ページ分割処理周辺の不具合等の報告

Godai Takashina

1. 概要

このレポートでは，ページ分割処理や margin の処理に関わる部分に関する不具合の報告や改善方法の提案などをいくつかさせていただきたいと思っています．はじめレポジトリ内の issue として報告させていただこうと思ったのですが，量が多くなってしまったので一度 Slack でご相談させていただければと思いました．至らない部分もあるかと思いますが，少しでもご参考になれば幸いに思います．

2. 不具合等の報告と対処

ソースファイルとしては基本的に `src/backend/pageBreak.ml` に関する部分に言及するので，関数名等はこのファイル内のものと考えてください．

2.1. assertion failure による強制終了の発生

2.1.1. 概要

`pageBreak.ml` 内の関数 `chop_single_column` 内に存在する assertion が失敗することによって組版が強制終了されてしまう問題が発生しています．

2.1.2. 問題の詳細

`chop_single_column` は文書内の `block-boxes` に対応するリスト (厳密には `pb_vert_box list`) を適切な高さのページに切り分ける処理を行うページ分割の主要部分になっています．補助的な関数として `normalize_after_break` が定義されていて，おおまかには手動で挿入した `clear-page` と一致する箇所で通常のページ分割が発生した場合に，改ページが重複しないように手動で挿入した `clear-page` をひとつ削除するという役割となっているようです．

この `chop_single_column.normalize_after_break` 内には 2 箇所 `assert false` による assertion が挿入されているのですが，ユーザーの作成した文書ファイルの入力によってどちらの assertion も失敗する可能性があることがわかりました．すでに報告されている

issue #323 も一方の assertion の失敗に関するものです.

1 つ目の assertion (108 行目) は, 行分割の候補地点の直後に `PBVertSkip`, `PBHookPageBreak` が並んだ場合に到達します. `PBVertSkip` は段落間の margin や `block-skip` により挿入され, `PBHookPageBreak` は `hook-page-break-block` により挿入されるものです.

2 つ目の assertion (117 行目) は, 行分割の候補地点の直後に `PBVertSkip`, `PBClearPage`, `PBClearPage` (`clear-page` に対応) が並んだ場合に到達します.

2.1.3. 再現

次の入力は 1 つ目の assertion を失敗させます.

```
let text-width = 440pt
let text-height = 630pt
let-inline ctx \math m = embed-math ctx m
let ctx = get-initial-context text-width (command \math)

let bb-main =
  let ib = read-inline ctx {some text} in
  line-break true true ctx (ib ++ inline-fil)

let vhook = hook-page-break-block (fun _ _ -> ())
let bb = bb-main +++ (block-skip 1pt) +++ vhook

in

page-break A4Paper
  (fun _ -> (| text-height = text-height; text-origin = (0pt,0pt); |))
  (fun _ -> (| header-content = block-nil; header-origin = (0pt,0pt);
               footer-content = block-nil; footer-origin = (0pt,0pt); |))
  bb
```

次の入力は 2 つ目の assertion を失敗させます.

```
let text-width = 440pt
let text-height = 630pt
let-inline ctx \math m = embed-math ctx m
```

```
4  let ctx = get-initial-context text-width (command \math)
5
6  let bb-main =
7    let ib = read-inline ctx {some text} in
8    line-break true true ctx (ib ++ inline-fil)
9
10 let bb = bb-main +++ (block-skip 1pt) +++ clear-page +++ clear-page
11
12 in
13
14 page-break
15   A4Paper
16   (fun _ -> (| text-height = 630pt; text-origin = (0pt,0pt) |))
17   (fun _ -> (| header-content = block-nil; header-origin = (0pt,0pt);
18                footer-content = block-nil; footer-origin = (0pt,0pt); |))
19   bb
```

2.1.4. 対応

これらの assertion を導入した経緯をあまり知らないので判断しづらいのですが，assertion 自体は削除してしまっても問題ないのではないかと思っています．現状の `normalize_after_break` の挙動は正規表現風に書くと

`(PBVertSkip)*(PBClearPage)?(PBVertSkip)?`

の並びを削除するような挙動になっていると思いますが，この挙動が妥当であるかという点を含めて検討する必要があるかもしれません．`PBHookPageBreak` の assertion については，ページ分割候補点直後の `PBHookPageBreak` が改ページ前のページに含まれるようにしておくなどの処理が想定されていたような感じもあり，そうであるとすれば別の箇所の修正で対応する必要がありそうです．

2.2. hook-page-break-block 周辺の垂直方向の空白

2.2.1. 概要

`hook-page-break-block` の上下の margin が消失する問題を確認しました．

2.2.2. 問題の詳細

まず，前提として `hook-page-break-block f` は `f` による副作用を伴う以外は基本的に `block-nil` と同様に振る舞うべきだと考えています．そのため，`hook-page-break-block` を入れることで段落間の空白が変わってしまうというのは望ましくない挙動と言えると思います．しかし，段落間に `hook-page-break-block` を挿入した場合には margin が消失して段落がくっついてしまうような出力になるようです．

2.2.3. 再現

再現用のコードを下に示します．

```
@require: stdjareport

let-block ctx +vhook = hook-page-break-block (fun _ _ -> ())

in

document (|
  title = {Test of vertical spacing around hook-page-break-block};
  author = {sankantsu};
|) '<
  +chapter {No hook} <
    +p {
      First paragraph
    }
    +p {
      Second paragraph
    }
  >
  +chapter {Insert vhook} <
    +p {
      First paragraph
    }
    +vhook;
    +p {
      Second paragraph
    }
```

```
27      >
28    >
```

出力結果を下に示します.

Test of vertical spacing around hook-page-break-block

sankantsu

1. No hook

First paragraph

Second paragraph

2. Insert vhook

First paragraph
Second paragraph

2.2.4. 対応

主に `squash_margins` の実装の修正によって対処できます．また，修正の実装方法によっては段落間に前節で説明した assertion を失敗させるパターンが発生しうるので，対処については合わせて考える必要があるように思います．

2.3. 最終ページでのページ分割処理

2.3.1. 概要

最終ページでのページ分割処理が通常より overfull 気味になる現象を確認しました．

2.3.2. 問題の詳細

最終ページのページ分割は，block-box 列の終端に達した時点でその時点で残っている block-box 列をすべて最終ページに置くことで行います．現在の実装では，block-box 列の最終要素の直前でのページ分割が実質的に不可能になっているようです．そのため，最終ページではちょうどいいページ分割地点よりも 1 つ多く block-box を足したぶんのページが出力されることがあります．

2.3.3. 再現

本質から外れる部分のコードが長くなってしまっていますが，次のようなコードを例に考えます．

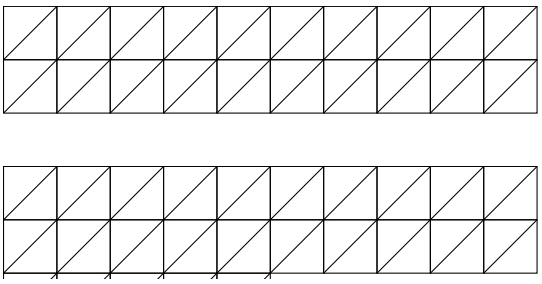
```
@require: gr
@require: color
@require: list

let num-char = 45

let char-size = 10pt
let text-width = char-size *' 10.
let-inline ctx \math m = embed-math ctx m
let ctx = get-initial-context text-width (command \math)
            |> set-leading 0pt
            |> set-min-gap-of-lines 0pt

```

```
% pseudo-character ((w,h,d) = (char-size,char-size,0pt))
let gr-char =
  let thickness = 0.1pt in
  let draw = stroke thickness Color.black in
  let gr (x,y) = [
    draw (Gr.rectangle (x,y) (x +' char-size, y +' char-size));
    draw (Gr.line (x,y) (x +' char-size, y +' char-size));]
  in
  inline-graphics char-size char-size 0pt gr

let dscr = discretionary 0 inline-nil inline-nil inline-nil
let ib-unit = gr-char ++ dscr % can break after each character

let-rec repeat n x =
  if n == 0 then inline-nil else x ++ repeat (n - 1) x

let ib = repeat num-char ib-unit
let bb = line-break true true ctx (ib ++ inline-fil)

in

page-break
  (UserDefinedPaper (text-width,char-size *' 2. +' 1pt))
  (fun _ -> (| text-height = (char-size *' 2.); text-origin = (0pt,0pt) |))
  (fun _ -> (| header-content = block-nil; header-origin = (0pt,0pt);
               footer-content = block-nil; footer-origin = (0pt,0pt); |) )
  bb
```

やや複雑になっているので中身について少し説明すると，ちょうど 10pt × 10pt の大きさの"文字" gr-char を行分割候補地点 (discretionary) と交互に並べ，gr-char がちょうど 10 個並んだ点で行分割が起こるようにページ幅を調整します． ページの高さは，gr-char を並べてできる行のちょうど 2 行分となるようにしています．(ページの表示範囲の高さは少しだけ大きくしています．)

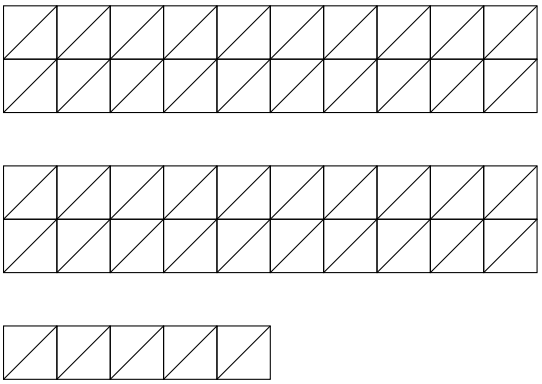
gr-char を並べる数 (num-char) を例えば 45 としたとき，行分割により 5 行に分割されます．1 ページに収まるのは 2 行分なので最後の 1 行は 3 ページ目に置かれるのが望ましいと考えられますが，実際には 2 ページ目に 3 行分置かれて overfull します．

出力結果を下に示します．



2.3.4. 対応

`squash_margins` の実装内の match 文における空リストのケースの返り値を書き換えることでページ分割位置を想定通りに変えることができました．具体的には，block-box 列終端を改ページ不可位置とみなしてページ分割をしていたのを，改ページ可能位置とみなすように変更します．空の最終ページが発生するなど意図しない挙動が起きないか詳細なテストはできていませんが，おそらく問題無いと思います．先程の例で，修正後の出力を下に示します．



2.4. paragraph のページ分割の実装詳細の変更

2.4.1. 概要

おそらく最終的な出力には影響しないと考えていますが，意味的に疑問のある箇所があったので変更を提案します．

2.4.2. 詳細

修正の diff を下に示します．

```
--- a/src/backend/pageBreak.ml
+++ b/src/backend/pageBreak.ml
@@ -177,7 +177,7 @@ let chop_single_column (pbinfo : page_break_info) (ar
 ↪ea_height : length) (pbvbls
             | Breakable ->
                 let ansA =
                   {
-                    division = Inside(prev.solid_body, normalize_after_
 ↪break pbvblst);
+                    division = Inside(bodyA, normalize_after_break pbvb
 ↪tail);
                     footnote = footnoteA;
                     height   = hgtA;
                     badness  = badnsA;
```

footnote の内容 (`footnoteA`) やページの高さ (`hgtA`) が現在行を含む計算である点から段落内の行直後のページ分割位置候補を生成するコードであるように思いますが，分割箇所の指定だけは現在行の直前になっているように見えます．おそらく結果的には意図通りのページ分割候補地点が block-box 列の直後の要素を走査するときに生成されているようなので結果的に問題にはなっていないようですが，意味的には修正したほうが適切に見えます．

2.5. `pageBreak.ml` 内のコメント行の削除

`pageBreak.ml` のソースファイルの最後 220 行がすべてコメント行になっています．特に残しておく必要がなければ，削除してしまったほうが可読性や編集しやすさの点で良いと思います．テスト用途のようにも見えますが，必要であれば別ファイルへの移動なども含めて検討してはいかがでしょうか？

3. その他提案など

主に垂直方向のスペーシングについて妄想に近い提案を書かせてもらいました．margin 周辺の仕様について v0.1.0 で見直す予定があるということはお聞きしたことがあるので，何か少しでもご参考になれば幸いです．

3.1. block-skip の挙動について

段落間に block-skip を挿入した際の垂直方向の空白の挙動が個人的にはやや非直観的に感じられます．例えば，段落の上下 margin が 18pt になっているとしたとき，普通に段落を並べる限りでは段落間の空白は 18pt です．ここで例えばある箇所の段落間の空白を 10pt だけ広げて 28pt にしたいと考えたとします．block-skip 10pt と書きたくなるかと思うのですが，実際には上下 margin の重なりが消えるぶんを考慮して $28 - 18 \times 2 = -8$ という計算をして block-skip -8pt を挿入する必要があります．空白を増やしたいのに負のスペースを入れるというのは少し直観に反するように思えます．

改善のためのアイデアとして思いつくものをいくつか挙げます．

- 段落間の block-skip の仕様の変更
- get-paragraph-margin プリミティブの導入

1 つ目の提案は，段落間の block-skip で挿入される空白の大きさについて，現状の仕様である

$$(\text{上の段落の下方 margin}) + (\texttt{block-skip}\ \text{の大きさ}) + (\text{下の段落の上方 margin})$$

から

$$\max\big((\text{上の段落の下方 margin}), (\text{下の段落の上方 margin})\big) + (\texttt{block-skip}\ \text{の大きさ})$$

に変更するというものです．この変更自体は pageBreak.ml 内の squash_margins の実装を変更することで比較的容易に実現することができると考えています．block-skip が連続した場合のページ分割などがやや非自明な感じもしますが，段落間かどうかに関わらずそもそも連続する空白要素はすべてつなげて 1 つに圧縮してしまうという方針もありかもしれません．

2 つ目の get-paragraph-margin は現在の context に設定された上下 margin を取得することを想定したものです．margin の大きさを文書ソース内から取得できれば，先ほどの例のように段落間の空白を xx pt 広げたいというような要求があったときに実際に挿入すべき block-skip の大きさを求める処理をライブラリ等のレベルで実装できます．ただし，空白挿入箇所の上の段落と下の段落で設定されてい margin の値が変わっているといった状況下ではこのような計算はできなさそうです．とはいえ，他の用途でも margin の値を取得するようなプリミティブが役立つ場面があるような気はします．

3.2. margin の見直し・廃止

前節で書いた `block-skip` の非直観的な挙動も margin 自体の仕様の影響によるところが大きいです． いっそ margin 自体を廃止してしまって単純な縦方向スキップ (あるいはグルー) のみにしてしまうというのもありうるのかもしれません．現在のような margin を廃止したとしても，`get-last-block-skip` のようなプリミティブを追加して直前に挿入した垂直方向の空白の大きさを取得できるようにすれば現状の margin に近い挙動もライブラリやドキュメントクラスのレベルで実現できるのではないかと思います．TeX はこれに近い仕様で，`\lastskip` で直前に挿入されたグルーを取得します．

また，margin を廃止した場合 margin に埋め込まれていたページ分割可否の情報も一緒に消えるので， ページ分割不可の空白を入れる `unbreakable-block-skip` のようなプリミティブが必要になるかもしれません．