Revisions to the paper "Staged Compilation With Two-Level Type Theory"

- I added an appendix as a supplement, which includes a complete listing of 2LTT rules and equations in a more traditional derivation notation. The appendix also includes a more compact specification of the same thing using a second-order algebraic signature.
- I extended the beginning of Section 3 with a motivation of algebraic models and quotients, and a description of extracting algorithms via a setoid interpretation.
- I added a bit more explanation to "Models and Syntax of 2LTT" in Section 3.3, trying to explain syntactic aspects.
- I added Section 3.3.2 as a short discussion of elaborating surface syntax to formal syntax, as a justification for the usage of surface syntax.
- I added Section 3.3.3 for a sketch of interpreting the syntax of 2LTT in setoids.
- I rephrased the correctness of staging, now I talk about soundness, stability and "strictness" in Definition 4.2. Strictness means that staging strictly preserves all object-level type and term formers, where "strictly" refers to equality in underlying non-quotiented types and terms. This captures that staging should not compute object-level redexes. As far as I see, it is easy to show strictness of staging so I added it as Theorem 4.16.
- I reworded Section 4.4 to be hopefully clearer and closer to what I actually did in the demo implementation.