Damian Franco

dfranco24@unm.edu

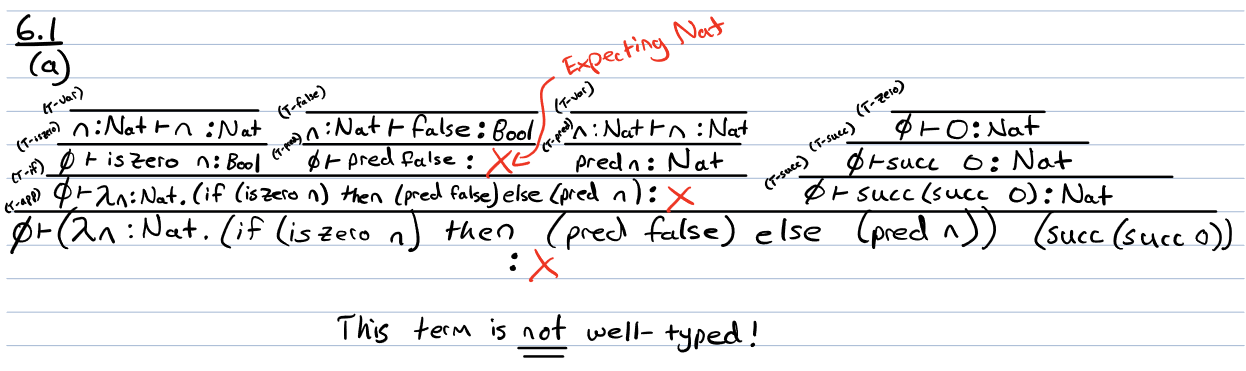
101789677

CS-558 001

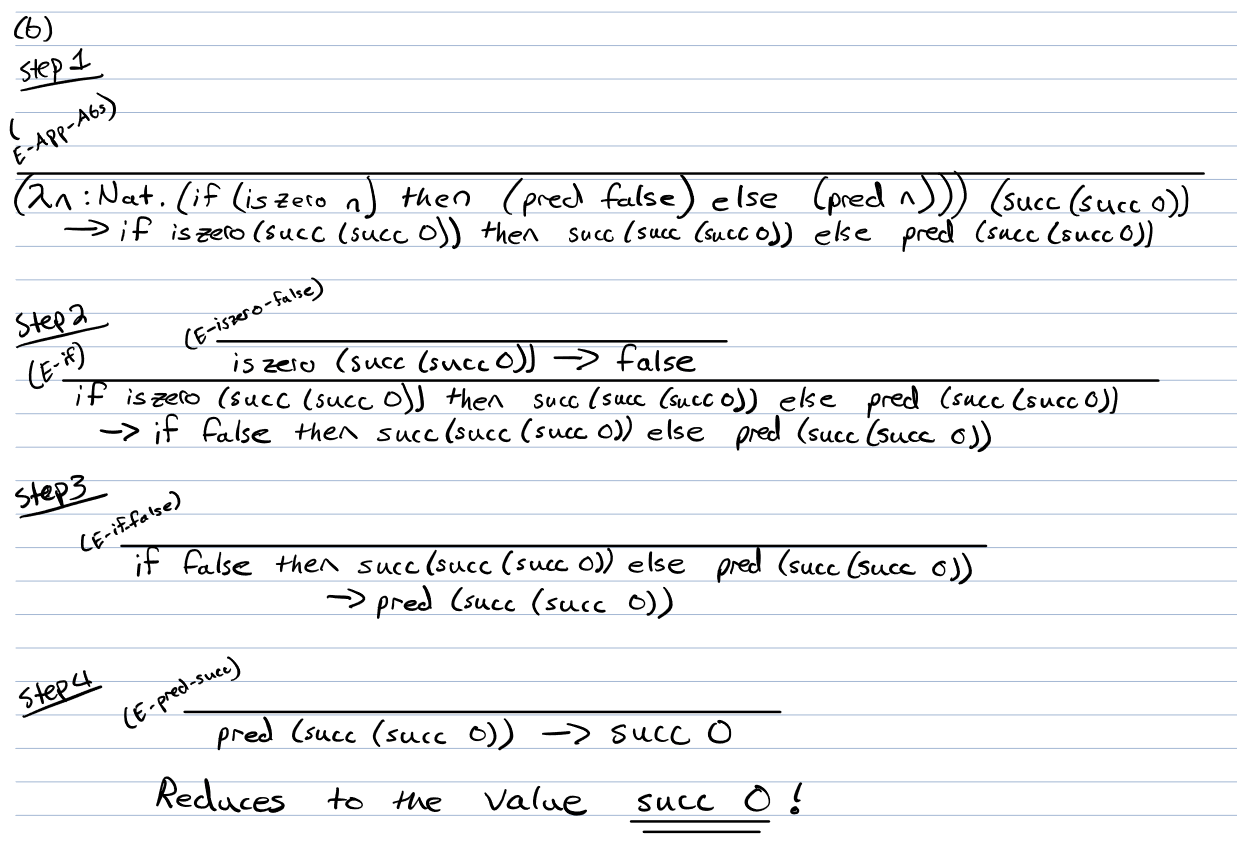
Homework 6

6.1 Typing and evaluation in the simply-typed lambda calculus (1)

1. For the first question, we were given the term listed below to check for proper typing. Below you will see the typing rules I used to derive an answer to deciding whether this term is well-typed or not. I found that this term is *not* well-typed due to the areas marked in red. The rule *T-pred* is expecting a type *Nat*, but instead gets the type *Bool* due to the *T-false* rule being used above. This causes the typing issues and is why I indicated that this term is not well-typed.

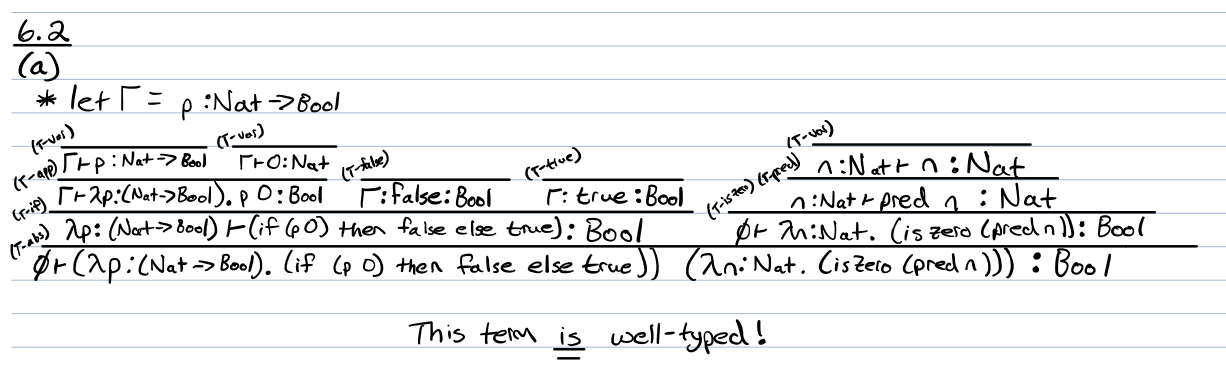


1. For the next part, we took the same term and derived a full reduction for the term. Below you will see my steps I took to derive the value *succ 0*. This reduction took four steps to complete and was able to fully derive to a value without getting stuck.

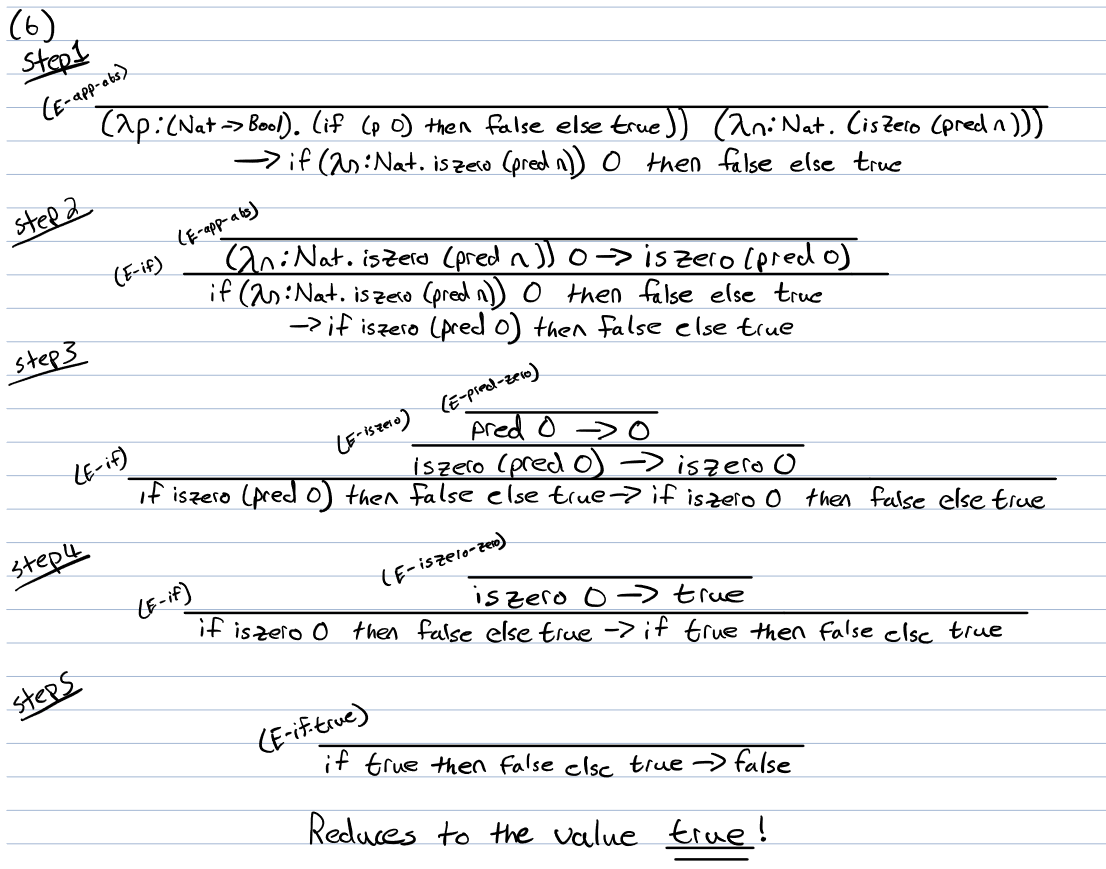


6.2 Typing and evaluation in the simply-typed lambda calculus (2)

1. For the first question, we were asked once again to type check the given term. Below you can see my full type checking using the typing rules given to us in the annex. I found that this term *is* well-typed. Everything seems to work out and all types all correlate with the correct rules. I also used the gamma to identify a longer term for the substitution just like Professor Lakin showed during lecture.

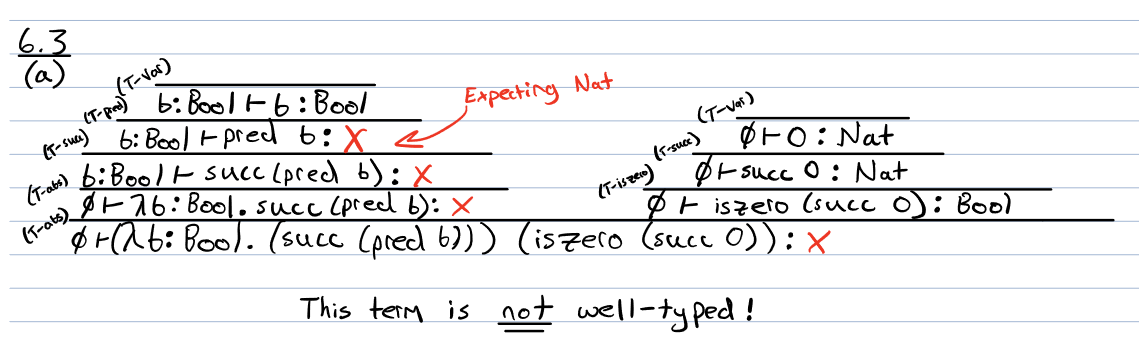


1. For the next part, we were asked to derive a reduction from the given term using the evaluation rules given to us in the annex. The work below shows my full reduction and the value that it reduced to which happens to be *true*. This evaluation took five steps to complete with no evaluation error occurring to avoid the reduction from getting stuck.



6.3 Typing and evaluation in the simply-typed lambda calculus (3)

1. For the final question, we were asked to first type check the term using the typing rules given to us. The handwork below shows that I found this term to *not* be well-typed. This is due to the spots where the red markings occur. The rule *T-Var* on the left side inner term has the type *Bool* from the lambda statement. The *T-pred* typing rule is expecting the type *Nat* from above which is why this error occurs and signifies that the term in not well-typed.



1. For the final part, we were again asked to use the evaluation rules to construct a proper reduction of the term given. This term lasted three steps until it was *stuck*. Below in the red you can see exactly where and why the term cannot be reduced further. The term *pred false* has no axiom that fits that term to reduce the term down further which causes the term to get stuck at the term *succ (pred false)*. 