**Task 1:**

1. Answer of given expression will be (if 1 then true else if true then 3 else 4). In untyped we mentioned that there will be stuck terms. And in some conditions, if stuck term will be applied, then, there are no other computations will happen. In out example we see that we have 1:NAT which is used in if clause. Clear that there are we need some True or False: BOOL. Also, I can mention that all code was checked, because we see that expression after else clause also was computed. But while first term will not be computed further ones will be stuck until computing of first ones.
2. **if succ(0) then true else (if true then succ(2) else succ(3)).** In typed language, there are can not be any stuck terms. Consequently, any expression or derivation will have further result. In our example, we use T-IF, which means that if t1 will be :BOOL typed, therefore, t2 also must be :BOOL. From this we move further to IF-TRUE point. There we say that t1 must be TRUE. Because we know that if t leads to t’, then t’ will give us t backwards. From there we can say that in our example, we meet counterexample which is incorrect.
3. We use safety. We know that in well-typed, expression never go to the Wrong state by two rules(Safety = Preservation + Progress), because they are never stuck and week-typed after processing also will be well-typed. In our situation, it identifies that t1 is: BOOL. Well-typed t1 allows us to further evaluations. But if t1 will be well-typed then t2 also must be well-typed. But we mention that t2 is: NAT. Consequently, it will stop calculation at this.
4. There are used NoRuleApplies in eval function. We can see that there are firstly done computation of the gives expression and then this computation given to main function eval. And there are used try catch to handle errors which will occur. Therefore, if there are not any further evaluation can be applied, it will throw NoRuleApplies, which say that this term is stuck or in normal form.
5. No, we cannot remove this rule. Clear that predecessor of 0 will return stuck term. By safety rules we know that by Progress rule, term cannot be stuck. Moreover, by safety rule, well-typed terms do not go wrong. Therefore, predecessor of 0 will broke safety rule.
6. In previous lecture there was counterexample: if true then 1 else false. This example shows that t1 was another type in comparing with t1’.

**Task 2:**

There are first of all we implement new type of data, which called TyWrong.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

We change everywhere in syntax.ml and syntax.mli with adding TyWrong which will print us “Wrong statement”. Then, in core.ml, in typeof function we replace the strings where IF statements are checked. Instead of strings, where we have error prints, we put TyWrong statements. It will give us Wrong output where we have NotBool Conditions.

Изображение выглядит как текст

Автоматически созданное описание

Also, I initialize TyNat, which will give us natural values data type. And when we test it, by giving Nat into the test, it gives us Wrong, because it is stuck. Nat type was added to next files:

* Parser.ml
* Parser.mli
* Syntax.ml
* Syntax.mli
* Lexer.mli
* Lexer.mll

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

All Nat typed data was added where TyBool was added.

Moreover, we add some TmWrong statements which is very useful there as we do it in HW2, with arith, where we also implement Wrong term.

Изображение выглядит как текст

Автоматически созданное описание

TmWrong was added into syntax.ml, syntax.mli and core.ml files. By the analogy of TmTrue or TmFalse which also will apply only info. And in core.ml file we add TmWrong terms, to explain the key points where Wrong statements may apply. And conequently, we must add TmWrong in typeof function, to show that we may have a type Wrong.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

At the end, I want to show test which I use to show that everything is works.

Изображение выглядит как текст

Автоматически созданное описание