Subject: Preserving round-trip integrity of CJK compatibility ideographs
Date: 2012-2-16
Source: Masahiro Sekiguchi (An expert's individual contribution)
Re: WG 2 N4246 (Stabilizing CJK Compatibility Ideographs through the use of Standardized Variants)

Summary

The author partly understands the concerns expressed in N4246 but does not think the recommendation in N4246 is a good solution. Alternatives and some analysis are given for further discussion.

Introduction

In early days of UCS/Unicode, CJK COMPATIBILITY IDEOGRAPHS are introduced into the standard to assure so-called "round-trip integrity" upon transcoding between the UCS/Unicode and some other coded character sets. Many of those added later have similar purposes, too.

Round-trip integrity is an important feature for some particular application, but not so important for some others. Use of compatibility ideographs is not recommended in an application that has no such requirements, so the presence of compatibility ideographs in their data, or any convention/profile that explicitly allows/requires use of them, is a clear indication that the application requires the round-trip integrity. (Note that in this context, *application* means not just an application program but a collection of multiple programs possibly connected through network, collaboratively performing a specific objective.)

I don't know why designers of Unicode Normalization Specification decided to fold compatibility ideographs into unified ideographs upon normalization, but the straight forward consequence of the decision is that an application that requires round-trip integrity upon transcoding should not normalize Unicode data anywhere inside the application.

The author of N4246 just writes "*it is not possible to guarantee that normalization will not be applied, except for completely closed environments.*" It may be true, but we today have a lot of programs that, say, blindly consider the data is pure ASCII, Shift-JIS, or something else, breaking Unicode data. It is not possible to guarantee that bad handling of data based on wrong encoding assumptions will not be applied, except for completely closed environments. Both of them are sad facts. I'm not sure why the author of N4246 emphasizes the former.

I consider it is primarily a mistake of end users, system integrators, or application program designers doing normalization in an application where compatibility ideographs are required.

At the same time, I know that, in many contexts, especially those related to modern Internet, many aspects of the normalization benefits, e.g., recognizing a composite sequence and a precomposed character of a *same* accented letter, are getting more and more important. There are applications that round-trip integrity is essential, and many normalization aspects are also important.

I agree the use community of UCS/Unicode requires some solution.

My objection to the recommendation in N4246

N4246 says "1,002 Standardized Variants ... would be equivalent to the CJK Compatibility Ideographs". It is not clear what the word *equivalent* means in this context. I want clarification how those Standardized Variants are meant to be used.

If the word "equivalent" means "to be normalized", the recommendation can't be a solution to the normalization problem, so I think it means something else.

I have a feeling it means "should be used in place of". If I'm correct, it requires a large amount of updates to the existing applications that depends on compatibility ideographs today, and I don't think the transition is feasible. I don't understand why use think " a wide variety of products, protocols, and environments normalize text data on a regular basis, and this cannot be changed" while use think transition from use of compatibility ideographs to the newly proposed sequences is easy.

Alternative 1

The first alternative proposed here is to define several other types of normalization forms that preserve compatibility ideographs as they are while normalizing composite sequences and precomposed characters. We need further study before deciding which parts of the mappings should be kept and which parts should be changed in details, but I believe it's doable.

The best part of this alternative is that most of the transcoding-dependent applications require very small changes; when the new normalization forms are supported on the platforms, application programs simply invoke the normalization services to use the newly-defined normalization. It is a big difference from the N4246 recommendation; it requires rewriting of all existing application that use compatibility ideographs to use the newly introduced *standardized variant sequences*.

Alternative 2

Recommend everyone to keep all Unicode data in their original no-normalized forms, and perform normalization only before comparison or similar operation, discarding the normalized data soon.

I understand this changes today's practice. I also remember the discussion took place in a W3C working group that published a "normalize everything before send it to the Net" recommendation. However, there are good chances this change is accepted by the community, because security environment surrounding the Internet application changed in the past decade, and application that communicate through Internet anyway need to normalize all received data before any normalization-critical operations since it can't assume the received data from someone else is properly normalized.

The transition need not be immediate nor synchronized. Yes, it will require some long time to this transition to complete all over the world, but anybody who concerns problem can start early without waiting others. This seems a big plus of this alternative.

What is the problem, by the way?

The document N4246 is busy to explain what happens when normalization is applied to compatibility ideographs and *solution*, it is silent on what is the problem. As an engineer working for an IT vendor, I receive many tough issues regarding Unicode or other codeset issues though the company's sales and support division in regular basis, but I have never heard of complains something like "My compatibility ideograph was lost after normalization!" I doubt you are only discussing possibilities of problems but real problems.

On the other hand, I sometimes receive query on composite sequence and precomposed character comparison issues recently, especially those regarding voiced kana letters. I sometimes recommend normalize-before-compare strategy, but, of course, I can't always because it breaks round-trip integrity if the application requires one. My problem is hence: how I can process compatibility ideographs maintaining round-trip integrity, while recognizing a corresponding composite sequence and a precomposed character identical. I have no easy and universal solution to the problem.

I love to work with those who share the same concern for a good solution (not limited to two proposals stated in this document,) although it is not appropriate to say it here, since it seems a separate goal from N4246.