With comments from the Integration Panel
Overall, the IP finds the direction of the discussions among the CJK-GPs very encouraging. In some instances we have comments on some particular suggestions in this documents. As a response, we have added annotations like this in the pages that follow, or simply placed a green "check mark" next to items with which we have no issues.

Coordination Steps between CJK Generation Panels (Draft Proposal)

26th March 2015

Yoshiro Yoneya, JGP

Acknowledgement : Wang Wei, Ma Di, Yan Zhiwei

Precondition

- Each of CJK Generation Panels (GPs generates LGR for each language TL before integration ✓
  - – CJK GPs pick up ideographic variants for CJK from domain name usage perspective ✓
  - – CJK GPs don’t elaborate ideographic variants for CJK from linguistic perspective ✓
- CJK GPs agree on the mechanism (steps) to integrate and extract each language LGR

we understand "domain name usage perspective" to imply the kind of variant code points that are appropriate for domain names, excluding variants of academic or historic interest only; if that is the intent, we fully support it.

Step 1

While the procedure defines LGRs based on BS-47 language ids, the language field is always set to "und", leaving the ISO-15924 script IDs as the differentiating element of the tag. The tags "und-Jpan" and "und-Kore" are effectively used with only one language each, so in practice, the distinction matters little.

- Each CJK GP generates its own LGR (hereinafter, LGR-1)
  - This step corresponds to Dr. Zhang’s definition 1 as copied below (as of 2015-03-10)

1. Attempt to Redefine Variants Within One Language Context ✓

VARIANT GROUP DEFINITION

Within one language context (~~CHS, CHT, KR, or JP~~Chinese, Japanese, or Korean), the ideographs with different glyphs are defined as Variant Groups whose major /modern pronunciations, meanings and usages are the same. ✓

Step 2 (1/2)

- CJK GPs collectively generate a merged table of each LGR-1 (hereinafter, LGR-M)
  - Repertoire (codepoints; CPs) of LGR-M is the union of all CJK LGR-1s ✓
  - Variants of each CP in LGR-M is the union of variants (for the CP) defined in all CJK LGR-1s ✓
  - LGR-M does not have
    - Language tag ✓
    - Disposition of variants ✓
    - WLE (Whole Label Evaluation rules)

WLE rules are not script-specific in the procedure: there is only a single, common set. The IP will have to ensure that WLEs contributed by the various LGRs do not conflict with each other. Unlike the repertoire, for example, WLEs can be expressed in the XML in a number of different, but equivalent ways. This makes a merge of WLEs less of a mechanical exercise. For these reason, it is probably best for GPs and IP to be in an active dialog on the subject of WLEs and how to integrate them. When the IP creates the integrated LGR, there will be a collective LGR file **that includes a set of merged WLEs.**

Step 2 (2/2)

– This step corresponds to Dr. Zhang’s definition 2 as copied below (as of 2015-03-10)

2. Attempt to Define Variants across Language Contexts

CJK VARIANT GROUP DEFINITION

In the CJK language environments, if in each Variant Group (~~CHS, CHT, KR, and JP~~Chinese, Japanese, or Korean) there exists at least one identical ideograph, those Variants Groups shall be integrated as a CJK Variant Groups. ✓

symmetry

According to the ~~transivity~~ symmetr~~ic~~ity and transitivity principle, the members of Variant Groups in different language contexts can be treated as CJK Variants of each other. ✓

~~In a CJK Variant Group, there will be one, two or three TYPICAL, ORTHOGRAPHIC, or PREFERRED VARIANTs coming from different languages.~~

Note: The last sentence is meaningless according to precondition

Step 3

Goal is to reduce # of allocatable variant **labels**. One way to do that is to reduce the number of allocatable variant code points.
However, having the identical **label** get same variants no matter which script ID it is applied under does not increase the number of labels, except where it would lead to allowing "simplified" variants for a J label that has trad Han with Hiragana or Katakana mixed in. The IP thinking is that such cases must be avoided, and that makes it not possible to simply make all C variants "allocatable" in the J LGR

- From LGR-M, each GP picks up the variants corresponding to every CP in its LGR-1
  - Disposition of variant(s) inherits its original disposition value in LGR-1 ✓
  - Disposition of variant(s) not defined in LGR-1 is defined as:
    - “blocked” if the variant is not in LGR-1 repertoire
    - “allocatable” otherwise (CJK consensus is needed on this)

Note: in the XML schema this is done via the "type" attribute, on the <var> element and only labels have dispositions

"blocked" is the default and has the benefit of not creating additional allocatable variant labels.

Making this "optional" on a case by case basis would be better. So that each GP should be free to pick whatever "type" works best.

Another option is : Otherwise, inherit its original disposition value in LGR-1 (one of “allocatable”, “simp”, “trad”, or “both”)

Step 4

- Each GP extracts additional CPs and corresponding variants from LGR-M to establish symmetr~~icit~~y
  - Additional CPs are the ones that are not CPs in LGR-1 but appear as VPs in LGR-M ✓
  - Disposition of each additional CP is "out-of-repertoire-var" ✓
  - Disposition of variant(s) of additional CPs are "blocked" if it is in LGR-1 repertoire, otherwise "out-of-repertoire-var" ✓

Step 5

- Each GP appends WLE defined in LGR-1 to the result of Step4

IP will need to work with GPs' to resolve WLEs so that WLE Rules do not conflict.

- And add following WLE rules
  - <action disp="invalid" any-variant="out-of-repertoire-var" /> ✓

If an LGR "inherits" type (disposition) values for variants from the original LGR-1 of another GP, then the corresponding <action> elements must also be present, so that these types evaluate to the correct disposition of variant labels. (This refers to the "option" from step 3)

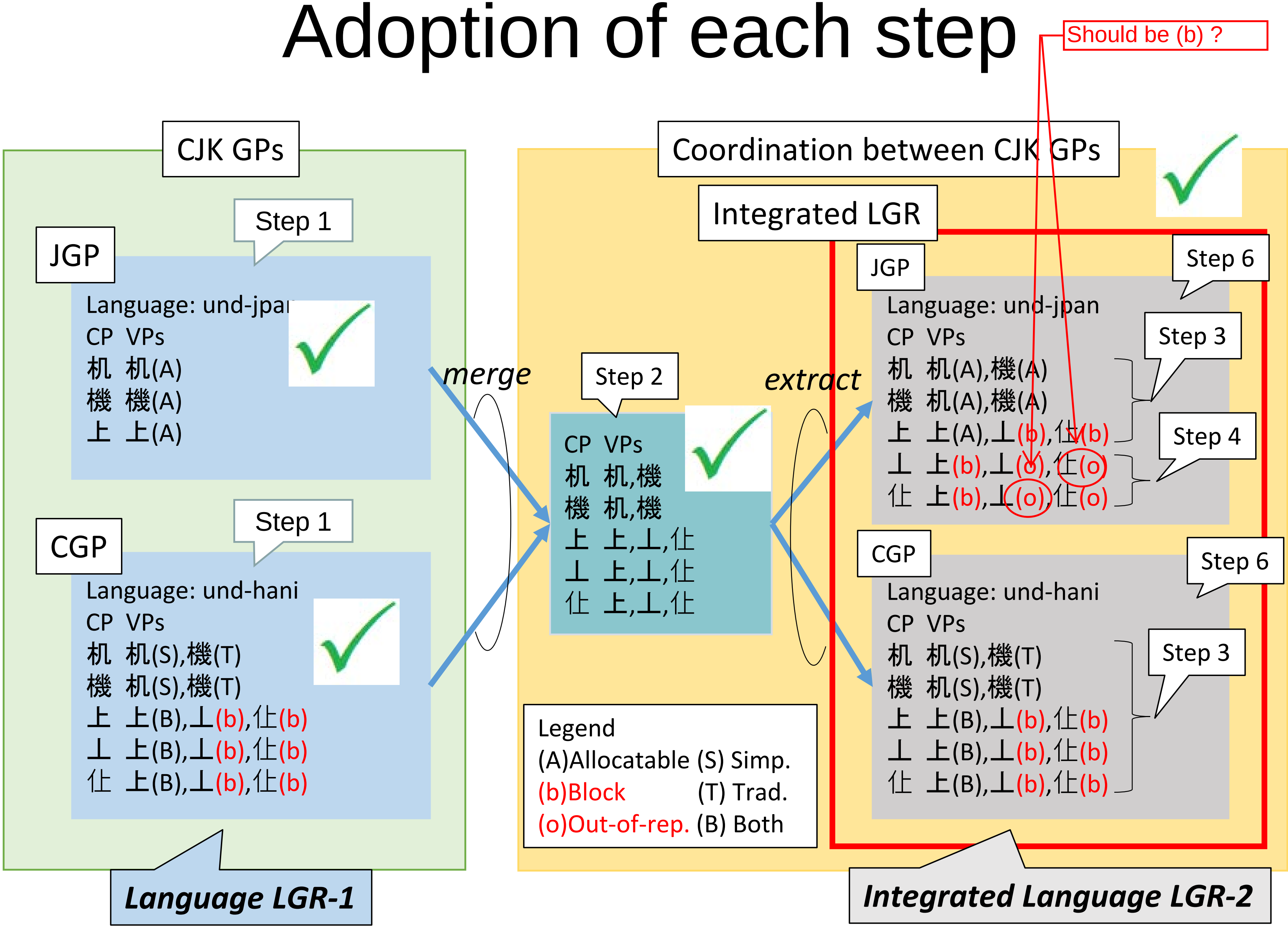
Step 6

- Each GP compiles extracted CPs and its corresponding variants (with dispositions) and WLE into integrated language LGR (hereinafter, LGR-2) ✓
  - And add language tag from its LGR-1 ✓

[Case Study]
Consideration on “機上”

- In case of JGP does not define any variants in Japanese LGR-1

Adoption of each step



Should be (b) ?

CJK GPs

JGP
Step 1
Language: und-jpan
CP VPs
机 机(A)
機 機(A)
上 上(A)

CGP
Step 1
Language: und-hani
CP VPs
机 机(S),機(T)
機 机(S),機(T)
上 上(B),丄(b),仩(b)
丄 上(B),丄(b),仩(b)
仩 上(B),丄(b),仩(b)

Language LGR-1

Coordination between CJK GPs

Step 2
CP VPs
机 机,機
機 机,機
上 上,丄,仩
丄 上,丄,仩
仩 上,丄,仩

Integrated LGR

JGP
Step 6
Step 3
Step 4
Language: und-jpan
CP VPs
机 机(A),機(A)
機 机(A),機(A)
上 上(A),丄(b),仩(b)
丄 上(b),丄(o),仩(o)
仩 上(b),丄(o),仩(o)

CGP
Step 6
Step 3
Language: und-hani
CP VPs
机 机(S),機(T)
機 机(S),機(T)
上 上(B),丄(b),仩(b)
丄 上(B),丄(b),仩(b)
仩 上(B),丄(b),仩(b)

Legend
(A)Allocatable (S) Simp.
(b)Block (T) Trad.
(o)Out-of-rep. (B) Both

Integrated Language LGR-2

Label generation examples

<In case of Japanese>

Language: und-jpan
Applied: 機上
Allocatable: 机上,機上
blocked: 机丄,机仩,機丄,機仩

Language: und-jpan
Applied: 机上
Allocatable: 机上,機上
blocked: 机丄,机仩,機丄,機仩

Language: und-jpan
Applied: 机丄
(no label generated due to out-of-repertoire-var)

Language: und-jpan
Applied: 機机
Allocatable: 机机,机機,機机,機機
blocked: (nothing)

These are **too many**, what if label was "**機机機机**" ? Would this result in 16 variant labels?

<In case of Chinese>

Language: und-hani
Applied: 機上
Allocatable: 机上,機上
blocked: 机丄,机仩,機丄,機仩

Good! Limit to two allocatable labels. Even if the label was longer, still only two allocatable variants.

Language: und-hani
Applied: 机上
Allocatable: 机上,機上
blocked: 机丄,机仩,機丄,機仩

Language: und-hani
Applied: 机丄
Allocatable: 机上,機上
blocked: 机丄,机仩,機丄,機仩

Language: und-hani
Applied: 機机
Allocatable: 机机,機機
blocked: 机機,機机 (S/T mixed)

Overall, the goal of the IP is to help the CJK-GPs to arrive at coordinated LGRs, such that the number of allocatable variant **labels** is minimized.

Normally this would be achieved simply by setting all "inherited" variants to "blocked". However, the IP understands that in some cases the results of that may not be not optimal from the perspective of the individual language.

In that case, the IP may accept some other variant types, however, mechanically setting all inherited variants to "allocatable" is unlikely to be acceptable. Without some WLE rules, the number of permuted variant labels would quickly grow as the label gets longer. Because of the risk that variant labels present to the DNS, the IP generally supports WLEs that limit the number of allocatable ***labels*** to 2-3 out of each variant label set.

For the C-LGR, this is done via the use of special types "trad", "simp" and "both" and corresponding WLE rules (<action> elements). These rules limit the number of allocatable labels to approximately three, no matter how long the label. If another GP prefers to avoid "blocked" for inherited variant code points, there will be a problem of overgeneration of allocatable variant labels. To fix that, it may be necessary to either inherit these special types and WLE rules as well, or to derive something comparable.

(blank page)

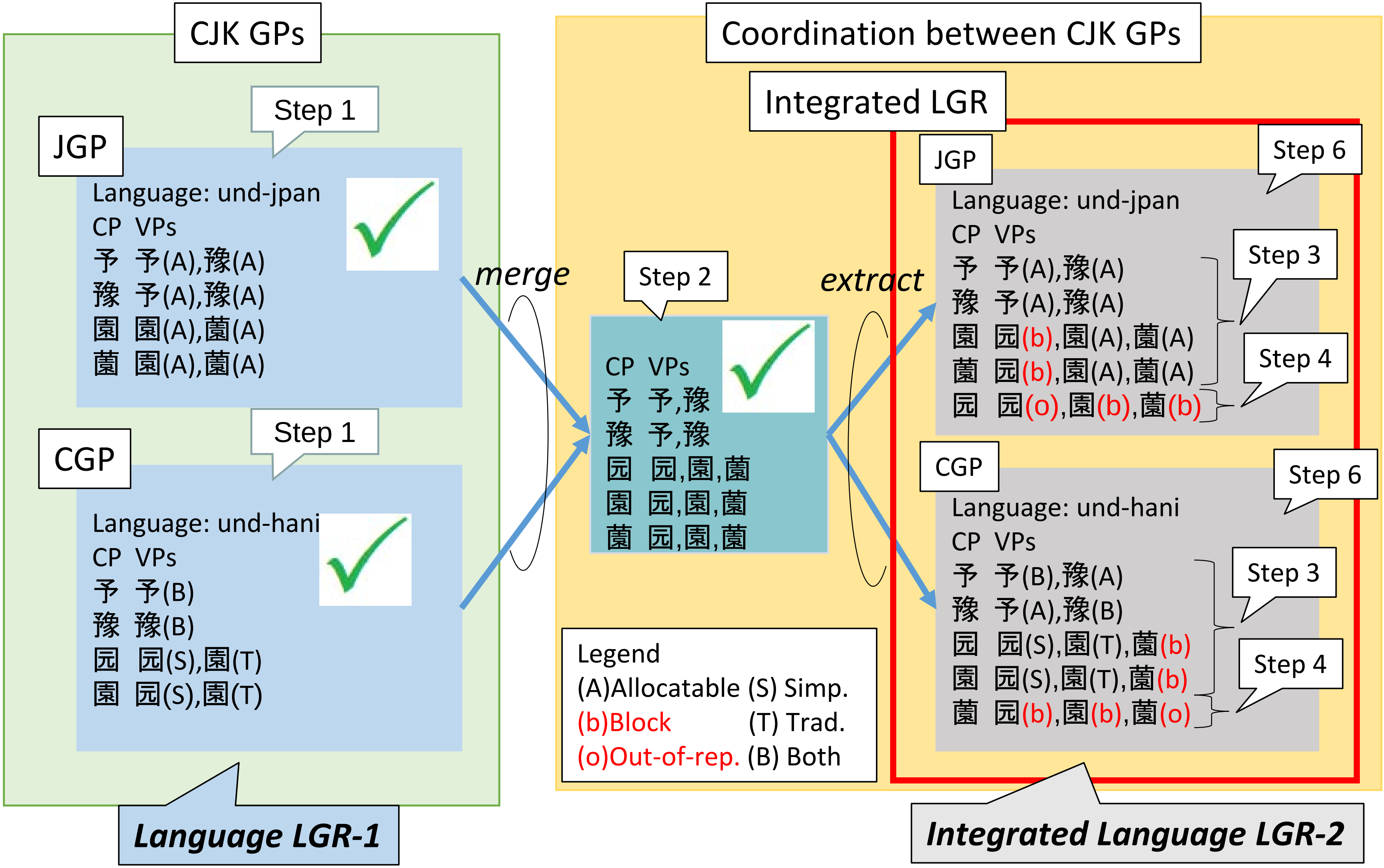
GPs that design LGRs where four-code point labels could lead to 16 allocatable variant labels (as the case studies appear to show) should expect some push-back from the IP.

Finally, the IP would like the JGP and KGP to address the issue of variants when used with mixed labels that include non-han code points.

[Case Study]
Consideration on “豫園”

- In case of JGP defines variants in Japanese LGR-1

Adoption of each step



CJK GPs

Step 1

JGP

Language: und-jpan ✓

CP	VPs
予	予(A),豫(A)
豫	予(A),豫(A)
園	園(A),薗(A)
薗	園(A),薗(A)

Step 1

CGP

Language: und-hani ✓

CP	VPs
予	予(B)
豫	豫(B)
园	园(S),園(T)
園	园(S),園(T)

Language LGR-1

merge

Coordination between CJK GPs

Step 2

CP	VPs
予	予,豫
豫	予,豫
园	园,園,薗
園	园,園,薗
薗	园,園,薗

✓

extract

Integrated LGR

JGP

Step 6

Language: und-jpan

CP	VPs
予	予(A),豫(A)
豫	予(A),豫(A)
園	园(b),園(A),薗(A)
薗	园(b),園(A),薗(A)
园	园(o),園(b),薗(b)

Step 3

Step 4

CGP

Step 6

Language: und-hani

CP	VPs
予	予(B),豫(A)
豫	予(A),豫(B)
园	园(S),園(T),薗(b)
園	园(S),園(T),薗(b)
薗	园(b),園(b),薗(o)

Step 3

Step 4

Legend

(A)Allocatable	(S) Simp.
(b)Block	(T) Trad.
(o)Out-of-rep.	(B) Both

Integrated Language LGR-2

Label generation examples

<In case of Japanese>

Language: und-jpan
Applied: 豫園
Allocatable: 予園,予薗,豫園,豫薗
blocked: 予园,豫园

Language: und-jpan
Applied: 豫园
(no label generated due to out-of-repertoire-var)

Language: und-jpan
Applied: 豫薗
Allocatable: 予園,予薗,豫園,豫薗
blocked: 予园,豫园

<In case of Chinese>

Language: und-hani
Applied:豫園
Allocatable: 予园,予園,豫园,豫園
blocked: 予薗,豫薗

Language: und-hani
Applied: 豫园
Allocatable:予园,予園,豫园,豫園
blocked: 予薗,豫薗

Language: und-hani
Applied: 豫薗
(no label generated due to out-of-repertoire-var)

These are **too many**, what if label was four code points long? Would this result in 16 variant labels?

What about the case where a label is applied for that combines Han and Kana characters in Japanese. Which cases do we expect that the Han-variant can be substituted for the applied for Han code point in such a mixed label? -- This issue is specific to JGP, but it is affected by the design of the Han variants -- unless JGP has a WLE rule to "block" variants for mixed labels altogether.