

LuaTeX-ja 和文処理グループについて

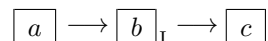
2011/6/1

本文書では、LuaTeX-ja が (現時点において) 和文処理に関わる glue/kern をどのように挿入するか内部処理について説明する。

予備知識

説明に入る前に、段落や hbox の中身は、TeX の内部では node 達による リストとして表現されていることに注意する。node の種類については、*The LuaTeX Reference* の第 8 章を参照して欲しい。代表的なものを挙げると、

- *glyph_node*: 文字 (合字も含む) を表現する。和文処理グループを挿入する際には、既に各 *glyph_node* が欧文文字のものか和文文字のものか区別がついている。また、しばしば *glyph_node p* と、その表す文字の文字コード *p.char* とを同一視する。
- *glue_node*: glue を表す。
- *kern_node*: kern を表す。各 *kern_node* には *subtype* という値があり、次の 3 種類を区別できるようになっている。
 - 0: 欧文用 TFM 由来
 - 1: 明示的な `\kern` か、イタリック補正 (`\/`) によるもの
 - 2: `\accent` による非数式アクセント用文字の左右位置調整のためのもの
- *penalty_node*: penalty を表す。
- *hlist_node*: hbox (水平ボックス) を表す。
- 次のように、node がどのように連続しているかを表すことにする。



右下についている添字は、LuaTeX-ja においてその node の役割を区別するためにつけられた値 (*jtype* と呼ぼう) であり、次のようになっている。

I: イタリック補正由来の kern T: `\[x]kanjiskip` に置換されうる kern
J: JFM 由来の glue/kern K: 禁則処理用 penalty
E: 「行末」との間に入る kern KS: `\kanjiskip` 用 glue
XS: `\xkanjiskip` 用 glue

- `jaxspmode` のようなサンセリフ体で、`\ltjsetparameter` で設定可能なパラメタ値を表す。
- タイプライタ体の `\kanjiskip`, `\xkanjiskip` は、それぞれ「和文間空白」「和欧文間空白」の意味で抽象的に用いている。
- nil 値は \emptyset と書く。

JFM 由来グルーの挿入 (luatexja-jfmglue.lua)

JFM 由来グルーの処理は、「連続する 2 つの node の間に何を入れるか」という単位で行われる。そのため、

- node 生成を伴わないもの (グループ境界, `\relax` 等) は全て無視される。
- 一方, node 生成を伴うものは全て「透過しない」。例えば, 次のソースにおいて, 閉じ括弧と開き括弧の間に入る物は, 左と右とで異なる:

```
) ( ) \hbox{}
```

- 「現在位置での JFM 由来グルーの挿入抑制」を行う `\inhibitglue` は, 内部では専用の whatsit node (subtype = 44, user_id = 30111) を作ることに よって実装している。これらの node は, 「現在位置で挿入しない」というフラグを立てるためだけに存在するものであって, 挿入処理中に全て削除される。

以下, q, p を連続する node とする。

2 つの和文文字の間

この場合, グルー挿入に関係する量は次の通りである。これら 3 つの量の値によって, q と p の間に何が挿入されるかが決定される。これらの記号は他の場合にも用いる。

- g : JFM で指定された, q と p の間に入る glue/kern。JFM で規定されていないときは \emptyset と書こう。両ノードで使われている JFM が異なる時の g の決定方法は, 後に記述する。
- w : JFM で指定された, 「 q の直後で改行が行われた場合, q と行末の間に入るカーン量」の値。また, $g - w$ で, g の自然長を w だけ減算した glue/kern を表すことにする (g が glue ならばこの node は glue, g が kern ならば kern)。
- P : q に対する行末禁則用ペナルティ (post-break penalty) と, p に対する行頭禁則用ペナルティ (pre-break penalty) との和。どちらも 設定されていないときは 0 となる。

なお, 間に `\inhibitglue` による指定があった場合, $g = \emptyset, w = 0$ として処理される。設計方針としては,

- JFM 由来で入るものが kern の場合, この場所では行分割は許さない。
- そうでない場合, (penalty の値 P があるが) この場所での行分割は可能である。

である。さて, 次が実際の場合わけである:

1. $w \neq 0, g = \emptyset$ のとき

$$\boxed{q} \longrightarrow \boxed{\text{kern } w}_{\text{E}} \longrightarrow \boxed{\text{penalty } P}_{\text{K}} \longrightarrow \boxed{\text{kern } -w}_{\text{T}} \longrightarrow \boxed{p}$$

この $\boxed{\text{kern } -w}_{\text{T}}$ は, 「 q と p の間で行分割されないときは間に何の glue/kern もないように見える」ために 挿入されたものである。次のステップで `\[x]kanjiskip` の挿入が行われる時に, この node は `\[x]kanjiskip` 用の glue に置換される。

2. $w \neq 0, g \neq \emptyset$ のとき:

$$\boxed{q} \longrightarrow \boxed{\text{kern } w}_{\text{E}} \longrightarrow \boxed{\text{penalty } P}_{\text{K}} \longrightarrow \boxed{g - w}_{\text{J}} \longrightarrow \boxed{p}$$

3. $w = 0, g$: kern のとき

$$\boxed{q} \longrightarrow \boxed{g}_{\text{J}} \longrightarrow \boxed{p}$$

4. $w = 0, g$: glue のとき

$$\boxed{q} \longrightarrow \boxed{\text{penalty } P}_{\text{K}} \longrightarrow \boxed{g}_{\text{J}} \longrightarrow \boxed{p}$$

5. $w = 0, g = \emptyset, P \neq 0$ のとき

$$\boxed{q} \longrightarrow \boxed{\text{penalty } P}_{\text{K}} \longrightarrow \boxed{p}$$

6. $w = 0, g = \emptyset, P = 0$ のとき

$$\boxed{q} \longrightarrow \boxed{p}$$

なお、両ノードで使われている JFM が異なる時の g の決定方法であるが、

1. g_L を、 q に使用されている JFM における、「 q と文字 'diffmet'」の間に入る glue/kern の値とする。
2. g_R を、 p に使用されている JFM における、「文字 'diffmet' と p 」の間に入る glue/kern の値とする。
3. 両方から、実際に入る g の値を計算する。
 - g_L, g_R の少なくとも片方が \emptyset のときは、 \emptyset でない方をそのまま採用する。
 - 両方とも \emptyset でない場合は、differentjfm の値にそって g の値を計算する。

和文文字と（和文文字，kern 以外の node）の間

「和文文字の間」の場合に対して、以下が異なる：

- g は、 q に使用されている JFM における、「 q と文字 'jcharbdd'」の間に入る glue/kern の値である。
- p が penalty でない場合は、いつもこの位置で行分割できるようにするため、case 6 ($w, P = 0, g = \emptyset$) の場合にも、 q と p の間には 0 という値の penalty が入る。即ち、次のようになる。

$$\boxed{q} \longrightarrow \boxed{\text{penalty } 0}_{\text{K}} \longrightarrow \boxed{p}$$

（和文文字，kern 以外の node）と和文文字の間

この場合も、基本的には「和文文字の間」と似ているが、以下が異なる：

- g は、 p の JFM における、「文字 'jcharbdd' と p 」の間に入る glue/kern の値である。
- 常に $w = 0$ である。
- いつもこの位置で行分割できるようにするため、case 6 ($w, P = 0, g = \emptyset$) の場合にも、 q と p の間には 0 という値の penalty が入る。

即ち、次の 3 通りになる。

1. g : kern のとき

$$\boxed{q} \longrightarrow \boxed{g}_{\text{J}} \longrightarrow \boxed{p}$$

2. g : glue のとき

$$\boxed{q} \longrightarrow \boxed{\text{penalty } P}_{\text{K}} \longrightarrow \boxed{g}_{\text{J}} \longrightarrow \boxed{p}$$

3. $g = \emptyset$ のとき

$$\boxed{q} \longrightarrow \boxed{\text{penalty } P}_{\text{K}} \longrightarrow \boxed{p}$$

和文字と kern の間, kern と和文字の間

和文字の後に kern が続いた場合,あるいは kern の後に和文字が続いた場合, この間で行分割はできないものとしている. そのため, 以下の 3 ケースに限られる:

1. g : kern のとき

$$\boxed{q} \rightarrow \boxed{g}_J \rightarrow \boxed{p}$$

2. g : glue のとき

$$\boxed{q} \rightarrow \boxed{\text{penalty 10000}}_K \rightarrow \boxed{g}_J \rightarrow \boxed{p}$$

3. $g = \emptyset$ のとき

$$\boxed{q} \rightarrow \boxed{p}$$

なお, ここでの g は,

- q が kern だった場合は, p の JFM における, 「'jcharbdd' と p 」の間に入る glue/kern の値.
- p が kern だった場合は, q の JFM における, 「 q と 'jcharbdd'」の間に入る glue/kern の値.

要検討の箇所

私が推測するに, 欧文では,

- 単語内ではフォントは変わらない;
- 単語内では, 明示的に/ハイフネーションにより挿入された discretionary break 以外では行分割がおきない;

という事情があるため, TFM 由来の kern や合字処理は (node を生成しないもの以外は) 何も透過しないという状態になっているものと思われます.

そのため, JFM グルー等の仕様を考える場合, 欧文でいう「単語」に対応するようなものは何か, というのを考える必要があります. 現実装では, 素直に欧文の合字処理と同様のものであると見え, 透過する node はない, という仕様になっています. しかし, `\[x]kanjiskip` の処理と共通にしてしまうというのも 考え方によってはありかもしれません.

• イタリック補正の kern の周囲

例えば, `jfm-ujis.lua` では, 「'jcharbdd」は文字クラス 0 であるため, 今の実装では, 「) \ (」という入力からは, 次の node の並びを得る:

$$\begin{aligned} \boxed{)} &\rightarrow \boxed{\text{penalty 10000}}_K \rightarrow \boxed{\text{glue 0.5 zw}_{-0.5}}_J \\ &\rightarrow \boxed{\text{kern \ /}} \rightarrow \boxed{\text{penalty 10000}}_K \rightarrow \boxed{\text{glue 0.5 zw}_{-0.5}}_J \rightarrow \boxed{(} \end{aligned}$$

一方, イタリック補正を JFM 由来グルーが透過するとしたならば, 当然

$$\boxed{)} \rightarrow \boxed{\text{kern \ /}} \rightarrow \boxed{\text{glue 0.5 zw}_{-0.5}} \rightarrow \boxed{(}$$

となる (実際の組版イメージでは, 「斜め」(「斜め」). どちらにするか?

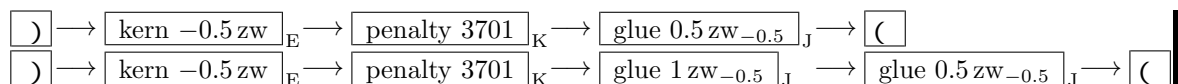
• penalty の周囲

これも, 例えば次の設定の下では, 「) \ penalty1701 (」からは以下を得る:

- 「)」と行末の間に $-0.5 zw$ だけ kern を入れる.
- 「)」 「(」の行頭/行末禁則用 penalty の値はどれも 1000.

$$\begin{aligned} \boxed{)} &\rightarrow \boxed{\text{kern } -0.5 zw}_E \rightarrow \boxed{\text{penalty 1000}}_K \rightarrow \boxed{\text{glue 1 zw}_{-0.5}}_J \\ &\rightarrow \boxed{\text{penalty 1701}} \rightarrow \boxed{\text{penalty 1000}}_K \rightarrow \boxed{\text{glue 0.5 zw}_{-0.5}}_J \rightarrow \boxed{(} \end{aligned}$$

例えば `penalty` を合算することとした場合，上の入力例では本来「)」「(」の間に 1701 の `penalty` があるのだから，



のどちらか(上は `penalty` を透過する場合，下は透過しない場合)にするのが良いと思われます。

• discretionary break の取り扱い

`discretionary break` (`disc_node`) は，行分割時の行末の内容 $\langle pre \rangle$ ，行頭の内容 $\langle post \rangle$ ，それに行分割しないときの内容 $\langle no_break \rangle$ の 3 つをリストの形で持っている。`linebreak.w` を見る限り，`LuaTeX` でも $\langle pre \rangle$ ， $\langle post \rangle$ ， $\langle no_break \rangle$ の中身には `glue` や `penalty` を許容していないようだ。

現行の実装では， $\langle pre \rangle$ ， $\langle post \rangle$ ， $\langle no_break \rangle$ のどれも，和文フォントへの置換の段階からして行われていない(だから中身は全部欧文扱いとなる)。単純にサボってました^^; $\langle pre \rangle$ ， $\langle post \rangle$ ， $\langle no_break \rangle$ の中身に `glue` や `penalty` が許容されないことから，これらに対する和文用処理の方法として，次の 2 種類が挙げられる。私は前者が良いのではないかと思っているのだが.....

- (現行のまま) `discretionary break` の中身に和文文字はないものと想定する。例えば $\langle pre \rangle$ の中身に和文文字を入れたい場合は， $\langle pre \rangle$ の中身全体を必ず `hbox` で括ることとする。
- 「`glue` を挿入」を全部「自然長だけを取り出した `kern` を挿入」に置き換え，普段の和文処理グルー挿入処理を流用する。

• 数式の取り扱い

まだ数式中に和文文字が (`hbox` でカプセル化されることなく) 出現することは想定していない。実用的にはそれでも十分だと思うが，もし `pTeX` のように単に

`$aあa$`

など書いても和文文字が出力されるようにするとなれば，「数式リストから変換されてできた水平リストでは，和文処理グルーの挿入処理を無効とする」ようにしないとイケないだろう。

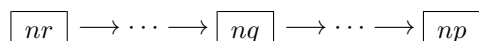
\[x]kanjiskip の挿入

現実装の `\[x]kanjiskip` の挿入の方針として，

- JFM グルーが挿入されていないところに「標準の空き量」として挿入する。
- 実際の段落/`hbox` の内容に即して，組版イメージの見た目に関係のないところは透過する。

処理の概要

`\[x]kanjiskip` 挿入処理では，次の 3 つの `node` を用いている。



- `nr` と `np` の間に `\[x]kanjiskip` を挿入しようとする。
- 実際に `node` の形で挿入しようとする場所は `nq` の直後である。
- `nr`，`nq` は異なる `node` とは限らない。
- `np` はリストの先頭から末尾までループで渡る。その過程で `nr`，`nq` を適宜更新し，実際の `node` 挿入処理を行っている。

- 厳密には、コード中では *nr* という変数は使っていない。代わりに使われているのは、
 - *insert_skip* ∈ {*no_skip*, *after_schar*, *after_wchar*}: 「node *nr*」の種類を表す：
 - *no_skip*: 「node *nr*」の後ろ (*nr* と *np* の間) に `\[x]kanjiskip` が入ることはない。
 - *after_schar*: 「node *nr*」を、欧文文字(のに入った *glyph_node*)であり、かつ `alxspmode` パラメータの指定により「*nr* の後ろに `\xkanjiskip` の挿入を許可する」ようなものとみなす。
 - *after_wchar*: 「node *nr*」を和文文字(のに入った *glyph_node*)とみなす。
 - *nrc*: 「node *nr* の文字コード」を表す。
 - *nrf*: 「*nr* のフォント」を表す。
 - *nr_spc*[1]: 「node *nr*」における `autospadding` (`\kanjiskip` の自動挿入を行うか否か) の設定値。
 - *nr_spc*[2]: 「node *nr*」における `autoxspacing` (`\kanjiskip` の自動挿入を行うか否か) の設定値。

nrc, *nrf* の値は、*insert_skip* = *after_wchar* のときのみ用いられる。*insert_skip* = *no_skip* のときには、それだけで情報は十分であるから、*nr_spc*, *nq* の値も用いられない。

ループの中で、以下の場合には *nr* は変化せず、*nq* ← *np* となる。つまり、これらの node に対して `\[x]kanjiskip` は透過する：

- *np* が `penalty` の場合
- *np* が `subtype = 0` の `kern` (TFM 由来) の場合。
- *np* が `subtype = 1` の `kern` (つまり、明示的 `kern` がイタリック補正由来) であって、*jtype* が I (イタリック補正), E (行末との間), T (一時的) であった場合。後者 2 つは JFM グルーの挿入で入るものなので、ユーザは「イタリック補正は透過」と考えればよい。
- *np* が `insertion`, `mark`, `\vadjust`, `whatsit` の node である場合。これらは水平リストからは消え去る運命にある。

np が文字 (*glyph_node*) の場合
この場合がやはり一番基本となる。

1. *insert_skip* = *after_schar*, *np*: 和文文字の場合

前に書いたように、「node *nr*」は(直後に `\xkanjiskip` の挿入が許可されている)欧文文字とみなされている。そのため、「node *nr*」と *np* の間に `\xkanjiskip` の入る条件は以下である。

- 文字 *np* に対する `jaxspmode` パラメータの指定において「直前への `\xkanjiskip` の挿入が許可」されている。
- 「node *nr* における」`autoxspacing` パラメータの値 (*nr_spc*[2]) が、*np* における `autoxspacing` の値の少なくとも一方が真である。

まず、実際に入る `\xkanjiskip` の量 *g* を次の方法で決定する：

- `\xkanjiskip` パラメータの自然長が `\maxdimen` でない場合、`\xkanjiskip` パラメータの値をそのまま採用する。
- `\xkanjiskip` パラメータの自然長が `\maxdimen` の場合は、*np* で使われている JFM に設定されている `\xkanjiskip` の量を用いる。
- 上の 2 つのどれでもない場合、`fallback` として 0 を用いる。

要検討：既に JFM グルー挿入処理で和欧文間の行分割は可能としているので 0 を挿入する意味はない？

次にこのようにして決定された g を実際に挿入する：

- ほとんどの場合， g の値をもつ glue を nq の直後に挿入する．

$$\boxed{nr} \rightarrow \cdots \rightarrow \boxed{nq} \rightarrow \boxed{\text{glue } g}_{XS} \rightarrow \cdots \rightarrow \boxed{np}$$

- np の直前が $jtype = T$ な node の場合，その node に g の分だけ自然長/伸び/縮み量を加算する．

$$\begin{array}{c} \boxed{nr} \rightarrow \cdots \rightarrow \boxed{nq} \rightarrow \cdots \rightarrow \boxed{\text{glue } h}_T \rightarrow \boxed{np} \\ \downarrow \\ \boxed{nr} \rightarrow \cdots \rightarrow \boxed{nq} \rightarrow \cdots \rightarrow \boxed{\text{glue } g + h}_{XS} \rightarrow \boxed{np} \\ \\ \boxed{nr} \rightarrow \cdots \rightarrow \boxed{nq} \rightarrow \cdots \rightarrow \boxed{\text{kern } k}_T \rightarrow \boxed{np} \\ \downarrow \\ \boxed{nr} \rightarrow \cdots \rightarrow \boxed{nq} \rightarrow \cdots \rightarrow \boxed{\text{glue } g + k}_{XS} \rightarrow \boxed{np} \end{array}$$

最後に，次のループに移るために，次の処理を行う：

- $nq \leftarrow np$, nr_spc の設定, $nrf \leftarrow np.\text{font}$, $nrc \leftarrow np.\text{char}$
- $np \leftarrow \text{next}(np)$
- $insert_skip \leftarrow \text{after}_w\text{char}$

2. $insert_skip = \text{after}_w\text{char}$, np : 欧文文字の場合

前に書いたように，「node nr 」は和文文字とみなされている．そのため，「node nr 」と np の間に $\backslash\text{xkanjiskip}$ の挿入が起こるための条件は 次の 3 条件が満たされていることである：

- nrc 番の和文文字に対する jaxspmode パラメータの設定で，「直後への $\backslash\text{xkanjiskip}$ 挿入」が許可されている．
- np の文字（もし np が合字であれば，合字の構成要素の最初の文字）に対する alxspmode パラメータの設定で，「直前への $\backslash\text{xkanjiskip}$ 挿入」が許可されている．
- nr における autoxspacing パラメータの値 ($nr_spc[2]$) が，「node nr 」における autoxspacing の値の少なくとも一方が真である．

この後，実際に $\backslash\text{xkanjiskip}$ の量を計算し，node の形で実際に挿入するところは，量の決定のところでは np の代わりに nrf を用いる以外は同じである．最後の，次のループに移るための処理では，次が行われる．

- $nq \leftarrow np$, $nrc \leftarrow np.\text{char}$, nr_spc の設定
- $np \leftarrow \text{next}(np)$
- $insert_skip$ の設定．

$insert_skip \leftarrow \text{after}_s\text{char}$ となるのは， np の文字（もし np が合字であれば，合字の構成要素の末尾の文字）における alxspmode パラメータの設定で，「直後への $\backslash\text{xkanjiskip}$ 挿入」が許可されている場合である．そうでないときは， $insert_skip \leftarrow \text{no_skip}$ となる．

3. $insert_skip = \text{after}_w\text{char}$, np : 和文文字の場合

この場合は $\backslash\text{xkanjiskip}$ の代わりに $\backslash\text{kanjiskip}$ を挿入することとなる． jaxspmode , alxspmode のように「直前/直後への $\backslash\text{kanjiskip}$ 挿入許可の制御」を行うパラメータは存在しない．「autospacing で自動挿入が禁止される」とマニュアルでは言っているが，それは挿入する $\backslash\text{kanjiskip}$ の量を一時的に 0 にしているだけで，「node nr 」と np の間には常に $\backslash\text{kanjiskip}$ が入ることには変わりはないことに注意．

実際に入る $\backslash\text{kanjiskip}$ の量 g は次の方法で決定される：

- nr における autospacing の値 ($nr_spc[1]$) が， np における autospacing の値が共に偽なら， $g = 0$ ．

- kanjiskip パラメータの自然長が `\maxdimen` でない場合， kanjiskip パラメータの値をそのまま採用する．
- xkanjiskip パラメータの自然長が `\maxdimen` の場合は， `np` で使われている JFM に設定されている `\kanjiskip` の量を用いる：
 - 「node *nr*」で使用されている JFM と， `np` で使用されている JFM それぞれに `\kanjiskip` の値が設定されている場合，……．
- 上の 3 つのどれでもない場合， `fallback` として 0 を用いる．

`g` を実際に挿入するところは，今の場合も 1. の場合と変わらない．次のループに移るために `ng` 等の設定処理も，1. と同じである．

`np` が `hbox` の場合

`\[x]kanjiskip` は，垂直変位が 0 である（即ち，`\raise`，`\lower` により上下に移動されていない）`hbox` の境界を跨ぐ．

1. `hbox` 内の「最初の node」`first_char` と「最後の node」`last_char` を探索する．この探索は，次を透過する：
 - 垂直変位が 0 である `hbox` の境界（但し，空 `hbox` は透過しない）．
 - `insertion`，`mark`，`\adjust`，`whatsit`，`penalty` 用の node ．
 - 「最初の node」「最後の node」それぞれにいえることだが，文字 (`glyph_node`) でない場合は `first_char`，`last_char` はそれぞれ \emptyset となる．

前者の具体例として，例えば，次の入力を考える．

あ`\hbox{a}`い`\hbox{\hbox{b}\hbox{}}`う`\hbox{c}`え`\hbox{\hbox{d}}`お

すると，

1. 「あ」「い」の間の `hbox` → `first_char` = 「a」，`last_char` = 「a」
2. 「い」「う」の間の `hbox` → `first_char` = \emptyset ，`last_char` = \emptyset
3. 「う」「c」の間の `hbox` → `first_char` = \emptyset ，`last_char` = \emptyset
4. 「え」「お」の間の `hbox` → `first_char` = 「d」，`last_char` = 「d」

となる．

2. `np` の前に `\[x]kanjiskip` を挿入するか否か，あるいは実際に挿入する量の決定は，`first_char` に対しての処理をそのまま適用する．つまり，上の例では
 - 「あ」と「a」の間に `\xkanjiskip` が挿入されることから，「あ」と `hbox 1.`の間には `\xkanjiskip` が挿入される．
 - 「い」と `hbox 2.`の間には `\xkanjiskip` が挿入されない．
3. 同様に，`np` の後ろに `\[x]kanjiskip` を挿入するか否かは，文字 `last_char` の後ろに対してどうなるかの値を用いる．上の例では，
 - 「a」と「い」の間に `\xkanjiskip` が挿入されることから，`hbox 1.`と「い」の間には `\xkanjiskip` が挿入される．
 - `hbox 2.`と「う」の間には `\xkanjiskip` が挿入されない．

次のループに進むための設定も，node `last_char` における値をもとに行う．

以上より，項目 1. で与えられた入力では，次の出力が得られる：

あ a い b う c え d お

要検討：空の `hbox` は跨がないようにしているのは，次の 2 つを使い分けられるようにするため：

- JFM グルー挿入の抑止：`\inhibitglue`

- 全ての和文処理グルー挿入の抑止：`\inhibitglue\hbox{}\inhibitglue`

なお， np の垂直変位が 0 でない場合は， np の前後への `\[x]kanjiskip` の挿入は行われない（即ち， $insert_skip \leftarrow no_skip$ となる）。

np が kern の場合

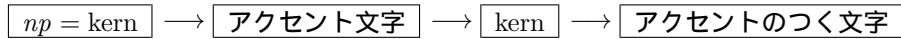
前に書いたように， $subtype$ が 0 の kern (TFM 由来) や， $subtype$ が 1 であっても $jtype$ が I, E, T の kern は 挿入処理を透過してしまうので，今問題にしているのはそうでない場合である。

- $subtype = 1$ の場合 .

挿入処理で透過されないのは $jtype$ がない (明示的 kern) か， $jtype$ が J (JFM 由来グルー) という 2 つの場合であるが，いずれの場合も， np の周囲には `\[x]kanjiskip` の挿入は行われない。そのため，次ループのために 行われる処理は， $insert_skip \leftarrow no_skip$ ， $np \leftarrow next(np)$ だけである。

- $subtype = 2$ (`\accent` 由来) の場合 .

np はリストの先頭から走査されていることから， np に続く node の並びは



となっている。pTeX-3.2 と同様に，LuaTeX-ja では `\[x]kanjiskip` 挿入処理で アクセント文字は無視することになっている。そのため， nq は変化せず，次のループで処理対象となる np は $np \leftarrow next(next(next(np)))$ となる。

np が数式境界 (`math_node`) の場合

数式境界は，便宜的に「文字コードが -1 である文字」とみなして内部で処理している。

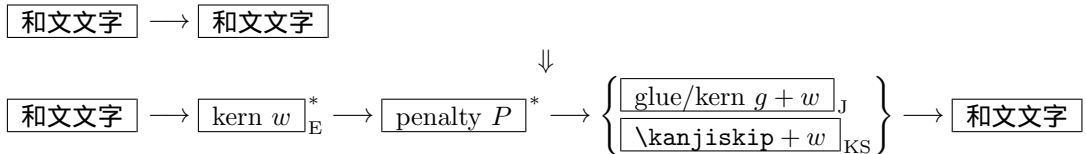
その他の場合

以上に出てきていない node (`vbox`, `rule`, `discretionary break`, `glue`, `margin_kern`) の 周囲には `\[x]kanjiskip` の挿入は行われない。

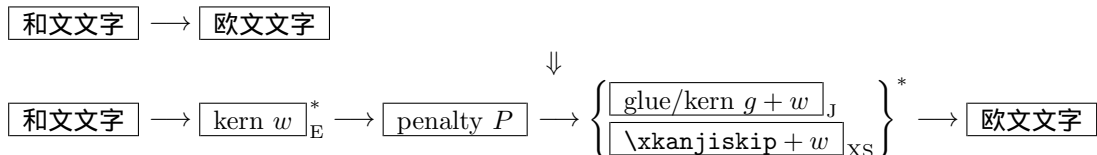
JFM 由来グルーとの関係の例

最後に，今まで説明した，JFM 由来グルーと `\[x]kanjiskip` の処理によって，実際にどのように node の 並びが変わるかをいくつかの例で示す。上付きで * がついている node は，値が 0 だと挿入されないことを示す。

- 例 1: 2 つの連続する和文文字の間



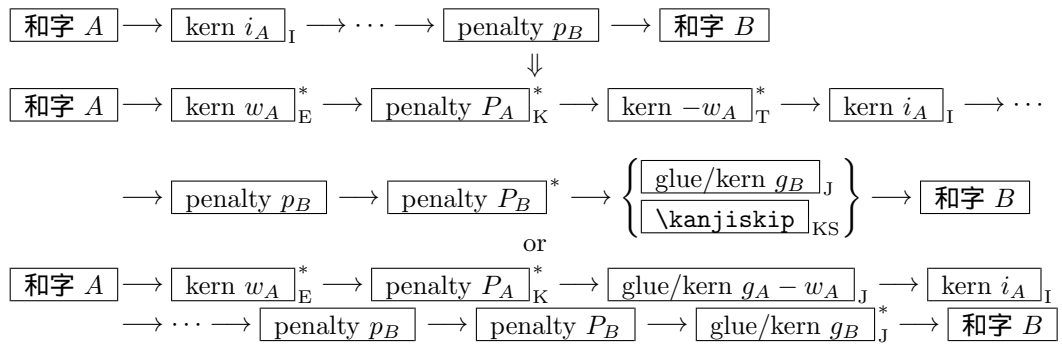
- 例 2: 和文文字と欧文文字の間



ここで，上の 2 つの例に出てきた記号の意味は次の通り：

- w : 前側の和文文字と行末の間に入る kern 量 .
- g : 2 つの文字の間に入る glue/kern (JFM 由来) .
- P : 2 つの文字の禁則用 penalty の合計値 .

● 例 3: 2つの和文文字の間にいくつかの「無視される」node 達



ここで,

- w_A : 和文文字 A と行末の間に入る kern 量 .
- P_A : 和文文字 A の行末禁則用 penalty .
- g_A : 和文文字 A と 'jcharbdd' との間に入る glue/kern .
- P_B : 和文文字 B の行頭禁則用 penalty .
- g_B : 'jcharbdd' と和文文字 B の間に入る glue/kern .