Discussion questions:

1. How do we get this to release? Do we, for example, just open up a public comment period? If yes, where, and how? Should we call it 1.1, or 2.0?
2. How, more generally, should we manage change in CSL going forward? What other barriers (say in terms of resources, including time/labor) impact our answer to this question, and how might we reduce those barriers?
    a. Styles repository
    b. Processors
    c. Schema
    d. Clients

Discussion

1. A need for reliable channel(s) for communication between schema maintainers, processor + client developers
    a. How do we know when the amount of the feedback is sufficient?
    b. How can we flag changes, request the type of feedback we need, ensure the right people give input?
2. A change in the versioning system
    a. Can minor changes like a new term be added without version incrementing?
        i. Cormac liked the idea of splitting
    b. Can we have smaller feature releases?
        i. Andras noted that one-by-one additions of larger features would be easier to implement for processors
        ii. But Sebastian noted this would be chaotic for style updates and the 35ish implementations of CSL
    c. What is our threshold and mechanism to ensure downstream implementation buy-in for a major release?
        i. We are concerned about further removed implementations being unable to adopt new features
        ii. Do we freeze features? Provide a very deliberate release process?
3. Processor development
    a. How do we support maintenance of the reference implementation?
        i. Cormac: The development of the test suite from citeproc-js provides a lot of support for development of other implementations independent of the citeproc-js codebase itself
        ii. Cormac: Does citeproc-rs provide an easier codebase for others to contribute to than citeproc-js, so that the burden on one individual is less?

iii. Sebastian: Can we make the process of writing and contributing new tests easier?
    1. Denis: Frank has a test-creation tool. Frank: it's cumbersome to set up [could that be made easier?], but smooth to use.
        a. https://github.com/juris-m/citeproc-test-runner
    2. Cormac: New test suite language using YAML. To write a test, take a copy of an old one
        a. YAML format: https://github.com/zotero/citeproc-rs/blob/master/crates/citeproc/tests/data/humans/collapse_YearSuffix_Locator_Ranged.yml
        b. Tool: https://github.com/cormacrelf/jest-csl

4. Styles repo maintenance
    a. Maintenance of citeproc-ruby to support CI is hard.
        i. Can we use a different processor into the CI chain? Or can we get support for maintaining citeproc-ruby?
    b. How can we get more people-power directed toward the styles, reviewing contributions, providing feedback?
    c. Can we build a test suite for the 10 major styles at least?
        i. Need a library of reference items
        ii. Need a method for correcting incorrect test output
        iii. Also tests for common errors like suffixes vs groups?
            1. Test-runner and CI checking could built into Zotero 7's CSL editor
        iv. Can we get a feature to click output and go to the part of the style that produced it?
    d. Can the visual editor be updated?
        i. Can the style search be made to work with arbitrary metadata?

Next steps: actions items

1. Open questions
    a. Can citeproc-rs be dropped into Zotero?
    b. Will community engage with citeproc-rs to make quality community-contributed code? How can we encourage engagement?
        i. Release to crates.io
        ii. Cormac: Most fixes probably come in the context of minor adjustments. Will this be doable to amateur programmers?
    c. CSL 2.0 (we should call it 2.0)

    i. What is the value proposition over 1.0.2?
        1. Can we adopt narrative citations without a full CSL 2.0 release
    ii. Modularity?
d. CSL 1.0.2 maintenance projects
    i. Test suite, editor, etc.
    ii. Roadmap to build these enhancements