In Lemma 1.1 (line 1):

- Line 5: l_j must exist in $\{\overline{l_i}\}$
- Must account for the case when t is typed by rule T-Sub
- Must handle application (t1 t2 : T)

In Lemma 1.3 (line 25):

- Evaluation is not deterministic given the rules E-Raise and E-RaiseRaise. Consider the following example:
    - Let t → t'
    - raise[T](raise[T] t) | μ → raise[T] t | μ by E-RaiseRaise
    - raise[T] t | μ → raise[T] t' | μ' by E-Raise
    - raise[T](raise[T] t) | μ → raise[T](raise[T] t') | μ' by E-Raise
- This can be fixed by requiring t in E-RaiseRase to be a value. The proof cannot be fixed without changing the definition
- Similarly, E-AppAbs is problematic. To prove the lemma, the definition must be changed to restrict t_2 to a value.

In Lemma 1.5 (line 41):

- The lemma can be strengthened by including the case that v : Unit. A type rule for unit must then be supplied, and the proof of soundness modified to account for the new rule.

In Lemma 1.7 (line 71):

- Line 87: Must use weakening to derive Γ,y : U |Σ |- s : S, and use that as the input for induction
- Line 107: Similarly, must use weakening.

In Lemma 1.9 (line 113):

- The lemma is too strong; the lemma should work on terms in the empty context. Otherwise, the case T-Var can occur.
- Line 128: We cannot get $S_1 = T_2$ from Lemma 1.1. We can instead strengthen the canonical forms lemma to tell us that $T_2 \leq S_1$ (the necessary information is already available in both cases of the lemma). Lemma 1.1 can tell us that $\Gamma|\Sigma|\text{-}\ fun\ x{:}S_1.\ t_{12} : S_1 \rightarrow T$ and $\Gamma,\ x{:}S_1\ |\ \Sigma\ |\text{-}\ t_{12} : T$. We can then use T-Sub to get $\Gamma|\Sigma|\text{-}\ t_2 : S_1$, which is passed on to Lemma 1.7 for QED.
- Line 144: The term *raise[T]v* cannot be a member of a record of type T. By T-Raise, *raise[T]v* is of type T. That means that the definition of T is recursive, yet we do not have recursive types in our definition. Therefore, the definition is flawed. The proof itself can be made to work by showing that such a term could not possibly be well typed, therefore proving the case by contradiction. The same applies to the T-Proj case below.
- Line 171: Must show that the new μ is well-typed using Lemma 1.8.
- Line 183: Lemma 1.1 does not show that Σ(p) = T, but that there is some R such that Σ(p) = R and R ≤ T. And can then be shown that μ(p) : R, and by T-Sub μ(p) : T. A similar case occurs on lines 201-202.

- Line 205: As mentioned above (under Lemma 1.5) there is no type rule for unit. Without being able to type unit, the lemma is unprovable.
- Line 229: In order to use E-HandleRaise, x must be of type T. Unfortunately, with the current definition of T-Try there's no reason it has to be. This case can not be proven without changing the definition of T-Try to use "x:T" instead of "x:T_1". Also, to prove type preservation, t_2 must have type T (given x:T). To prove this also requires modification of the typing rule.