COM axioms

COM001+1.ax Common axioms for progress/preservation proof
$\forall$ve: valphaEquivalent(ve, ve)   fof('alpha-equiv-refl', axiom)
$\forall ve_2, ve_1$: (valphaEquivalent(ve_1, ve_2) $\Rightarrow$ valphaEquivalent(ve_2, ve_1))   fof('alpha-equiv-sym', axiom)
$\forall ve_2, ve_1, ve_3$: ((valphaEquivalent(ve_1, ve_2) and valphaEquivalent(ve_2, ve_3)) $\Rightarrow$ valphaEquivalent(ve_1, ve_3))   fof('alpha-equ
$\forall$vS, vx, vy, ve: ($\neg$ visFreeVar(vy, ve) $\Rightarrow$ valphaEquivalent(vabs(vx, vS, ve), vabs(vy, vS, vsubst(vx, vvar(vy), ve))))   fof('alp
$\forall$ve, vC, ve_1, vT: ((vtcheck(vC, ve, vT) and valphaEquivalent(ve, ve_1)) $\Rightarrow$ vtcheck(vC, ve_1, vT))   fof('alpha-equiv-typing', a
$\forall$ve, vx, ve_1: (($\neg$ visFreeVar(vx, ve) and valphaEquivalent(ve, ve_1)) $\Rightarrow$ $\neg$ visFreeVar(vx, ve_1))   fof('alpha-equiv-FreeVar', ax

COM problems

COM001-1.p A program correctness theorem
A simple computing state space, with four states - P3, P4, P5, and P8 (the full version of this state space is in the problem COM002-1). There is a branch at P3 such that the following state is either P4 or P8. P8 has a loop back to P3, while P4 leads to termination. The problem is to show that there is a loop in the computation, passing through P3.
follows(goal_state, start_state) $\Rightarrow$ succeeds(goal_state, start_state)   cnf(direct_success, axiom)
(succeeds(goal_state, intermediate_state) and succeeds(intermediate_state, start_state)) $\Rightarrow$ succeeds(goal_state, start_state)
(has(start_state, goto(label)) and labels(label, goal_state)) $\Rightarrow$ succeeds(goal_state, start_state)   cnf(goto_success, axiom)
has(start_state, ifthen(condition, goal_state)) $\Rightarrow$ succeeds(goal_state, start_state)   cnf(conditional_success, axiom)
labels(loop, p_3)   cnf(label_state$_3$, hypothesis)
has(p_3, ifthen(equal_function(register_j, n), p_4))   cnf(state$_3$, hypothesis)
has(p_4, goto(out))   cnf(state$_4$, hypothesis)
follows(p_5, p_4)   cnf(transition_4_to$_5$, hypothesis)
follows(p_8, p_3)   cnf(transition_3_to$_8$, hypothesis)
has(p_8, goto(loop))   cnf(state$_8$, hypothesis)
$\neg$ succeeds(p_3, p_3)   cnf(prove_there_is_a_loop_through_p$_3$, negated_conjecture)

COM001_1.p A program correctness theorem
A simple computing state space, with four states - P3, P4, P5, and P8 (the full version of this state space is in the problem COM002-1). There is a branch at P3 such that the following state is either P4 or P8. P8 has a loop back to P3, while P4 leads to termination. The problem is to show that there is a loop in the computation, passing through P3.
state: $tType   tff(state_type, type)
label: $tType   tff(label_type, type)
statement: $tType   tff(statement_type, type)
register: $tType   tff(register_type, type)
number: $tType   tff(number_type, type)
boolean: $tType   tff(boolean_type, type)
p_3: state   tff(p3_type, type)
p_4: state   tff(p4_type, type)
p_5: state   tff(p5_type, type)
p_8: state   tff(p8_type, type)
n: number   tff(n_type, type)
register_j: register   tff(register_j_type, type)
out: label   tff(out_type, type)
loop: label   tff(loop_type, type)
equal_function: (register × number) → boolean   tff(equal_function_type, type)
goto: label → statement   tff(goto_type, type)
ifthen: (boolean × state) → statement   tff(ifthen_type, type)
follows: (state × state) → $o   tff(follows_type, type)
succeeds: (state × state) → $o   tff(succeeds_type, type)
labels: (label × state) → $o   tff(labels_type, type)
has: (state × statement) → $o   tff(has_type, type)
$\forall$start_state: state, goal_state: state: (follows(goal_state, start_state) $\Rightarrow$ succeeds(goal_state, start_state))   tff(direct_success
$\forall$start_state: state, intermediate_state: state, goal_state: state: ((succeeds(goal_state, intermediate_state) and succeeds(interme
succeeds(goal_state, start_state))   tff(transitivity_of_success, axiom)
$\forall$goal_state: state, label: label, start_state: state: ((has(start_state, goto(label)) and labels(label, goal_state)) $\Rightarrow$ succeeds(goal

∀goal_state: state, condition: boolean, start_state: state: (has(start_state, ifthen(condition, goal_state)) ⇒ succeeds(goal_state,
labels(loop, p_3) tff(label_state$_3$, hypothesis)
has(p_3, ifthen(equal_function(register_j, n), p_4)) tff(state$_3$, hypothesis)
has(p_4, goto(out)) tff(state$_4$, hypothesis)
follows(p_5, p_4) tff(transition_4_to$_5$, hypothesis)
follows(p_8, p_3) tff(transition_3_to$_8$, hypothesis)
has(p_8, goto(loop)) tff(state$_8$, hypothesis)
succeeds(p_3, p_3) tff(prove_there_is_a_loop_through_p$_3$, conjecture)

COM002-1.p A program correctness theorem
A computing state space, with eight states - P1 to P8. P1 leads to P3 via P2. There is a branch at P3 such that the following state is either P4 or P6. P6 leads to P8, which has a loop back to P3, while P4 leads to termination. The problem is to show that there is a loop in the computation, passing through P3.
follows(goal_state, start_state) ⇒ succeeds(goal_state, start_state) cnf(direct_success, axiom)
(succeeds(goal_state, intermediate_state) and succeeds(intermediate_state, start_state)) ⇒ succeeds(goal_state, start_state)
(has(start_state, goto(label)) and labels(label, goal_state)) ⇒ succeeds(goal_state, start_state) cnf(goto_success, axiom)
has(start_state, ifthen(condition, goal_state)) ⇒ succeeds(goal_state, start_state) cnf(conditional_success, axiom)
has(p_1, assign(register_j, n_0)) cnf(state$_1$, hypothesis)
follows(p_2, p_1) cnf(transition_1_to$_2$, hypothesis)
has(p_2, assign(register_k, n_1)) cnf(state$_2$, hypothesis)
labels(loop, p_3) cnf(label_state$_3$, hypothesis)
follows(p_3, p_2) cnf(transition_2_to$_3$, hypothesis)
has(p_3, ifthen(equal_function(register_j, n), p_4)) cnf(state$_3$, hypothesis)
has(p_4, goto(out)) cnf(state$_4$, hypothesis)
follows(p_5, p_4) cnf(transition_4_to$_5$, hypothesis)
follows(p_6, p_3) cnf(transition_3_to$_6$, hypothesis)
has(p_6, assign(register_k, times(n_2, register_k))) cnf(state$_6$, hypothesis)
follows(p_7, p_6) cnf(transition_6_to$_7$, hypothesis)
has(p_7, assign(register_j, register_j + n_1)) cnf(state$_7$, hypothesis)
follows(p_8, p_7) cnf(transition_7_to$_8$, hypothesis)
has(p_8, goto(loop)) cnf(state$_8$, hypothesis)
¬ succeeds(p_3, p_3) cnf(prove_there_is_a_loop_through_p$_3$, negated_conjecture)

COM002-2.p A program correctness theorem.
A computing state space, with eight states - P1 to P8. P1 leads to P3 via P2. There is a branch at P3 such that the following state is either P4 or P6. P6 leads to P8, which has a loop back to P3, while P4 leads to termination. The problem is to show that there is a loop in the computation, passing through P3.
fails(goal_state, start_state) ⇒ ¬ follows(goal_state, start_state) cnf(direct_success, axiom)
fails(goal_state, start_state) ⇒ (fails(goal_state, intermediate_state) or fails(intermediate_state, start_state)) cnf(transitiv
(fails(goal_state, start_state) and has(start_state, goto(label))) ⇒ ¬ labels(label, goal_state) cnf(goto_success, axiom)
fails(goal_state, start_state) ⇒ ¬ has(start_state, ifthen(condition, goal_state)) cnf(conditional_success, axiom)
has(p_1, assign(register_j, n_0)) cnf(state$_1$, hypothesis)
follows(p_2, p_1) cnf(transition_1_to$_2$, hypothesis)
has(p_2, assign(register_k, n_1)) cnf(state$_2$, hypothesis)
labels(loop, p_3) cnf(label_state$_3$, hypothesis)
follows(p_3, p_2) cnf(transition_2_to$_3$, hypothesis)
has(p_3, ifthen(equal_function(register_j, n), p_4)) cnf(state$_3$, hypothesis)
has(p_4, goto(out)) cnf(state$_4$, hypothesis)
follows(p_5, p_4) cnf(transition_4_to$_5$, hypothesis)
follows(p_6, p_3) cnf(transition_3_to$_6$, hypothesis)
has(p_6, assign(register_k, times(n_2, register_k))) cnf(state$_6$, hypothesis)
follows(p_7, p_6) cnf(transition_6_to$_7$, hypothesis)
has(p_7, assign(register_j, register_j + n_1)) cnf(state$_7$, hypothesis)
follows(p_8, p_7) cnf(transition_7_to$_8$, hypothesis)
has(p_8, goto(loop)) cnf(state$_8$, hypothesis)
fails(p_3, p_3) cnf(prove_there_is_a_loop_through_p$_3$, negated_conjecture)

COM003+1.p The halting problem is undecidable
$\exists x$: (algorithm(x) and $\forall y$: (program(y) ⇒ $\forall z$: decides(x, y, z))) ⇒ $\exists w$: (program(w) and $\forall y$: (program(y) ⇒ $\forall z$: decides(w, y, z))) fof(p_1, axiom)

$\forall w$: ((program(w) and $\forall y$: (program(y) $\Rightarrow$ $\forall z$: decides(w, y, z))) $\Rightarrow$ $\forall y, z$: (((program(y) and halts$_2$(y, z)) $\Rightarrow$
(halts$_3$(w, y, z) and outputs(w, good))) and ((program(y) and $\neg$ halts$_2$(y, z)) $\Rightarrow$ (halts$_3$(w, y, z) and outputs(w, bad)))))
$\exists w$: (program(w) and $\forall y$: (((program(y) and halts$_2$(y, y)) $\Rightarrow$ (halts$_3$(w, y, y) and outputs(w, good))) and ((program(y) and
(halts$_3$(w, y, y) and outputs(w, bad))))) $\Rightarrow$ $\exists v$: (program(v) and $\forall y$: (((program(y) and halts$_2$(y, y)) $\Rightarrow$ (halts$_2$(v, y) and ou
(halts$_2$(v, y) and outputs(v, bad)))))   fof(p_3, axiom)
$\exists v$: (program(v) and $\forall y$: (((program(y) and halts$_2$(y, y)) $\Rightarrow$ (halts$_2$(v, y) and outputs(v, good))) and ((program(y) and $\neg$ ha
(halts$_2$(v, y) and outputs(v, bad))))) $\Rightarrow$ $\exists u$: (program(u) and $\forall y$: (((program(y) and halts$_2$(y, y)) $\Rightarrow$ $\neg$ halts$_2$(u, y)) and ((pr
(halts$_2$(u, y) and outputs(u, bad)))))   fof(p_4, axiom)
$\neg \exists x_1$: (algorithm(x_1) and $\forall y_1$: (program(y_1) $\Rightarrow$ $\forall z_1$: decides(x_1, y_1, z_1)))   fof(prove_this, conjecture)

COM003+2.p The halting problem is undecidable
$\forall x$: (program_decides(x) $\iff$ $\forall y$: (program(y) $\Rightarrow$ $\forall z$: decides(x, y, z)))   fof(program_decides_def, axiom)
$\forall x$: (program_program_decides(x) $\iff$ (program(x) and program_decides(x)))   fof(program_program_decides_def, axiom)
$\forall x$: (algorithm_program_decides(x) $\iff$ (algorithm(x) and program_decides(x)))   fof(algorithm_program_decides_def, axi
$\forall x, y$: (program_halts$_2$(x, y) $\iff$ (program(x) and halts$_2$(x, y)))   fof(program_halts2_def, axiom)
$\forall x, y, z, w$: (halts3_outputs(x, y, z, w) $\iff$ (halts$_3$(x, y, z) and outputs(x, w)))   fof(halts3_outputs_def, axiom)
$\forall x, y$: (program_not_halts$_2$(x, y) $\iff$ (program(x) and $\neg$ halts$_2$(x, y)))   fof(program_not_halts2_def, axiom)
$\forall x, y, w$: (halts2_outputs(x, y, w) $\iff$ (halts$_2$(x, y) and outputs(x, w)))   fof(halts2_outputs_def, axiom)
$\forall x, y, z, w$: (program_halts2_halts3_outputs(x, y, z, w) $\iff$ (program_halts$_2$(y, z) $\Rightarrow$ halts3_outputs(x, y, z, w)))   fof(progr
$\forall x, y, z, w$: (program_not_halts2_halts3_outputs(x, y, z, w) $\iff$ (program_not_halts$_2$(y, z) $\Rightarrow$ halts3_outputs(x, y, z, w)))
$\forall x, y, w$: (program_halts2_halts2_outputs(x, y, w) $\iff$ (program_halts$_2$(y, y) $\Rightarrow$ halts2_outputs(x, y, w)))   fof(program_ha
$\forall x, y, w$: (program_not_halts2_halts2_outputs(x, y, w) $\iff$ (program_not_halts$_2$(y, y) $\Rightarrow$ halts2_outputs(x, y, w)))   fof(pro
$\exists x$: algorithm_program_decides(x) $\Rightarrow$ $\exists w$: program_program_decides(w)   fof(p_1, axiom)
$\forall w$: (program_program_decides(w) $\Rightarrow$ $\forall y, z$: (program_halts2_halts3_outputs(w, y, z, good) and program_not_halts2_halts3_ou
$\exists w$: (program(w) and $\forall y$: (program_halts2_halts3_outputs(w, y, y, good) and program_not_halts2_halts3_outputs(w, y, y, bad))
$\exists v$: (program(v) and $\forall y$: (program_halts2_halts2_outputs(v, y, good) and program_not_halts2_halts2_outputs(v, y, bad)))   f
$\exists v$: (program(v) and $\forall y$: (program_halts2_halts2_outputs(v, y, good) and program_not_halts2_halts2_outputs(v, y, bad))) $\Rightarrow$
$\exists u$: (program(u) and $\forall y$: ((program_halts$_2$(y, y) $\Rightarrow$ $\neg$ halts$_2$(u, y)) and program_not_halts2_halts2_outputs(u, y, good)))   fo
$\neg \exists x$: algorithm_program_decides(x)   fof(prove_this, conjecture)

COM003+3.p The halting problem is undecidable
$\exists x$: (algorithm(x) and $\forall y$: (program(y) $\Rightarrow$ $\forall z$: decides(x, y, z))) $\Rightarrow$ $\exists w$: (program(w) and $\forall y$: (program(y) $\Rightarrow$
$\forall z$: decides(w, y, z)))   fof(p_1, axiom)
$\forall w$: ((program(w) and $\forall y$: (program(y) $\Rightarrow$ $\forall z$: decides(w, y, z))) $\Rightarrow$ $\forall y, z$: (((program(y) and halts$_2$(y, z)) $\Rightarrow$
(halts$_3$(w, y, z) and outputs(w, good))) and ((program(y) and $\neg$ halts$_2$(y, z)) $\Rightarrow$ (halts$_3$(w, y, z) and outputs(w, bad)))))
$\forall w$: ((program(w) and $\forall y, z$: (((program(y) and halts$_2$(y, z)) $\Rightarrow$ (halts$_3$(w, y, z) and outputs(w, good))) and ((program(y) an
(halts$_3$(w, y, z) and outputs(w, bad))))) $\Rightarrow$ $\exists v$: (program(v) and $\forall y$: (((program(y) and halts$_3$(w, y, y) and outputs(w, good)
$\neg$ halts$_2$(v, y)) and ((program(y) and halts$_3$(w, y, y) and outputs(w, bad)) $\Rightarrow$ (halts$_2$(v, y) and outputs(v, bad))))))   fof(p
$\neg \exists x_1$: (algorithm(x_1) and $\forall y_1$: (program(y_1) $\Rightarrow$ $\forall z_1$: decides(x_1, y_1, z_1)))   fof(prove_this, conjecture)

COM003_1.p The halting problem is undecidable
program: \$tType   tff(program_type, type)
algorithm: \$tType   tff(algorithm_type, type)
input: \$tType   tff(input_type, type)
output: \$tType   tff(output_type, type)
bad: output   tff(bad_type, type)
good: output   tff(good_type, type)
decides: (algorithm $\times$ program $\times$ input) $\rightarrow$ \$o   tff(decides_type, type)
halts$_2$: (program $\times$ input) $\rightarrow$ \$o   tff(halts2_type, type)
halts$_3$: (program $\times$ program $\times$ input) $\rightarrow$ \$o   tff(halts3_type, type)
outputs: (program $\times$ output) $\rightarrow$ \$o   tff(outputs_type, type)
algorithm_of: program $\rightarrow$ algorithm   tff(algorithm_of_type, type)
as_input: program $\rightarrow$ input   tff(as_input_type, type)
$\exists x$: algorithm: $\forall y$: program, z: input: decides(x, y, z) $\Rightarrow$ $\exists w$: program: $\forall y$: program, z: input: decides(algorithm_of(w), y, z)
$\forall w$: program, y: program, z: input: (decides(algorithm_of(w), y, z) $\Rightarrow$ $\forall y$: program, z: input: ((halts$_2$(y, z) $\Rightarrow$
(halts$_3$(w, y, z) and outputs(w, good))) and ($\neg$ halts$_2$(y, z) $\Rightarrow$ (halts$_3$(w, y, z) and outputs(w, bad)))))   tff(p_2, axiom)
$\exists w$: program: $\forall y$: program: ((halts$_2$(y, as_input(y)) $\Rightarrow$ (halts$_3$(w, y, as_input(y)) and outputs(w, good))) and ($\neg$ halts$_2$(y, as_i
(halts$_3$(w, y, as_input(y)) and outputs(w, bad)))) $\Rightarrow$ $\exists v$: program: $\forall y$: program: ((halts$_2$(y, as_input(y)) $\Rightarrow$ (halts$_2$(v, as_inpu
(halts$_2$(v, as_input(y)) and outputs(v, bad))))   tff(p_3, axiom)

∃v: program: ∀y: program: ((halts$_2$(y, as_input(y)) ⇒ (halts$_2$(v, as_input(y)) and outputs(v, good))) and (¬ halts$_2$(y, as_inpu
(halts$_2$(v, as_input(y)) and outputs(v, bad)))) ⇒ ∃u: program: ∀y: program: ((halts$_2$(y, as_input(y)) ⇒ ¬ halts$_2$(u, as_input(
(halts$_2$(u, as_input(y)) and outputs(u, bad)))) tff(p_4, axiom)
¬ ∃x_1: algorithm: ∀y_1: program, z_1: input: decides(x_1, y_1, z_1) tff(prove_this, conjecture)

COM004-1.p Part of completeness of resolution
Part of [Bun83]'s proof of the completeness of resolution uses the notion of failure nodes. This proves a special case when a parent is the empty failure node.
(failure_node(x, or(c, p)) and failure_node(y, or(d, q)) and contradictory(p, q) and siblings(x, y)) ⇒ failure_node(parent_of(x,
contradictory($-x$, x) cnf(not_x_contradicts_x, axiom)
contradictory(x, $-x$) cnf(x_contradicts_not_x, axiom)
siblings(left_child_of(x), right_child_of(x)) cnf(n_left_and_n_right_are_siblings, axiom)
failure_node(n_left, or(empty, atom)) cnf(n_left_is_atom, hypothesis)
failure_node(n_right, or(empty, −atom)) cnf(n_right_is_not_atom, hypothesis)
n_left = left_child_of(n) cnf(n_left_equals_left_child_of_n, hypothesis)
n_right = right_child_of(n) cnf(n_right_equals_right_child_of_n, hypothesis)
¬ failure_node(z, or(empty, empty)) cnf(goal_is_there_an_empty_node, negated_conjecture)

COM007+1.p Preservation of the Diamond Property under reflexive closure
reflexive_rewrite(a, b) and reflexive_rewrite(a, c) fof(assumption, axiom)
∀a: ((reflexive_rewrite(b, a) and reflexive_rewrite(c, a)) ⇒ goal) fof(goal_ax, axiom)
∀a: a=a fof(reflexivity, axiom)
∀a, b: (a=b ⇒ b=a) fof(symmtery, axiom)
∀a, b, c: ((a=b and reflexive_rewrite(b, c)) ⇒ reflexive_rewrite(a, c)) fof(substitution, axiom)
∀a, b: (a=b ⇒ reflexive_rewrite(a, b)) fof(equalish_in_reflexive_rewrite, axiom)
∀a, b: (rewrite(a, b) ⇒ reflexive_rewrite(a, b)) fof(rewrite_in_reflexive_rewrite, axiom)
∀a, b: (reflexive_rewrite(a, b) ⇒ (a=b or rewrite(a, b))) fof(equalish_or_rewrite, axiom)
∀a, b, c: ((rewrite(a, b) and rewrite(a, c)) ⇒ ∃d: (rewrite(b, d) and rewrite(c, d))) fof(rewrite_diamond, axiom)
goal fof(goal_to_be_proved, conjecture)

COM007+2.p Preservation of the Diamond Property under reflexive closure
reflexive_rewrite(a, b) and reflexive_rewrite(a, c) fof(assumption, axiom)
∀a: ((reflexive_rewrite(b, a) and reflexive_rewrite(c, a)) ⇒ goal) fof(goal_ax, axiom)
∀a, b: ($a = b$ ⇒ reflexive_rewrite(a, b)) fof(equal_in_reflexive_rewrite, axiom)
∀a, b: (rewrite(a, b) ⇒ reflexive_rewrite(a, b)) fof(rewrite_in_reflexive_rewrite, axiom)
∀a, b: (reflexive_rewrite(a, b) ⇒ ($a = b$ or rewrite(a, b))) fof(equal_or_rewrite, axiom)
∀a, b, c: ((rewrite(a, b) and rewrite(a, c)) ⇒ ∃d: (rewrite(b, d) and rewrite(c, d))) fof(rewrite_diamond, axiom)
goal fof(goal_to_be_proved, conjecture)

COM008+1.p Induction step in Newman's Lemma
∀a: ((transitive_reflexive_rewrite(b, a) and transitive_reflexive_rewrite(c, a)) ⇒ goal) fof(found, axiom)
transitive_reflexive_rewrite(a, b) and transitive_reflexive_rewrite(a, c) fof(assumption, axiom)
∀a: a=a fof(reflexivity, axiom)
∀a, b: (a=b ⇒ b=a) fof(symmetry, axiom)
∀a, b: (a=b ⇒ transitive_reflexive_rewrite(a, b)) fof(equality_in_transitive_reflexive_rewrite, axiom)
∀a, b: (rewrite(a, b) ⇒ transitive_reflexive_rewrite(a, b)) fof(rewrite_in_transitive_reflexive_rewrite, axiom)
∀a, b, c: ((transitive_reflexive_rewrite(a, b) and transitive_reflexive_rewrite(b, c)) ⇒ transitive_reflexive_rewrite(a, c)) fof(tr
∀a, b, c: ((rewrite(a, b) and rewrite(a, c)) ⇒ ∃d: (transitive_reflexive_rewrite(b, d) and transitive_reflexive_rewrite(c, d))) fo
∀a, b, c: ((rewrite(a, a) and transitive_reflexive_rewrite(a, b) and transitive_reflexive_rewrite(a, c)) ⇒ ∃d: (transitive_reflexive_
∀a, b: (transitive_reflexive_rewrite(a, b) ⇒ (a=b or ∃c: (rewrite(a, c) and transitive_reflexive_rewrite(c, b)))) fof(equalish_o
goal fof(goal_to_be_proved, conjecture)

COM008+2.p Induction step in Newman's Lemma
∀a: ((transitive_reflexive_rewrite(b, a) and transitive_reflexive_rewrite(c, a)) ⇒ goal) fof(found, axiom)
transitive_reflexive_rewrite(a, b) and transitive_reflexive_rewrite(a, c) fof(assumption, axiom)
∀a, b: ($a = b$ ⇒ transitive_reflexive_rewrite(a, b)) fof(equality_in_transitive_reflexive_rewrite, axiom)
∀a, b: (rewrite(a, b) ⇒ transitive_reflexive_rewrite(a, b)) fof(rewrite_in_transitive_reflexive_rewrite, axiom)
∀a, b, c: ((transitive_reflexive_rewrite(a, b) and transitive_reflexive_rewrite(b, c)) ⇒ transitive_reflexive_rewrite(a, c)) fof(tr
∀a, b, c: ((rewrite(a, b) and rewrite(a, c)) ⇒ ∃d: (transitive_reflexive_rewrite(b, d) and transitive_reflexive_rewrite(c, d))) fo
∀a, b, c: ((rewrite(a, a) and transitive_reflexive_rewrite(a, b) and transitive_reflexive_rewrite(a, c)) ⇒ ∃d: (transitive_reflexive_
∀a, b: (transitive_reflexive_rewrite(a, b) ⇒ ($a = b$ or ∃c: (rewrite(a, c) and transitive_reflexive_rewrite(c, b)))) fof(equal_or_
goal fof(goal_to_be_proved, conjecture)

COM009-1.p Problem about UNITY theory

include('Axioms/MSC001-2.ax')

include('Axioms/MSC001-0.ax')

c_in(c_Relation_OId, c_UNITY_OActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OId__in__Acts$_0$, axiom)

c_in(c_Relation_OId, c_UNITY_OAllowedActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OId__in__AllowedActs$_0$, a

c_UNITY_Oall__total(v_F, t_a) ⇒ c_UNITY_Ototalize(v_F, t_a) = v_F cnf(cls_UNITY_Oall__total__imp__totalize$_0$, axiom)

c_UNITY_Oall__total(c_UNITY_Ototalize(v_F, t_a), t_a) cnf(cls_UNITY_Oall__total__totalize$_0$, axiom)

c_in(v_F, c_UNITY_Oconstrains(v_A, c_UNIV, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__UNIV2$_0$, axio

c_in(v_F, c_UNITY_Oconstrains(c_UNIV, v_B, t_a), tc_UNITY_Oprogram(t_a)) ⇒ v_B = c_UNIV cnf(cls_UNITY_Oconstr

c_in(v_F, c_UNITY_Oconstrains(c_UNIV, c_UNIV, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__UNIV__if

c_in(v_F, c_UNITY_Oconstrains(v_A, c_emptyset, t_a), tc_UNITY_Oprogram(t_a)) ⇒ v_A = c_emptyset cnf(cls_UNITY_O

c_in(v_F, c_UNITY_Oconstrains(c_emptyset, c_emptyset, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__em

c_in(v_F, c_UNITY_Oconstrains(c_emptyset, v_B, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__empty$_0$, ax

c_insert(c_Relation_OId, c_UNITY_OActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) = c_UNITY_OActs(v_F, t_a) cnf(cls_UNITY_

c_insert(c_Relation_OId, c_UNITY_OAllowedActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) = c_UNITY_OAllowedActs(v_F, t_a)

c_UNITY_Ototalize(v_F, t_a) = v_F ⇒ ¬ c_UNITY_Oall__total(v_F, t_a) cnf(cls_conjecture$_0$, negated_conjecture)

c_UNITY_Oall__total(v_F, t_a) or c_UNITY_Ototalize(v_F, t_a) = v_F cnf(cls_conjecture$_1$, negated_conjecture)

COM009-2.p Problem about UNITY theory

c_UNITY_Ototalize(v_F, t_a) = v_F ⇒ ¬ c_UNITY_Oall__total(v_F, t_a) cnf(cls_conjecture$_0$, negated_conjecture)

c_UNITY_Oall__total(v_F, t_a) or c_UNITY_Ototalize(v_F, t_a) = v_F cnf(cls_conjecture$_1$, negated_conjecture)

c_UNITY_Oall__total(v_F, t_a) ⇒ c_UNITY_Ototalize(v_F, t_a) = v_F cnf(cls_UNITY_Oall__total__imp__totalize$_0$, axiom)

c_UNITY_Oall__total(c_UNITY_Ototalize(v_F, t_a), t_a) cnf(cls_UNITY_Oall__total__totalize$_0$, axiom)

COM010-1.p Problem about UNITY theory

include('Axioms/MSC001-1.ax')

include('Axioms/MSC001-0.ax')

c_UNITY_OActs(c_UNITY_Omk__program(c_Pair(v_init, c_Pair(v_acts, v_allowed, tc_set(tc_set(tc_prod(t_a, t_a))), tc_set(tc_se

c_insert(c_Relation_OId, v_acts, tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OActs__eq$_0$, axiom)

c_UNITY_OActs(v_F, t_a) ≠ c_emptyset cnf(cls_UNITY_OActs__nonempty$_0$, axiom)

c_UNITY_OAllowedActs(c_UNITY_Omk__program(c_Pair(v_init, c_Pair(v_acts, v_allowed, tc_set(tc_set(tc_prod(t_a, t_a))), tc_s

c_insert(c_Relation_OId, v_allowed, tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OAllowedActs__eq$_0$, axiom)

c_in(c_Relation_OId, c_UNITY_OActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OId__in__Acts$_0$, axiom)

c_in(c_Relation_OId, c_UNITY_OAllowedActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OId__in__AllowedActs$_0$, a

c_UNITY_OInit(c_UNITY_Omk__program(c_Pair(v_y, c_Pair(v_acts, v_allowed, tc_set(tc_set(tc_prod(t_a, t_a))), tc_set(tc_set(tc

v_y cnf(cls_UNITY_OInit__eq$_0$, axiom)

c_insert(c_Relation_OId, c_UNITY_OActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) = c_UNITY_OActs(v_F, t_a) cnf(cls_UNITY_

c_insert(c_Relation_OId, c_UNITY_OAllowedActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) = c_UNITY_OAllowedActs(v_F, t_a)

c_UNITY_Omk__program(c_Pair(c_UNITY_OInit(v_y, t_a), c_Pair(c_UNITY_OActs(v_y, t_a), c_UNITY_OAllowedActs(v_y, t_a

v_y cnf(cls_UNITY_Osurjective__mk__program$_0$, axiom)

c_UNITY_OInit(v_F, t_a) = c_UNITY_OInit(v_G, t_a) cnf(cls_conjecture$_0$, negated_conjecture)

c_UNITY_OActs(v_F, t_a) = c_UNITY_OActs(v_G, t_a) cnf(cls_conjecture$_1$, negated_conjecture)

c_UNITY_OAllowedActs(v_F, t_a) = c_UNITY_OAllowedActs(v_G, t_a) cnf(cls_conjecture$_2$, negated_conjecture)

v_F ≠ v_G cnf(cls_conjecture$_3$, negated_conjecture)

COM010-2.p Problem about UNITY theory

c_UNITY_OInit(v_F, t_a) = c_UNITY_OInit(v_G, t_a) cnf(cls_conjecture$_0$, negated_conjecture)

c_UNITY_OActs(v_F, t_a) = c_UNITY_OActs(v_G, t_a) cnf(cls_conjecture$_1$, negated_conjecture)

c_UNITY_OAllowedActs(v_F, t_a) = c_UNITY_OAllowedActs(v_G, t_a) cnf(cls_conjecture$_2$, negated_conjecture)

v_F ≠ v_G cnf(cls_conjecture$_3$, negated_conjecture)

c_UNITY_Omk__program(c_Pair(c_UNITY_OInit(v_y, t_a), c_Pair(c_UNITY_OActs(v_y, t_a), c_UNITY_OAllowedActs(v_y, t_a

v_y cnf(cls_UNITY_Osurjective__mk__program$_0$, axiom)

COM011-1.p Problem about UNITY theory

include('Axioms/MSC001-2.ax')

include('Axioms/MSC001-0.ax')

c_in(c_Relation_OId, c_UNITY_OActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OId__in__Acts$_0$, axiom)

c_in(c_Relation_OId, c_UNITY_OAllowedActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) cnf(cls_UNITY_OId__in__AllowedActs$_0$, a

c_in(v_F, c_UNITY_Oconstrains(v_A, c_UNIV, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__UNIV2$_0$, axio

c_in(v_F, c_UNITY_Oconstrains(c_UNIV, v_B, t_a), tc_UNITY_Oprogram(t_a)) ⇒ v_B = c_UNIV cnf(cls_UNITY_Oconstr

c_in(v_F, c_UNITY_Oconstrains(c_UNIV, c_UNIV, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__UNIV__if

c_in(v_F, c_UNITY_Oconstrains(v_A, c_emptyset, t_a), tc_UNITY_Oprogram(t_a)) $\Rightarrow$ v_A = c_emptyset cnf(cls_UNITY_C
c_in(v_F, c_UNITY_Oconstrains(c_emptyset, c_emptyset, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__em
c_in(v_F, c_UNITY_Oconstrains(c_emptyset, v_B, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__empty$_0$, ax
(c_in(v_F, c_UNITY_Oconstrains(v_A, v_A_H, t_a), tc_UNITY_Oprogram(t_a)) and c_lessequals(v_B, v_A, tc_set(t_a))) $\Rightarrow$
c_in(v_F, c_UNITY_Oconstrains(v_B, v_A_H, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__weaken__L$_0$, ax
c_insert(c_Relation_OId, c_UNITY_OActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) = c_UNITY_OActs(v_F, t_a) cnf(cls_UNITY_
c_insert(c_Relation_OId, c_UNITY_OAllowedActs(v_F, t_a), tc_set(tc_prod(t_a, t_a))) = c_UNITY_OAllowedActs(v_F, t_a)
(c_in(v_F, c_UNITY_Oconstrains(c_minus(v_A, v_B, tc_set(t_a)), c_union(v_A, v_B, t_a), t_a), tc_UNITY_Oprogram(t_a)) and c_
c_in(v_F, c_WFair_Oensures(v_A, v_B, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_WFair_OensuresI$_0$, axiom)
c_in(v_F, c_WFair_Oensures(v_A, v_B, t_a), tc_UNITY_Oprogram(t_a)) $\Rightarrow$ c_in(v_F, c_WFair_OleadsTo(v_A, v_B, t_a), tc_UNIT
(c_in(v_F, c_WFair_Otransient(v_A, t_a), tc_UNITY_Oprogram(t_a)) and c_lessequals(v_B, v_A, tc_set(t_a))) $\Rightarrow$ c_in(v_F, c_W
c_in(v_F, c_UNITY_Oconstrains(v_A, c_union(v_A, v_B, t_a), t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_conjecture$_0$, negated_c
c_in(v_F, c_WFair_Otransient(v_A, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_conjecture$_1$, negated_conjecture)
$\neg$ c_in(v_F, c_WFair_OleadsTo(v_A, v_B, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_conjecture$_2$, negated_conjecture)

COM011-2.p Problem about UNITY theory
c_in(v_F, c_UNITY_Oconstrains(v_A, c_union(v_A, v_B, t_a), t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_conjecture$_0$, negated_c
c_in(v_F, c_WFair_Otransient(v_A, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_conjecture$_1$, negated_conjecture)
$\neg$ c_in(v_F, c_WFair_OleadsTo(v_A, v_B, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_conjecture$_2$, negated_conjecture)
c_in(v_c, c_minus(v_A, v_B, tc_set(t_a)), t_a) $\Rightarrow$ c_in(v_c, v_A, t_a) cnf(cls_Set_ODiffE$_1$, axiom)
c_in(c_Main_OsubsetI$_{-1}$(v_A, v_B, t_a), v_A, t_a) or c_lessequals(v_A, v_B, tc_set(t_a)) cnf(cls_Set_OsubsetI$_0$, axiom)
c_in(c_Main_OsubsetI$_{-1}$(v_A, v_B, t_a), v_B, t_a) $\Rightarrow$ c_lessequals(v_A, v_B, tc_set(t_a)) cnf(cls_Set_OsubsetI$_1$, axiom)
(c_in(v_F, c_UNITY_Oconstrains(v_A, v_A_H, t_a), tc_UNITY_Oprogram(t_a)) and c_lessequals(v_B, v_A, tc_set(t_a))) $\Rightarrow$
c_in(v_F, c_UNITY_Oconstrains(v_B, v_A_H, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_UNITY_Oconstrains__weaken__L$_0$, ax
(c_in(v_F, c_UNITY_Oconstrains(c_minus(v_A, v_B, tc_set(t_a)), c_union(v_A, v_B, t_a), t_a), tc_UNITY_Oprogram(t_a)) and c_
c_in(v_F, c_WFair_Oensures(v_A, v_B, t_a), tc_UNITY_Oprogram(t_a)) cnf(cls_WFair_OensuresI$_0$, axiom)
c_in(v_F, c_WFair_Oensures(v_A, v_B, t_a), tc_UNITY_Oprogram(t_a)) $\Rightarrow$ c_in(v_F, c_WFair_OleadsTo(v_A, v_B, t_a), tc_UNIT
(c_in(v_F, c_WFair_Otransient(v_A, t_a), tc_UNITY_Oprogram(t_a)) and c_lessequals(v_B, v_A, tc_set(t_a))) $\Rightarrow$ c_in(v_F, c_W

COM012+1.p Newman's lemma on rewriting systems 01, 00 expansion
$\forall w_0$: (aElement$_0$(w_0) $\Rightarrow$ \$true) fof(mElmSort, axiom)
$\forall w_0$: (aRewritingSystem$_0$(w_0) $\Rightarrow$ \$true) fof(mRelSort, axiom)
$\forall w_0, w_1$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1)) $\Rightarrow$ $\forall w_2$: (aReductOfIn$_0$(w_2, w_0, w_1) $\Rightarrow$ aElement$_0$(w_2))) fof(mR
$\forall w_0, w_1$: ((aElement$_0$(w_0) and aElement$_0$(w_1)) $\Rightarrow$ (iLess$_0$(w_0, w_1) $\Rightarrow$ \$true)) fof(mWFOrd, axiom)
$\forall w_0, w_1, w_2$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1) and aElement$_0$(w_2)) $\Rightarrow$ (sdtmndtplgtdt$_0$(w_0, w_1, w_2) $\Rightarrow$
\$true)) fof(mTCbr, axiom)
$\forall w_0, w_1, w_2$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1) and aElement$_0$(w_2)) $\Rightarrow$ (sdtmndtplgtdt$_0$(w_0, w_1, w_2) $\iff$
(aReductOfIn$_0$(w_2, w_0, w_1) or $\exists w_3$: (aElement$_0$(w_3) and aReductOfIn$_0$(w_3, w_0, w_1) and sdtmndtplgtdt$_0$(w_3, w_1, w_2))))) fo
$\forall w_0, w_1, w_2, w_3$: ((aElement$_0$($w_0$) and aRewritingSystem$_0$($w_1$) and aElement$_0$($w_2$) and aElement$_0$($w_3$)) $\Rightarrow$ ((sdtmndtplgtdt$_0$
sdtmndtplgtdt$_0$(w_0, w_1, w_3))) fof(mTCTrans, axiom)
$\forall w_0, w_1, w_2$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1) and aElement$_0$(w_2)) $\Rightarrow$ (sdtmndtasgtdt$_0$(w_0, w_1, w_2) $\iff$
($w_0 = w_2$ or sdtmndtplgtdt$_0$(w_0, w_1, w_2)))) fof(mTCRDef, definition)
aElement$_0$(xx) and aRewritingSystem$_0$(xR) and aElement$_0$(xy) and aElement$_0$(xz) fof(m$_{-349}$, hypothesis)
(sdtmndtasgtdt$_0$(xx, xR, xy) and sdtmndtasgtdt$_0$(xy, xR, xz)) $\Rightarrow$ sdtmndtasgtdt$_0$(xx, xR, xz) fof(m$_{--}$, conjecture)

COM012+3.p Newman's lemma on rewriting systems 01, 02 expansion
$\forall w_0$: (aElement$_0$(w_0) $\Rightarrow$ \$true) fof(mElmSort, axiom)
$\forall w_0$: (aRewritingSystem$_0$(w_0) $\Rightarrow$ \$true) fof(mRelSort, axiom)
$\forall w_0, w_1$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1)) $\Rightarrow$ $\forall w_2$: (aReductOfIn$_0$(w_2, w_0, w_1) $\Rightarrow$ aElement$_0$(w_2))) fof(mR
$\forall w_0, w_1$: ((aElement$_0$(w_0) and aElement$_0$(w_1)) $\Rightarrow$ (iLess$_0$(w_0, w_1) $\Rightarrow$ \$true)) fof(mWFOrd, axiom)
$\forall w_0, w_1, w_2$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1) and aElement$_0$(w_2)) $\Rightarrow$ (sdtmndtplgtdt$_0$(w_0, w_1, w_2) $\Rightarrow$
\$true)) fof(mTCbr, axiom)
$\forall w_0, w_1, w_2$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1) and aElement$_0$(w_2)) $\Rightarrow$ (sdtmndtplgtdt$_0$(w_0, w_1, w_2) $\iff$
(aReductOfIn$_0$(w_2, w_0, w_1) or $\exists w_3$: (aElement$_0$(w_3) and aReductOfIn$_0$(w_3, w_0, w_1) and sdtmndtplgtdt$_0$(w_3, w_1, w_2))))) fo
$\forall w_0, w_1, w_2, w_3$: ((aElement$_0$($w_0$) and aRewritingSystem$_0$($w_1$) and aElement$_0$($w_2$) and aElement$_0$($w_3$)) $\Rightarrow$ ((sdtmndtplgtdt$_0$
sdtmndtplgtdt$_0$(w_0, w_1, w_3))) fof(mTCTrans, axiom)
$\forall w_0, w_1, w_2$: ((aElement$_0$(w_0) and aRewritingSystem$_0$(w_1) and aElement$_0$(w_2)) $\Rightarrow$ (sdtmndtasgtdt$_0$(w_0, w_1, w_2) $\iff$
($w_0 = w_2$ or sdtmndtplgtdt$_0$(w_0, w_1, w_2)))) fof(mTCRDef, definition)
aElement$_0$(xx) and aRewritingSystem$_0$(xR) and aElement$_0$(xy) and aElement$_0$(xz) fof(m$_{-349}$, hypothesis)

$((\mathrm{xx} = \mathrm{xy} \text{ or } ((\mathrm{aReductOfIn}_0(\mathrm{xy}, \mathrm{xx}, \mathrm{xR}) \text{ or } \exists w_0\colon (\mathrm{aElement}_0(w_0) \text{ and } \mathrm{aReductOfIn}_0(w_0, \mathrm{xx}, \mathrm{xR}) \text{ and } \mathrm{sdtmndtplgtdt}_0(w_0, \mathrm{xR}$
$\mathrm{xz} \text{ or } ((\mathrm{aReductOfIn}_0(\mathrm{xz}, \mathrm{xy}, \mathrm{xR}) \text{ or } \exists w_0\colon (\mathrm{aElement}_0(w_0) \text{ and } \mathrm{aReductOfIn}_0(w_0, \mathrm{xy}, \mathrm{xR}) \text{ and } \mathrm{sdtmndtplgtdt}_0(w_0, \mathrm{xR}, \mathrm{xz}))) \text{ a}$
$(\mathrm{xx} = \mathrm{xz} \text{ or } \mathrm{aReductOfIn}_0(\mathrm{xz}, \mathrm{xx}, \mathrm{xR}) \text{ or } \exists w_0\colon (\mathrm{aElement}_0(w_0) \text{ and } \mathrm{aReductOfIn}_0(w_0, \mathrm{xx}, \mathrm{xR}) \text{ and } \mathrm{sdtmndtplgtdt}_0(w_0, \mathrm{xR}, \mathrm{xz}$

COM024^5.p TPS problem THM9
A very naive version of the recursion theorem. TM X Y is the output of Turing machine X on input Y, TH F is the number of a Turing machine that computes function F.
cTM: $i → $i → $i   thf(cTM, type)
cTH: ($i → $i) → $i   thf(cTH, type)
$\forall g\colon \$i \to \$i\colon (\mathrm{cTM}@(\mathrm{cTH}@g)) = g \Rightarrow \forall f\colon \$i \to \$i\colon \exists n\colon \$i\colon (\mathrm{cTM}@(f@n)) = (\mathrm{cTM}@n)$ thf(cTHM_9, conjecture)

COM123+1.p T-Weak-FreeVar-abs-1 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT))   fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT))   fof('T-Strong',
∀vx, vS, vC, vT: ((¬ visFreeVar(vx, veabs) and vtcheck(vC, veabs, vT)) ⇒ vtcheck(vbind(vx, vS, vC), veabs, vT))   fof('T-W
$\forall$vx, vS, vC, vy, vS_1, vT: ((vx ≠ vy and ¬ visFreeVar(vx, vabs(vy, vS_1, veabs)) and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) ⇒ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT))   fof('T-Weak-FreeVar-abs-1', conjecture)

COM124+1.p T-Weak-FreeVar-abs-2 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT))   fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT))   fof('T-Strong',
∀vx, vS, vC, vy, vS_1, ve_1, vT: ((vx ≠ vy and ¬ visFreeVar(vx, vabs(vy, vS_1, ve_1)) and vtcheck(vC, vabs(vy, vS_1, ve_1), vT)) ⇒ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, ve_1), vT))   fof('T-Weak-FreeVar-abs-1-gen', axiom)
∀vx, vS, vC, vy, vS_1, vT: ((vx = vy and ¬ visFreeVar(vx, vabs(vy, vS_1, veabs)) and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) ⇒ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT))   fof('T-Weak-FreeVar-abs-2', conjecture)

COM125+1.p T-Weak-FreeVar-abs step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT))   fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT))   fof('T-Strong',
∀vx, vS, vC, vT: ((¬ visFreeVar(vx, veabs) and vtcheck(vC, veabs, vT)) ⇒ vtcheck(vbind(vx, vS, vC), veabs, vT))   fof('T-W
∀vx, vS, vC, vy, vS_1, vT: ((vx ≠ vy and ¬ visFreeVar(vx, vabs(vy, vS_1, veabs)) and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) ⇒ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT))   fof('T-Weak-FreeVar-abs-1', axiom)
∀vx, vS, vC, vy, vS_1, vT: ((vx = vy and ¬ visFreeVar(vx, vabs(vy, vS_1, veabs)) and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) ⇒ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT))   fof('T-Weak-FreeVar-abs-2', axiom)
∀vx, vS, vC, vy, vS_1, vT: ((¬ visFreeVar(vx, vabs(vy, vS_1, veabs)) and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) ⇒ vtcheck(vbinc

COM126+1.p T-Weak-FreeVar-app step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT))   fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT))   fof('T-Strong',
∀vx, vS, vC, vT: ((¬ visFreeVar(vx, ve1app) and vtcheck(vC, ve1app, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve1app, vT))   fof(
∀vx, vS, vC, vT: ((¬ visFreeVar(vx, ve2app) and vtcheck(vC, ve2app, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve2app, vT))   fof(
∀vx, vS, vC, vT: ((¬ visFreeVar(vx, vapp(ve1app, ve2app)) and vtcheck(vC, vapp(ve1app, ve2app), vT)) ⇒ vtcheck(vbind(vx,

COM127+1.p T-Weak-FreeVar-var step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT))   fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT))   fof('T-Strong',
∀vx, vS, vC, vy, vT: ((¬ visFreeVar(vx, vvar(vy)) and vtcheck(vC, vvar(vy), vT)) ⇒ vtcheck(vbind(vx, vS, vC), vvar(vy), vT)

COM128+1.p Fresh-unequal-var-3 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
∀vT, vC, vx, ve, vy, vS, ve_1, vT_2: ((vx ≠ vy and ¬ visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), v
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, ve_1)), vT_2)) fof('T-subst-abs-2-gen', axiom)
∀ve, ve_1, vx, vfresh: (vfresh = vgensym(vapp(vapp(ve, ve_1), vvar(vx))) ⇒ vx ≠ vfresh) fof('fresh-unequal-var-3', conjectur

COM129+1.p Fresh-free-2 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
∀vT, vC, vx, ve, vy, vS, ve_1, vT_2: ((vx ≠ vy and ¬ visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), v
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, ve_1)), vT_2)) fof('T-subst-abs-2-gen', axiom)
∀ve, ve_1, vx, vfresh: (vfresh = vgensym(vapp(vapp(ve, ve_1), vvar(vx))) ⇒ vx ≠ vfresh) fof('fresh-unequal-var-3', axiom)
∀ve, vx, vfresh, ve_1: (vfresh = vgensym(vapp(vapp(ve, ve_1), vvar(vx))) ⇒ ¬ visFreeVar(vfresh, ve_1)) fof('fresh-free-2', conj

COM130+1.p T-subst-abs-1 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
∀vT, vC, vx, ve, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), veabs, vT_2)) ⇒ vtcheck(vC, vsubst(vx, ve, veabs),
∀vT, vC, vx, ve, vy, vS, ve_1, vT_2: ((vx = vy and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(vy, vS, ve_1), vT_2)) ⇒
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, ve_1)), vT_2)) fof('T-subst-abs-1', conjecture)

COM131+1.p T-subst-abs-2 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
∀vT, vC, vx, ve, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), veabs, vT_2)) ⇒ vtcheck(vC, vsubst(vx, ve, veabs),
∀vT, vC, vx, ve, vy, vS, ve_1, vT_2: ((vx = vy and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(vy, vS, ve_1), vT_2)) ⇒
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, ve_1)), vT_2)) fof('T-subst-abs-1', axiom)
∀vT, vC, vx, ve, vy, vS, vT_2: ((vx ≠ vy and ¬ visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, veabs)), vT_2)) fof('T-subst-abs-2', conjecture)

COM132+1.p T-subst-abs-3 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
∀vT, vC, vx, ve, vy, vS, ve_1, vT_2: ((vx ≠ vy and ¬ visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), v
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, ve_1)), vT_2)) fof('T-subst-abs-2-gen', axiom)
∀ve, ve_1, vx, vfresh: (vfresh = vgensym(vapp(vapp(ve, ve_1), vvar(vx))) ⇒ vx ≠ vfresh) fof('fresh-unequal-var-3', axiom)
∀ve, vx, vfresh, ve_1: (vfresh = vgensym(vapp(vapp(ve, ve_1), vvar(vx))) ⇒ ¬ visFreeVar(vfresh, ve_1)) fof('fresh-free-2', axio

$\forall$vT, vC, vx, ve, vy, vS, vT_2: ((vx $\neq$ vy and visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(vy
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, veabs)), vT_2)) fof('T-subst-abs-3', conjecture)

COM133+1.p T-subst-abs step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) $\Rightarrow$ vtcheck(vC, ve, vT)) fof('T-Strong',
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
$\forall$vT, vC, vx, ve, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), veabs, vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve, veabs),
$\forall$vT, vC, vx, ve, vy, vS, ve_1, vT_2: ((vx = vy and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(vy, vS, ve_1), vT_2)) $\Rightarrow$
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, ve_1)), vT_2)) fof('T-subst-abs-1', axiom)
$\forall$vT, vC, vx, ve, vy, vS, vT_2: ((vx $\neq$ vy and $\neg$ visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, veabs)), vT_2)) fof('T-subst-abs-2', axiom)
$\forall$vT, vC, vx, ve, vy, vS, vT_2: ((vx $\neq$ vy and visFreeVar(vy, ve) and vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(vy
vtcheck(vC, vsubst(vx, ve, vabs(vy, vS, veabs)), vT_2)) fof('T-subst-abs-3', axiom)
$\forall$vT, vC, vx, ve, vy, vS, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vabs(vy, vS, veabs), vT_2)) $\Rightarrow$ vtcheck(vC, v

COM134+1.p T-subst-app step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) $\Rightarrow$ vtcheck(vC, ve, vT)) fof('T-Strong',
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
$\forall$vT, vC, vx, ve, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), ve_1, vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve, ve_1), vT_2)
$\forall$vT, vC, vx, ve, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), ve_2, vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve, ve_2), vT_2)
$\forall$vT, vC, vx, ve, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vapp(ve_1, ve_2), vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve

COM135+1.p T-subst-var step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) $\Rightarrow$ vtcheck(vC, ve, vT)) fof('T-Strong',
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
$\forall$vT, vC, vx, ve, vy, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), vvar(vy), vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve, v

COM136+1.p T-Strong-abs step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
$\forall$vx, vS, vC, vT: (($\neg$ visFreeVar(vx, veabs) and vtcheck(vbind(vx, vS, vC), veabs, vT)) $\Rightarrow$ vtcheck(vC, veabs, vT)) fof('T-S
$\forall$vx, vS, vC, vy, vS_1, vT: (($\neg$ visFreeVar(vx, vabs(vy, vS_1, veabs)) and vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT)) $\Rightarrow$
vtcheck(vC, vabs(vy, vS_1, veabs), vT)) fof('T-Strong-abs', conjecture)

COM137+1.p T-Strong-app step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
$\forall$vx, vS, vC, vT: (($\neg$ visFreeVar(vx, ve1app) and vtcheck(vbind(vx, vS, vC), ve1app, vT)) $\Rightarrow$ vtcheck(vC, ve1app, vT)) fof(
$\forall$vx, vS, vC, vT: (($\neg$ visFreeVar(vx, ve2app) and vtcheck(vbind(vx, vS, vC), ve2app, vT)) $\Rightarrow$ vtcheck(vC, ve2app, vT)) fof(
$\forall$vx, vS, vC, vT: (($\neg$ visFreeVar(vx, vapp(ve1app, ve2app)) and vtcheck(vbind(vx, vS, vC), vapp(ve1app, ve2app), vT)) $\Rightarrow$
vtcheck(vC, vapp(ve1app, ve2app), vT)) fof('T-Strong-app', conjecture)

COM138+1.p T-Strong-var step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-

$\forall$vx, vS, vC, vy, vT: (($\neg$ visFreeVar(vx, vvar(vy)) and vtcheck(vbind(vx, vS, vC), vvar(vy), vT)) $\Rightarrow$ vtcheck(vC, vvar(vy), vT))

COM139+1.p T-Weak-abs-1 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, veabs, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), veabs, vT)) fof
$\forall$vx, vS, vC, vy, vS_1, vT: ((vx $\neq$ vy and vlookup(vx, vC) = vnoType and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT)) fof('T-Weak-abs-1', conjecture)

COM140+1.p T-Weak-abs-2 step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')
$\forall$vx, vS, vC, vy, vS_1, ve_1, vT: ((vx $\neq$ vy and vlookup(vx, vC) = vnoType and vtcheck(vC, vabs(vy, vS_1, ve_1), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, ve_1), vT)) fof('T-Weak-abs-1-gen', axiom)
$\forall$vx, vS, vC, vy, vS_1, vT: ((vx = vy and vlookup(vx, vC) = vnoType and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT)) fof('T-Weak-abs-2', conjecture)

COM141+1.p T-Weak-abs step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, veabs, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), veabs, vT)) fof
$\forall$vx, vS, vC, vy, vS_1, vT: ((vx $\neq$ vy and vlookup(vx, vC) = vnoType and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT)) fof('T-Weak-abs-1', axiom)
$\forall$vx, vS, vC, vy, vS_1, vT: ((vx = vy and vlookup(vx, vC) = vnoType and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), vabs(vy, vS_1, veabs), vT)) fof('T-Weak-abs-2', axiom)
$\forall$vx, vS, vC, vy, vS_1, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, vabs(vy, vS_1, veabs), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, v

COM142+1.p T-Weak-app step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve1app, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve1app, vT))
$\forall$vx, vS, vC, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve2app, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve2app, vT))
$\forall$vx, vS, vC, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, vapp(ve1app, ve2app), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), va

COM143+1.p T-Weak-var step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, vy, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, vvar(vy), vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), vvar(vy), vT

COM144+1.p T-Preservation-T-abs step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
$\forall$vT, vC, vx, ve, ve_2, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), ve_2, vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve, ve_2),
$\forall$vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) $\Rightarrow$ vtcheck(vC, ve, vT)) fof('T-Strong',
$\forall$vC, veout, vT: ((vreduce(ve_1) = vsomeExp(veout) and vtcheck(vC, ve_1, vT)) $\Rightarrow$ vtcheck(vC, veout, vT)) fof('T-Preserva
$\forall$vx, vS, vC, veout, vT: ((vreduce(vabs(vx, vS, ve_1)) = vsomeExp(veout) and vtcheck(vC, vabs(vx, vS, ve_1), vT)) $\Rightarrow$ vtcheck(vC, veout, vT)) fof('T-Preservation-T-abs', conjecture)

COM145+1.p T-Preservation-T-app step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
$\forall$vx, vS, vC, ve, vT: (($\neg$ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) $\Rightarrow$ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
$\forall$vT, vC, vx, ve, ve_2, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), ve_2, vT_2)) $\Rightarrow$ vtcheck(vC, vsubst(vx, ve, ve_2),

∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vC, veout, vT: ((vreduce(ve_1) = vsomeExp(veout) and vtcheck(vC, ve_1, vT)) ⇒ vtcheck(vC, veout, vT)) fof('T-Preserva
∀vC, veout, vT: ((vreduce(ve_2) = vsomeExp(veout) and vtcheck(vC, ve_2, vT)) ⇒ vtcheck(vC, veout, vT)) fof('T-Preserva
∀vC, veout, vT: ((vreduce(vapp(ve_1, ve_2)) = vsomeExp(veout) and vtcheck(vC, vapp(ve_1, ve_2), vT)) ⇒ vtcheck(vC, veout, vT

COM146+1.p T-Preservation-T-var step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-Weak-F
∀vT, vC, vx, ve, ve_2, vT_2: ((vtcheck(vC, ve, vT) and vtcheck(vbind(vx, vT, vC), ve_2, vT_2)) ⇒ vtcheck(vC, vsubst(vx, ve, ve_2),
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vx, vC, veout, vT: ((vreduce(vvar(vx)) = vsomeExp(veout) and vtcheck(vC, vvar(vx), vT)) ⇒ vtcheck(vC, veout, vT)) f

COM147+1.p T-Progress-T-abs step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vT: ((vtcheck(vempty, ve_1, vT) and ¬ visValue(ve_1)) ⇒ ∃veout: vreduce(ve_1) = vsomeExp(veout)) fof('T-Progress-T-al
∀vTin, vx, vS: ((vtcheck(vempty, vabs(vx, vS, ve_1), vTin) and ¬ visValue(vabs(vx, vS, ve_1))) ⇒ ∃veout: vreduce(vabs(vx, vS, v
vsomeExp(veout)) fof('T-Progress-T-abs', conjecture)

COM148+1.p T-Progress-T-app step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vT: ((vtcheck(vempty, ve_1, vT) and ¬ visValue(ve_1)) ⇒ ∃veout: vreduce(ve_1) = vsomeExp(veout)) fof('T-Progress-T-ap
∀vT: ((vtcheck(vempty, ve_2, vT) and ¬ visValue(ve_2)) ⇒ ∃veout: vreduce(ve_2) = vsomeExp(veout)) fof('T-Progress-T-ap
∀vT: ((vtcheck(vempty, vapp(ve_1, ve_2), vT) and ¬ visValue(vapp(ve_1, ve_2))) ⇒ ∃veout: vreduce(vapp(ve_1, ve_2)) =
vsomeExp(veout)) fof('T-Progress-T-app', conjecture)

COM149+1.p T-Progress-T-var step in progress/preservation proof
This problem is a step within the proof of progress and preservation for the standard type system for the simply-typed lambda calculus.
include('Axioms/COM001+0.ax')
∀vx, vS, vC, ve, vT: ((vlookup(vx, vC) = vnoType and vtcheck(vC, ve, vT)) ⇒ vtcheck(vbind(vx, vS, vC), ve, vT)) fof('T-
∀vx, vS, vC, ve, vT: ((¬ visFreeVar(vx, ve) and vtcheck(vbind(vx, vS, vC), ve, vT)) ⇒ vtcheck(vC, ve, vT)) fof('T-Strong',
∀vT, vx: ((vtcheck(vempty, vvar(vx), vT) and ¬ visValue(vvar(vx))) ⇒ ∃veout: vreduce(vvar(vx)) = vsomeExp(veout)) f

COM150+1.p Axioms for progress/preservation proof
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')

COM151+1.p Common axioms for progress/preservation proof
include('Axioms/COM001+0.ax')
include('Axioms/COM001+1.ax')