to (2cm, 2cm)  
  |> close-with-bezier (2cm, 0cm) (2cm, 0cm)  
  |> stroke 5pt (Color.blue)  
)
```



1.3. 組み込み型との変換

`XPath.t`, `XPath.pre` を `SATySFI` に組み込みの `path`, `pre-path` に変換するため、以下のコマンドが用意されています。^{*1}

```
XPath.to-embedded-path : XPath.t -> path  
XPath.to-embedded-prepath : XPath.pre -> pre-path
```

逆に、`SATySFI` の組み込み型から `satysfi-xpath` の型に変換する方法はありません。

2. 便利な機能

ここでは、`SATySFI` の組み込み `path` 機能にはない、`satysfi-xpath` 独自の便利な機能について説明します。

2.1. 交点を求めるコマンド

交点を求める機能として以下が提供されています。

```
XPath.get-intersections : length -> XPath.t -> point list
```

¹ これらのコマンドは通常使用する必要はありませんが、他の `path` を操作するライブラリ等と連携させる場合に使用できます。

```
XPath.get-intersections-with : length -> XPath.pre -> XPath.t -> point  
list
```

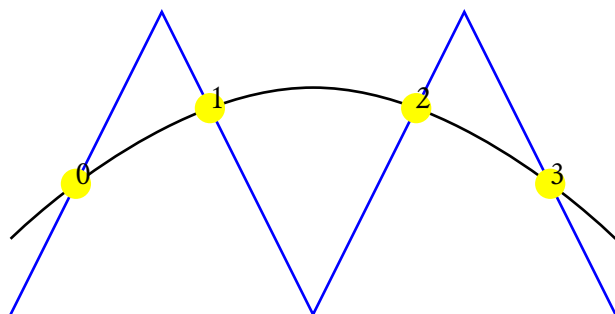
- `get-intersections delta pat...` 与えられたパス `pat` の内部の交点 (`point` 型) のリストを得ます。 `delta` は精度です。
- `get-intersections-with delta ppat pat...` 与えられた未完パス `ppat` がパス `pat` と交わる点のリストを得ます。得られる点列は `ppat` 上で順番になっています。

例えば、以下のコード

```
let pat = XPath.(  
  start-path (4cm, -1cm) |> line-to (2cm, 3cm)  
  |> line-to (0cm, -1cm) |> line-to (-2cm, 3cm)  
  |> line-to (-4cm, -1cm) |> terminate-path  
) in  
let ppat = XPath.(  
  start-path (-4cm, 0cm)  
  |> bezier-to (-4cm, 0cm) (-2cm, 2cm) (0cm, 2cm)  
  |> bezier-to (2cm, 2cm) (4cm, 0cm) (4cm, 0cm)  
) in  
let pts = XPath.get-intersections-with 0.001cm ppat pat in  
let labels = pts |> List.mapi (fun i p -> (  
  let () = show-point p |> display-message in  
  [  
    Gr.circle p 0.2cm |> fill (Color.yellow);  
    arabic i |> embed-string |> read-inline ctx |> draw-text p  
  ]  
) |> List.concat in  
XPath.(  
  List.append [  
    pat |> stroke 1pt (Color.blue);  
    ppat |> terminate-path |> stroke 1pt (Color.black);  
  ] labels  
)
```

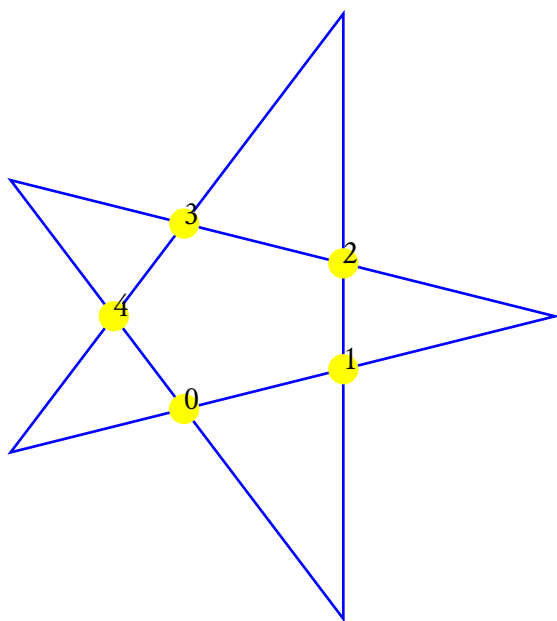
は以下のように表示されます。このとき、図中の番号 (すなわち `get-intersections-`

with が交点を出力する順番) は与えられた XPath.pre(図の曲線右向き) と同じ順番で表示されていることを確認してください。



また、get-intersections の例を示します。

```
let pat = XPath.(
  start-path (4cm, 0pt) |> line-to (-3.2cm, 1.8cm)
  |> line-to (1.2cm, -4cm) |> line-to (1.2cm, 4cm)
  |> line-to (-3.2cm, -1.8cm) |> close-with-line
) in
let pts = XPath.get-intersections 0.001cm pat in
let labels = pts |> List.mapi (fun i p -> (
  let () = show-point p |> display-message in
  [
    Gr.circle p 0.2cm |> fill (Color.yellow);
    arabic i |> embed-string |> read-inline ctx |> draw-text p
  ]
)) |> List.concat in
XPath.(
  List.append [
    pat |> stroke 1pt (Color.blue);
  ] labels
)
```



2.2. 長さに関するコマンド

ここでは、曲線の長さを取得したり、点の曲線上における位置を取得したり、長さを用いて曲線进行操作するコマンドを紹介します。

```
XPath.get-rough-length : length -> XPath.pre -> length
XPath.get-point-of-len : length -> length -> pre -> point
XPath.get-projection : length -> point -> pre -> point
XPath.get-projection-length : length -> point -> pre -> length
XPath.split : length -> length -> pre -> pre * pre
```

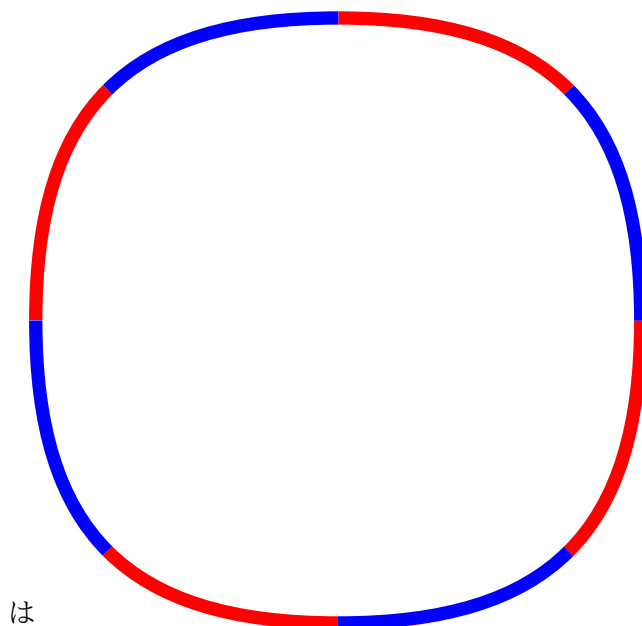
- `get-rough-length delta ppat...` 与えられた未完パス `ppat` の長さを得ます。これは厳密なアルゴリズムではなく、精度を `delta` で指定します。
- `get-point-of-len delta len ppat...` 与えられた未完パス `ppat` 上で始点から長さ `len` の地点を取得します。
- `get-projection delta p ppat...` 与えられた未完パス `ppat` 上で与えられた点 `p` に最も近い点を取得します。
- `get-projection-length delta p ppat...` 与えられた未完パス `ppat` 上で与えられた点 `p` に最も近い点の位置を表す長さを取得します。
- `split delta len ppat...` 与えられた未完パス `ppat` を長さ `len` の位置で分割します。

例えば、

```

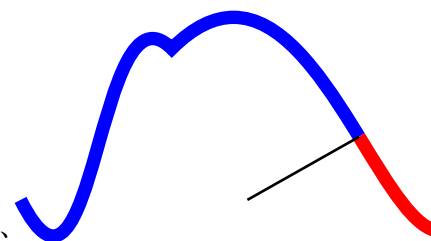
let ppat = XPath.(
  start-path (-4cm, 0pt)
  |> bezier-to (-4cm, 2.8cm) (-2.8cm, 4cm) (0pt, 4cm)
  |> bezier-to (2.8cm, 4cm) (4cm, 2.8cm) (4cm, 0pt)
  |> bezier-to (4cm, -2.8cm) (2.8cm, -4cm) (0pt, -4cm)
  |> bezier-to (-2.8cm, -4cm) (-4cm, -2.8cm) (-4cm, 0pt)
) in
let len = XPath.get-rough-length 0.001cm ppat in
let n = 8 in
range-span 0. 1. n |> List.mapi (fun i (a, b) -> (
  let (_, ppat) = ppat |> XPath.split 0.01cm (len *' a) in
  let (ppat, _) = ppat |> XPath.split 0.01cm (len *' (b -. a)) in
  let clr = if i mod 2 == 0 then Color.red else Color.blue in
  ppat |> XPath.terminate-path |> XPath.stroke 5pt clr
))

```



は

と表示されます。また、



3. 今後追加する機能

4. 内部実装