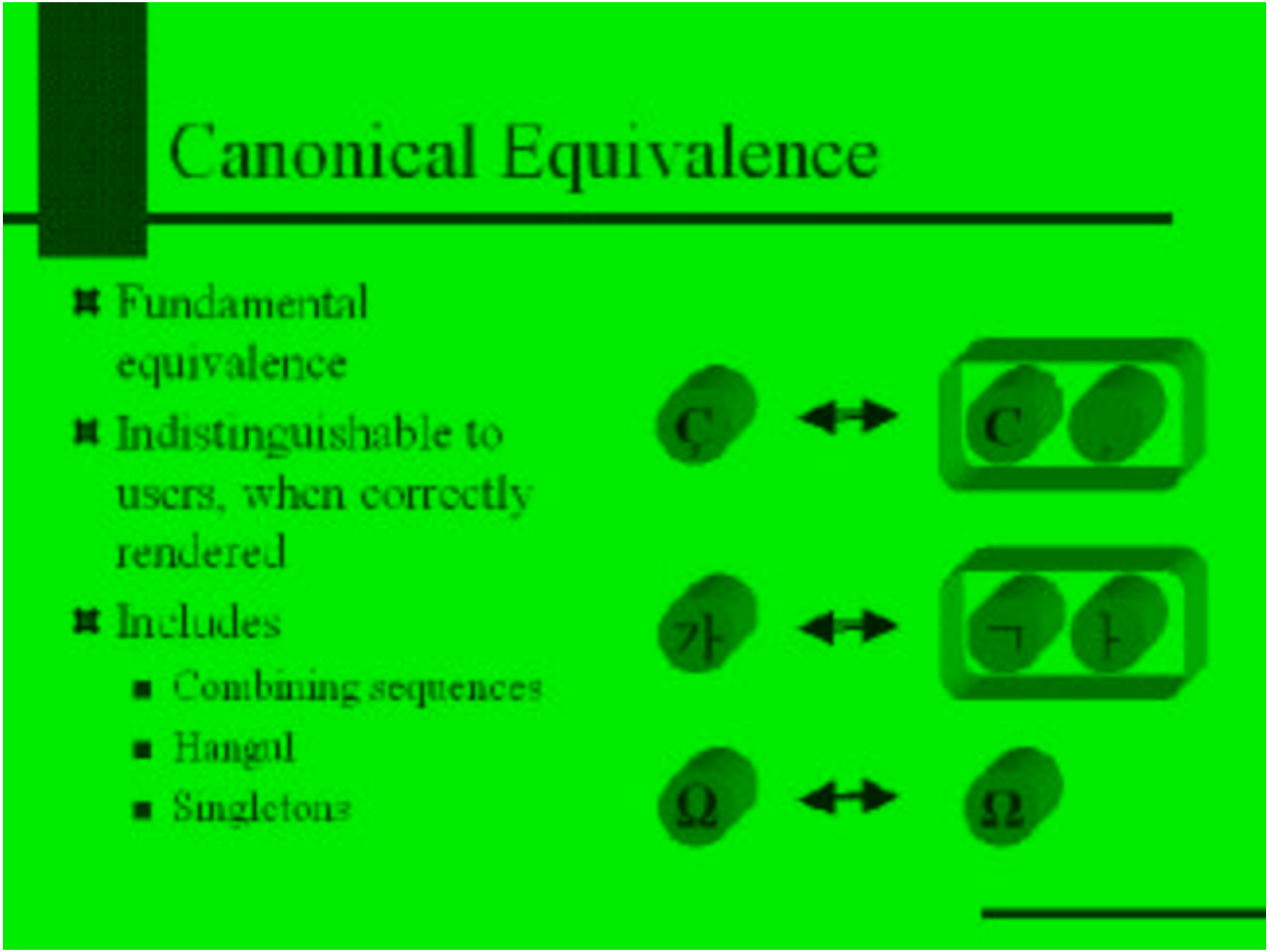
Unicode support in OpenLDAP 2.1

Stig Venaas
UNINETT
Stig@OpenLDAP.org

Intro

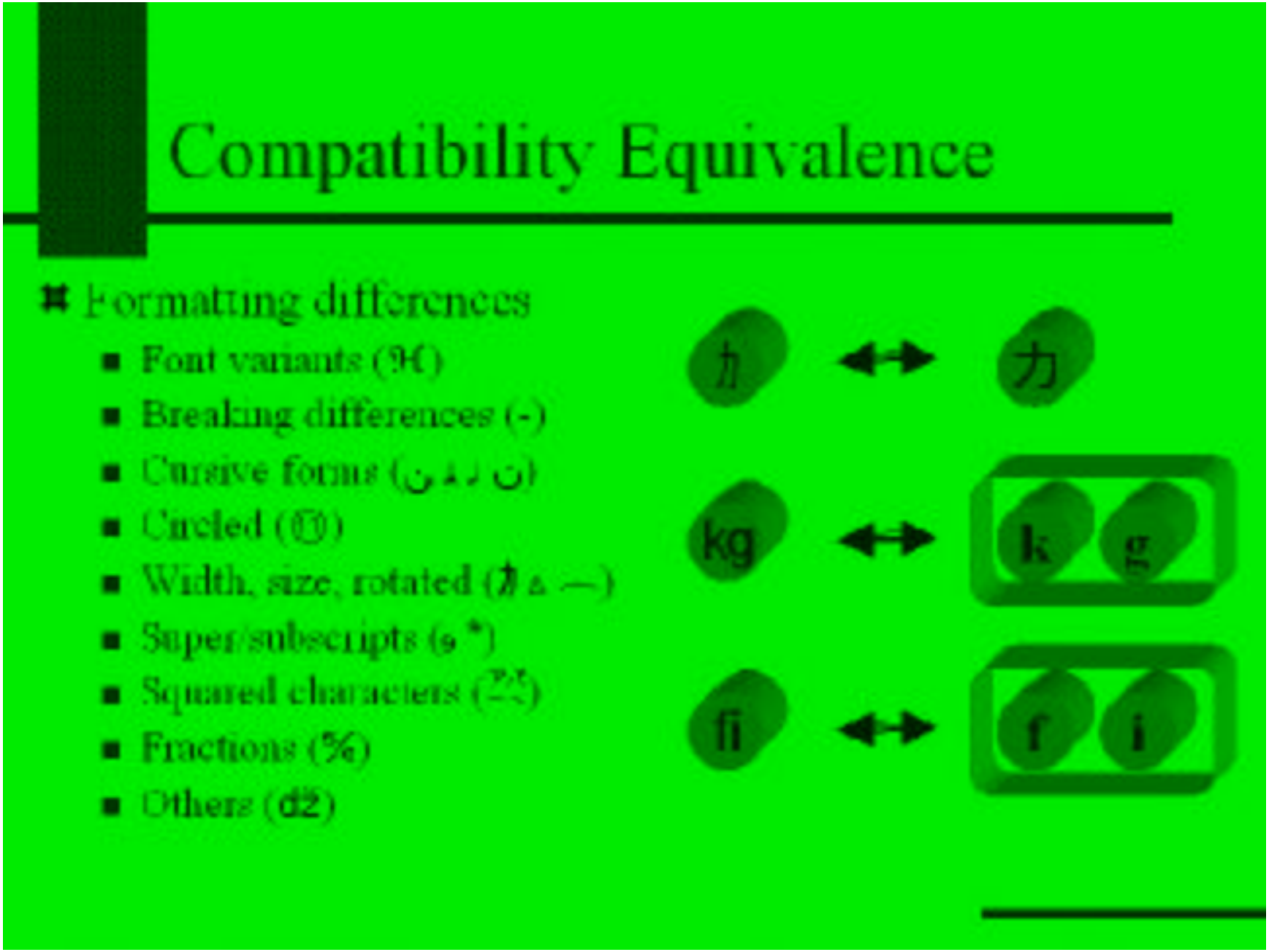
- Prior to 2.1 one could easily store Unicode strings as UTF-8
- Fine, except for matching
- 2.1 does Unicode NKDC normalization and case folding
- Will look at what it is, why, how it works, and issues

Unicode canonical equivalence



From Unicode Standard Annex #15

Unicode compatibility equivalence



From Unicode Standard Annex #15

Normalization

- In OpenLDAP we want to ignore compatibility differences
  - Two strings that are comp equiv should be equal
- We use the normalization form KC (NFKC)
  - Compatibility Decomposition + Canonical Composition
- After KC we can do binary comparison (memcmp())
- Uses UCData library to do the work

Equality match in OpenLDAP

- Attribute values stored "as is"
- When indexing, normalize before create hash values
- When searching, normalize assertions
- If indexed, compute hash and do look-up
- For each candidate (after index filter)
  - Normalize stored value and binary compare with assertion

CaseIgnoreMatch

- In this case we fold to lower case in addition to the normalization previously described
- We use Unicode folding tables and UCData library.

Substring matching

- Same principles, normalize and compare bytes
- There is one potential issue
- A character might consist of multiple code points
- We do substring on code points not character
  - E.g. last character in assertion might be part of a character in the value
  - something like: searching for bla* matches bla" (a-umlaut)
  - this example isn’t valid though since we compose first
- Not a problem, I think...

Issues(1)

- Biggest problem is speed
- Have tried to maintain speed for ASCII
- Data often normalized, should check whether data already normalized
- Might cache normalized strings
- When normalizing we should use stringprep [RFC 3454]
  - transcode, map, normalize, prohibit, check bidi, insignificant character removal
- See draft-zeilenga-ldapbis-strmatch-02

Issues(2)

- Unicode and regexp
  - perfectly possible, but performance...
  - do we really need regexp acls? component matching?
- Unicode and sorting, a bit tricky
  - Want to avoid locale... how to know clients locale preferences
  - perhaps lang tags
    - what if sorting on attributes with different lang tags