add

```
Fixpoint add (m n : nat):
                                   call stack
                                              : []
match m with
                                   stack
                                               : []
\mid 0 \Rightarrow n
| S m' => add m' (S n)
                                   redex stack : []
end.
                                   guard env : []
                                   loc env
                                               : []
fun (n : nat) =>
                                              : [tLambda]
                                   call stack
match m with
                                   stack
                                               : []
\mid 0 \Rightarrow n
| S m' => add m' (S n)
                                   redex stack : [NoNeed]
end.
                                   guard env : [Large]
                                   loc env
                                               : [m, add]
match m with
                                   call stack
                                              : [tCase, tLambda]
\mid 0 \Rightarrow n
                                   stack
                                               : []
| S m' => add m' (S n)
end.
                                   redex stack : [NoNeed]
                                   guard env : [Bound{1}, Large]
                                   loc env
                                               : [n, m, add]
m (* discriminant *)
                                   call stack : [tRel, tCase, tLambda]
                                   stack
                                               : []
                                   redex stack : [NoNeed, NoNeed]
                                   guard env : [Bound{1}, Large]
                                   loc env
                                               : [n, m, add]
n (* 0-th branch *)
                                   call stack
                                              : [tRel, tCase, tLambda]
                                   stack
                                               : []
                                   redex stack : [NoNeed, NoNeed]
                                   guard env : [Bound{1}, Large]
                                   loc env
                                               : [n, m, add]
fun m': nat \Rightarrow add m' (S n)
                                   call stack : [tLambda, tCase, tLambda]
(* 1-st branch *)
                                   stack
                                   redex stack : [NoNeed, NoNeed]
                                   guard env : [Bound{1}, Large]
                                   loc env
                                               : [n, m, add]
add m' (S n)
                                   call stack : [tApp, tCase, tLambda]
                                   stack
                                               : []
                                   redex stack : [NoNeed, NoNeed]
                                   guard env : [Strict, Bound{1}, Large]
                                               : [m', n, m, add]
                                   loc env
```

| S n | call stack : [tApp, tApp, tCase, tLambda] stack : [] redex stack : [NoNeed, NoNeed] |
|---|---|
| | $\begin{array}{ll} \text{guard env} &: [\text{Strict}, \text{Bound}\{1\}, \text{Large}] \\ \text{loc env} &: [m', n, m, \text{add}] \end{array}$ |
| n | $ \begin{array}{lll} \text{call stack} & : [\text{tRel}, \text{tApp}, \text{tApp}, \text{tCase}, \text{tLambda}] \\ \text{stack} & : [] \\ \text{redex stack} : [\text{NoNeed}, \text{NoNeed}, \text{NoNeed}, \text{NoNeed}] \\ \text{guard env} & : [\text{Strict}, \text{Bound}\{1\}, \text{Large}] \\ \text{loc env} & : [m', n, m, \text{add}] \\ \end{array} $ |
| S | $ \begin{array}{ll} \text{call stack} & : [\text{tConstruct}, \text{tApp}, \text{tApp}, \text{tCase}, \text{tLambda}] \\ \text{stack} & : [\text{SClosure } n] \\ \text{redex stack} & : [\text{NoNeed}, \text{NoNeed}, \text{NoNeed}] \\ \text{guard env} & : [\text{Strict}, \text{Bound}\{1\}, \text{Large}] \\ \text{loc env} & : [m', n, m, \text{add}] \\ \end{array} $ |
| m' | $ \begin{array}{l} {\rm call\ stack} &: [{\rm tRel}, {\rm tApp}, {\rm tCase}, {\rm tLambda}] \\ {\rm stack} &: [] \\ {\rm redex\ stack} : [{\rm NoNeed}, {\rm NoNeed}, {\rm NoNeed}] \\ {\rm guard\ env} &: [{\rm Strict}, {\rm Bound}\{1\}, {\rm Large}] \\ {\rm loc\ env} &: [m', n, m, {\rm add}] \\ \end{array} $ |
| add | $ \begin{array}{lll} \text{call stack} & : [\text{tRel}, \text{tApp}, \text{tCase}, \text{tLambda}] \\ \text{stack} & : [\text{SClosure } m', \text{SClosure } (\text{S n})] \\ \text{redex stack} & : [\text{NoNeed}, \text{NoNeed}] \\ \text{guard env} & : [\text{Strict}, \text{Bound}\{1\}, \text{Large}] \\ \text{loc env} & : [m', n, m, \text{add}] \\ \end{array} $ |
| <pre>(* internal *) check_is_subterm (subterm_specif m') (wf_paths nat) == NeedReduceSubterm {}</pre> | $ \begin{array}{lll} \text{call stack} & : [\text{tRel}, \text{tApp}, \text{tCase}, \text{tLambda}] \\ \text{stack} & : [\text{SClosure } m', \text{SClosure } (\text{S n})] \\ \text{redex stack} & : [\text{NoNeed}, \text{NoNeed}] \\ \text{guard env} & : [\text{Strict}, \text{Bound}\{1\}, \text{Large}] \\ \text{loc env} & : [m', n, m, \text{add}] \\ \end{array} $ |
| <pre>(* internal *) reduce_if (needreduce discriminant needreduce branches)</pre> | $ \begin{array}{ll} \text{call stack} & : [\text{tRel}, \text{tApp}, \text{tCase}, \text{tLambda}] \\ \text{stack} & : [] \\ \text{redex stack} : [\text{NoNeed}] \\ \text{guard env} & : [\text{Bound}\{1\}, \text{Large}] \\ \text{loc env} & : [n, m, \text{add}] \\ \end{array} $ |

add_typo

```
Fixpoint add_typo (m n : nat) :=
                                       call stack
                                                  : []
match m with
                                       stack
                                                   : []
\mid 0 \Rightarrow n
| S unused => add_typo m (S n)
                                       redex stack: []
                                       guard env : []
                                       loc env
                                                   : []
fun (n : nat) =>
                                       call stack
                                                  : [tLambda]
match m with
                                       stack
                                                   : []
\mid 0 \Rightarrow n
| S unused => add m (S n)
                                       redex stack : [NoNeed]
end.
                                       guard env : [Large]
                                       loc env
                                                   : [m, add]
match m with
                                       call stack
                                                  : [tCase, tLambda]
\mid 0 \Rightarrow n
                                       stack
                                                   : []
| S unused \Rightarrow add m (S n)
end.
                                       redex stack : [NoNeed]
                                       guard env : [Bound{1}, Large]
                                       loc env
                                                   : [n, m, add]
m (* discriminant *)
                                       call stack : [tRel, tCase, tLambda]
                                       stack
                                                   : []
                                       redex stack : [NoNeed, NoNeed]
                                       guard env : [Bound{1}, Large]
                                       loc env
                                                   : [n, m, add]
n (* 0-th branch *)
                                       call stack
                                                  : [tRel, tCase, tLambda]
                                       stack
                                                   : []
                                       redex stack : [NoNeed, NoNeed]
                                       guard env : [Bound{1}, Large]
                                       loc env
                                                   : [n, m, add]
fun unused : nat \Rightarrow add m (S n)
                                       call stack
                                                  : [tLambda, tCase, tLambda]
(* 1-st branch *)
                                       stack
                                       redex stack : [NoNeed, NoNeed]
                                       guard env : [Bound{1}, Large]
                                       loc env
                                                   : [n, m, add]
add m (S n)
                                       call stack
                                                  : [tApp, tCase, tLambda]
                                       stack
                                                   : []
                                       redex stack : [NoNeed, NoNeed]
                                       guard env : [Strict, Bound{1}, Large]
                                       loc env
                                                   : [unused, n, m, add]
```

| S n | $\begin{bmatrix} call\ stack & : [tApp, tApp, tCase, tLambda] \end{bmatrix}$ |
|---|--|
| | stack : [] |
| | ${\it redex\ stack: [NoNeed, NoNeed, NoNeed]}$ |
| | $guard\ env\ : [Strict, Bound\{1\}, Large]$ |
| | loc env : [unused, n, m, add] |
| m | $call\ stack : [tRel, tApp, tCase, tLambda]$ |
| | stack : [] |
| | ${\it redex\ stack: [NoNeed, NoNeed, NoNeed]}$ |
| | $guard\ env\ : [Strict, Bound\{1\}, Large]$ |
| | loc env : [unused, n, m, add] |
| add | $call\ stack : [tRel, tApp, tCase, tLambda]$ |
| | stack : [SClosure m , SClosure (S n)] |
| | redex stack : [NoNeed, NoNeed] |
| | |
| | $[Strict, Bound\{1\}, Large]$ |
| | |
| (* internal *) | |
| <pre>(* internal *) check_is_subterm (subterm_specif m)</pre> | loc env : [unused, n, m, add] |
| <pre>check_is_subterm (subterm_specif m) (wf_paths nat)</pre> | |
| <pre>check_is_subterm (subterm_specif m)</pre> | |