3 Inversion of Typing Relations

- If $\langle integer \rangle : T$, then $T = \text{Int}$. (T-Int)
- If $\langle real \rangle : T$, then $T = \text{Real}$. (T-Real)
- If $\langle string \rangle : T$, then $T = \text{String}$. (T-String)
- If $\texttt{true} : T$, then $T = \text{Bool}$. (T-True)
- If $\texttt{false} : T$, then $T = \text{Bool}$. (T-False)
- If $\Gamma \vdash x : T$, then $x : T \in \Gamma$. (T-Var)
- If $\Gamma; \mathrm{R}; \mathrm{E}; \mathrm{L} \vdash \texttt{fn}(x : T_{\text{arg}}) \to T_{\text{ret}}\ \texttt{throw}\ T_{\text{err}}\ \texttt{do}\ S\ \texttt{end} : T$, then there exist T_{arg}, T_{ret}, T_{err} such that $T = T_{\text{arg}} \to T_{\text{ret}}$ throw T_{err} and $\Gamma, x : T_{\text{arg}};\ \mathrm{R} :: T_{\text{ret}};\ \mathrm{E} = \{T_{\text{err}}\};\ \mathrm{L} = \emptyset \vdash S : \text{Void}$. (T-Fn)
- If $\Gamma; \mathrm{R}; \mathrm{E}; \mathrm{L}; \mu \vdash f(x : T_{arg}) \to T_{ret}\ \texttt{throw}\ T_{err}\ \texttt{do}\ S\ \texttt{end} : \text{Void}$, then there exists T_{arg}, T_{ret}, T_{err} such that $\Gamma, f : T_{arg} \to T_{ret}$ throw $T_{err}, x : T_{arg}; \mathrm{R} :: T_{ret}; \mathrm{E} = \{T_{err}\}; \mathrm{L} = \emptyset; \mu[f \mapsto \text{const}] \vdash V : \text{Void}$. (T-FnDef)
- If $\Gamma; \mathrm{R} \vdash \texttt{return}\ e : \text{Void}$, then $\mathrm{R} \neq \emptyset$ and there exists T_{ret} such that $\Gamma;\ \mathrm{R} :: T_{\text{ret}} \vdash e : T_{\text{ret}}$. (T-Return)
- If $\Gamma; \mathrm{E} \vdash f(e) : T$, then there exists $T_{\text{arg}}, T_{\text{ret}}, T_{\text{err}}$ such that $T = T_{\text{ret}}$, $\Gamma \vdash f : T_{\text{arg}} \to T_{\text{ret}}$ throw T_{err}, $\Gamma \vdash e : T_{\text{arg}}$, and $T_{\text{err}} \in \mathrm{E}$. (T-App)
- If $\Gamma, x : T;\ \mu[x \mapsto \texttt{var}] \vdash \texttt{let}\ x = e : \text{Void}$, then $x \notin \text{dom}(\Gamma)$ and $\Gamma;\ \mu \vdash e : T$. (T-Let)
- If $\Gamma, x : T;\ \mu[x \mapsto \texttt{const}] \vdash \texttt{const}\ x = e : \text{Void}$, then $x \notin \text{dom}(\Gamma)$ and $\Gamma;\ \mu \vdash e : T$. (T-Const)
- If $\Gamma \vdash S_1\ ;\ S_2 : \text{Void}$, then $\Gamma \vdash S_1 : \text{Void}$ and $\Gamma \vdash S_2 : \text{Void}$. (T-Sequence)
- If $\Gamma; \mu \vdash x = e : \text{Void}$, then $\Gamma(x) = T$, $\mu(x) = \texttt{var}$, $\Gamma; \mu \vdash e : S$, and $S <: T$. (T-Assign)
- If $\Gamma \vdash \texttt{do}\ S\ \texttt{end} : \text{Void}$, then $\Gamma \vdash S : \text{Void}$. (T-Scope)
- If $\Gamma \vdash \texttt{if}\ e\ \texttt{do}\ S_1\ \texttt{end else do}\ S_2\ \texttt{end} : \text{Void}$, then $\Gamma \vdash e : \text{Bool}$, $\Gamma \vdash S_1 : \text{Void}$, and $\Gamma \vdash S_2 : \text{Void}$. (T-IfElse)
- If $\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \text{lub}(T_1, T_2)$, then $\Gamma \vdash e_1 : \text{Bool}$, $\Gamma \vdash e_2 : T_1$, and $\Gamma \vdash e_3 : T_2$. (T-IfThen)
- If $\Gamma \vdash \texttt{case}\ p\ \texttt{if}\ g\ \Rightarrow e : T_r$, then $\Gamma, p \vdash g : \text{Bool}$ and $\Gamma, p \vdash e : T_r$.
- If $\Gamma \vdash \texttt{match}\ e\ \texttt{as}\ x\ \texttt{case}\ p_i\ \texttt{if}\ g_i \Rightarrow e_i\ \ldots\ \texttt{end} : T$, then there exists $T_s, T_1, \ldots, T_n$ such that $\Gamma \vdash e : T_s$, for all i, $\Gamma, x : T_s \vdash \texttt{case}\ p_i\ \texttt{if}\ g_i \Rightarrow e_i : T_i$, and $T = \text{lub}(T_1, \ldots, T_n)$. (T-MatchExpr)

- If $\Gamma \vdash \texttt{case}\ p\ \texttt{if}\ g\ \texttt{do}\ S\ \texttt{end} : \text{Void}$, then $\Gamma, p \vdash g : \text{Bool}$ and $\Gamma, p \vdash S : \text{Void}$. (T-CaseStmt)

- If $\Gamma \vdash \texttt{match}\ e\ \texttt{as}\ x\ \texttt{case}\ p_i\ \texttt{if}\ g_i\ \texttt{do}\ S_i\ \texttt{end}\ \ldots\ \texttt{end} : \text{Void}$, then there exists T_s such that $\Gamma \vdash e : T_s$ and, for all i, $\Gamma, x : T_s \vdash \texttt{case}\ p_i\ \texttt{if}\ g_i\ \texttt{do}\ S_i\ \texttt{end} : \text{Void}$. (T-MatchStmt)

- If $\Gamma;\ \text{L} \vdash \texttt{while}\ e\ \texttt{do}\ S\ \texttt{end} : \text{Void}$, then $\Gamma \vdash e : \text{Bool}$ and $\Gamma;\ \text{L}, \ell \vdash S : \text{Void}$. (T-While)

- If $\Gamma;\ \text{L} \vdash \texttt{break} : \text{Void}$, then $\text{L} \neq \emptyset$. (T-Break)

- If $\Gamma;\ \text{L} \vdash \texttt{continue} : \text{Void}$, then $\text{L} \neq \emptyset$. (T-Continue)

- If $\Gamma \vdash \texttt{throw}\ e : \text{Void}$, then $\Gamma \vdash e : T_{err}$ and $T_{err} \in \text{E}$. (T-Throw)

- If $\Gamma \vdash \texttt{try}\ e\ \texttt{else}\ e_{\text{def}} : \text{lub}(T_1, T_2)$, then $\Gamma \vdash e : T_1$ and $\Gamma \vdash e_{\text{def}} : T_2$. (T-TryElse)

- If $\Gamma \vdash \texttt{try}\ S\ \texttt{catch}\ T_i\ \texttt{as}\ x_i\ \texttt{do}\ S_i\ \texttt{end}\ \ldots\ \texttt{end} : \text{Void}$, then there exists $T_1, \ldots, T_n$ such that $\Gamma \vdash S : \text{Void}$ and, for all i, $\Gamma, x_i : T_i \vdash S_i : \text{Void}$. (T-TryCatch)

4 Canonical Forms

1. If $v : \text{Int}$, then v is an integer.
2. If $v : \text{Real}$, then v is a real number.
3. If $v : \text{String}$, then v is a string.
4. If $v : \text{Bool}$, then $v = \text{true}$ or $v = \text{false}$.
5. If $v : T_1 \to T_2 \text{ throw } T_3$, then $v = \texttt{fn}(x : T_1) \to T_2 \texttt{ throw } T_3 \texttt{ do} \ldots \texttt{end}$.
6. If $v : T$ and T is a struct type, then $v = \{f_1 = v_1, \ldots, f_n = v_n\}$ where $f_i : T_i$ and $v_i : T_i$.
7. If $v : T$ and T is a class type, then $v = \ell$ for some location ℓ such that $\sigma(\ell) = \{f_1 = v_1, \ldots, f_n = v_n\}$ where $f_i : T_i$ and $v_i : T_i$.
8. If $v : T$ and T is an enum type, then $v = \texttt{case}\ C$ for some case C of T.
9. If $v : T$ and T is an enum struct type, then $v = \texttt{case}\ C(v_1, \ldots, v_n)$ for some constructor C of T where $v_i : T_i$.
10. If $v : T$ and T is an enum class type, then $v = \ell$ for some location ℓ such that $\sigma(\ell) = \texttt{case}\ C(v_1, \ldots, v_n)$ where $v_i : T_i$.

5 Progress

If t is a closed, well-typed term, then either t is a value or else there is some t' with $t \to t'$.

Proof: By induction on a derivation of $t : T$.

Case T-INT, T-REAL, T-STRING, T-TRUE, T-FALSE:
Since t is a value, no further reduction can be applied. Thus, the property holds vacuously.

Case T-VAR: $t = x$
This case cannot occur, since t must be closed.

Case T-FN: $t = \texttt{fn}(t_1 : T_1) \to T_2\ \texttt{do} \ldots \texttt{end}$
t is a function definition, thus a value. The property holds vacuously.

Case T-FNDEF, T-RETURN, T-LET, T-CONST, T-SEQUENCE, T-ASSIGN, T-SCOPE, T-IFELSE, T-CASESTMT, T-MATCHSTMT, T-WHILE, T-BREAK, T-CONTINUE, T-THROW, T-TRYCATCH:
Since t is a statement (typed Void), no reduction applies. The property holds vacuously.

Case T-APP: $t = t_1(t_2) \quad t_1 : T_{11} \to T_{12}\,\text{throw}\,T_{13} \quad t_2 : T_{11} \quad T = T_{12}\,\text{throw}\,T_{13}$
By the induction hypothesis, either t_1 is a value or else there is some t_1' such that $t_1 \to t_1'$, and likewise for t_2. If t_1 can take a step, then rule E-APPSTEP applies to t. If t_1 is a value and t_2 can take a step, the rule E-APPARGS applies. Finally, if both t_1 and t_2 are values, then the canonical form 5 tells us that t_1 has the form $\texttt{fn}(x : T_{11}) \to T_{12}\ \texttt{throw}\ T_{13}\ \texttt{do}\ \ldots \texttt{end}$, so rule E-APP applies to t. Therefore, in all cases, t can take a step.

Case T-IFTHEN: $t = \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \quad t_1 : \text{Bool} \quad t_2 = T_2 \quad t_3 = T_3$
By the induction hypothesis, either t_1 is a value or else there is some t_1' such that $t_1 \to t_1'$. If t_1 can take a step, the rule E-IFSTEP applies to t. If t_1 is a value, then the canonical forms 4 assures us that it must be either true or false, in which case either E-IFTRUE or E-IFFALSE applies to t. Therefore, in all cases, t can take a step.

Case T-MATCHEXPR:
$t = \texttt{match}\ t_1\ \texttt{as}\ x\ \texttt{case}\ p_i\ \texttt{if}\ g_i \Rightarrow t_2 \ldots \texttt{end} \quad t_1 : T_s \quad x : T_s \quad g_i : \text{Bool}$
$t_2 : T_r$

By the induction hypothesis, either t_1 is a value or there is some t_1' such that $t_1 \to t_1'$. If t_1 can take a step, the rule E-MATCHSTEP applies to t_1.

If t_1 is a value and t_1 matches some p_i, by the induction hypothesis under $\Gamma, x : T_s$, either g_i is a value or it can take a step. If $g_i \to g_i'$, then E-MatchIfStep applies. If g_i is a value, then the canonical forms 4 assures us that it must be either `true` or `false`, in which case either E-MatchIfTrue or E-MatchIfFalse applies.

If t_1 is a value but does not match any p_i, then E-MatchExhausted applies.

Therefore, in all cases, t can take a step.

Case T-TryElse: $t = \texttt{try}\ t_1\ \texttt{else}\ t_2 \quad t_1 : T_1 \quad t_2 : T_2$
By the induction hypothesis, either t_1 is a value or else there is some t_1' such that $t_1 \to t_1'$. If t_1 can take a step, the rule E-TryElseStep applies to t. If t_1 is a value, E-TryElseNoThrow applies to t. If t_1 throws, E-TryElseThrow applies to t. Therefore, in all cases, t can take a step.

6 Preservation

If $t : T$ and $t \to t'$, then $t' : T$.

Proof: By induction on a derivation of $t : T$. At each step of the induction, we assume that the desired property holds for all subderivations (i.e., that if $s : S$ and $s \to s'$, then $s : S'$, whenever $s : S$ is proved by subderivation of the present one) and proceed by case analysis on the final rule in the derivation.

Case T-Int: $t = \langle \mathit{integer} \rangle \quad T = \texttt{Int}$

If the last rule in the derivation is T-Int, then we know from the form of this rule that t must be an integer value and T must be `Int`. But then t is a value, so it cannot be the case that $t \to t'$ for any t', and the requirements of the theorem are vacuously satisfied.

Case T-Real: $t = \langle \mathit{real} \rangle \quad T = \texttt{Real}$

If the last rule in the derivation is T-Real, then we know from the form of this rule that t must be a real number value and T must be `Real`. But then t is a value, so it cannot be the case that $t \to t'$ for any t', and the requirements of the theorem are vacuously satisfied.

Case T-String: $t = \langle \mathit{string} \rangle \quad T = \texttt{String}$

If the last rule in the derivation is T-String, then we know from the form of this rule that t must be a string value and T must be `String`. But then t is a value, so it cannot be the case that $t \to t'$ for any t', and the requirements of the theorem are vacuously satisfied.

Case T-True: $t = \texttt{true} \quad T = \texttt{Bool}$

If the last rule in the derivation is T-True, then we know from the form of this rule that t must be the constant `true` and T must be `Bool`. But then t is a value, so it cannot be the case that $t \to t'$ for any t', and the requirements of the theorem are vacuously satisfied.

Case T-False: $t = \texttt{false} \quad T = \texttt{Bool}$

If the last rule in the derivation is T-False, then we know from the form of this rule that t must be the constant `false` and T must be `Bool`. But then t is a value, so it cannot be the case that $t \to t'$ for any t', and the requirements of the theorem are vacuously satisfied.

Case T-Var: $t = x$

Cannot happen since there are no evaluation rules for variables.

Case T-FN: $t = \texttt{fn}(t_1 : T_1) \to T_2$ $\texttt{do} \ldots \texttt{end}$
Cannot happen since t is already a value.

Case T-FNDEF, T-RETURN, T-LET, T-CONST, T-SEQUENCE, T-ASSIGN, T-SCOPE, T-IFELSE, T-CASESTMT, T-MATCHSTMT, T-WHILE, T-BREAK, T-CONTINUE, T-THROW, T-TRYCATCH:
Cannot happen since t is a statement (typed Void), no reduction applies.

Case T-APP: $t = t_1(t_2) \quad t_1 : T_{11} \to T_{12}\, \texttt{throw}\, T_{13} \quad t_2 : T_{11} \quad T = T_{12}\, \texttt{throw}\, T_{13}$
From the evaluation rules, we see that there are three rules by which $t \to t'$ can be derived: E-APPSTEP, E-APPARGS, and E-APP. Proceed by cases.

Subcase E-APPSTEP: $t_1 \to t_1' \quad t' = t_1'(t_2)$
By the induction hypothesis and the subderivation of the original typing of t_1, we have $\Gamma \vdash t_1' : T_{11} \to T_{12}\, \texttt{throw}\, T_{13}$. Combining with the fact that $\Gamma \vdash t_2 : T_{11}$, we can conclude that $\Gamma;\ E \cup T_{13} \vdash t' : T_{12}$ with rule T-APP.

Subcase E-APPARGS: $t_1 = v_1 \quad t_2 \to t_2' \quad t' = v_1(t_2')$
Similar to the E-APPSTEP subcase.

Subcase E-APPSTEP: $t_1 = \texttt{fn}(x : T_{11}) \to T_{12}\, \texttt{throw}\, T_{13}\, \texttt{do} \ldots \texttt{return}\, t_{12} \ldots \texttt{end} \quad t' = [x \mapsto v_2]t_{12}$
Using the inversion lemma of T-FN, we get $\Gamma, x : T_{11};\ \mathrm{R} :: T_{12};\ \mathrm{E} = \{T_{13}\};\ \mathrm{L} \neq \emptyset \vdash S : \mathrm{Void}$. From $\mathrm{R} :: T_{12}$, we obatin $\Gamma \vdash t' : T_{12}$.

Case T-IFTHEN: $t = \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \quad t_1 : \mathrm{Bool} \quad t_2 = T_2 \quad t_3 = T_3$
If the last rule in derivation is T-IFTHEN, then we know from the form of this rule that t must have the form $\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3$, for some t_1, t_2, and t_3. We must also have subderivations with conditions $t_1 : \texttt{Bool}$, $t_2 : T$, $t_3 : T$, and the result type is $T = \mathrm{lub}(T_2, T_3)$. From the evaluation rules, we see that there are three rules by which $t \to t'$ can be derived: E-IFTRUE, E-IFFALSE, and E-IFSTEP. Proceed by cases.

Subcase E-IFTRUE: $t_1 = \texttt{true} \quad t' = t_2$
If $t \to t'$ is derived using E-IFTRUE, then from the form of this rule we see that t_1 must be $\texttt{true}$ and the resulting term t' is the second subexpression t_2. This means we are finished, since we know (by the assumptions of the T-IFTHEN case) that $t_2 : T_2$ and $T_2 <: T = \mathrm{lub}(T_2, T_3)$. Therefore, by rule T-SUB, we have $t' : T$, which is what we need.

Subcase E-IFFALSE: $t_1 = \texttt{false} \quad t' = t_3$
Similar to the E-IFTRUE subcase.

Subcase E-IFSTEP: $t_1 \to t_1' \quad t' = \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3$
From the assumptions of the T-IFTHEN case, we have a subderivation of the original typing derivation whose conclusion is $t_1 : \texttt{Bool}$. We can apply the induction hypothesis to this subderivation, obtaining $t_1' : \texttt{Bool}$.

Combining this with the facts (from the assumptions of the T-IfThen case) that $t_2 : T$ and $t_3 : T$, we can apply rule T-IfThen to conclude that `if` t_1' `then` t_2 `else` $t_3 : T$, that is $t' : T$.

Case T-MatchExpr: $t =$ `match` t_1 `as` x `case` p_i `if` $g_i \Rightarrow e_i \ldots$ `end`

If the last rule in the derivation is T-MatchExpr, then we know from the form of this rule that t must have the form `match` t_1 `as` x `case` p_i `if` $g_i \Rightarrow e_i \ldots$ `end` for some t_1, p_i, g_i, and e_i. We must also have a subderivation with $t_1 : T_s$ and, for every i, $\Gamma, x : T_s \vdash$ `case` p_i `if` $g_i \Rightarrow e_i : T_i$, and $T = \text{lub}(T_1, \ldots, T_n)$. From the evaluation rules, there are three kinds of steps for match expressions: E-MatchStep, E-MatchMatched, E-MatchGuardStep, E-MatchGuardTrue, E-MatchGuardFalse, E-MatchNext, and E-MatchExhausted. Proceed by cases.

Subcase E-MatchStep: $t_1 \to t_1' \quad t' =$ `match` t_1' `as` x `case` p_i `if` $g_i \Rightarrow e_i \ldots$ `end`

From the assumptions of the T-MatchExpr case, we have a subderivation of the original typing derivation whose conclusion is $t_1 : T_s$. We can apply the induction hypothesis to this subderivation, obtaining $t' : T$.

Subcase E-MatchMatched: $t_1 = v \quad t' =$ `match` v `as` x `case` p_i' `if` $g_i \Rightarrow e_i \ldots$ `end`

The branches remain unchanged and well-typed, so by rule T-MatchExpr we have $\Gamma \vdash t' : T$.

Subcase E-MatchGuardStep: $g_i \to g_i' \quad t' =$ `match` v `as` x `case` p_i' `if` $g_i' \Rightarrow e_i \ldots$ `end`

By the induction hypothesis and the subderivation typing g_i, we have $\Gamma \vdash g_i' : \text{Bool}$. Applying rule T-MatchExpr yields $\Gamma \vdash t' : T$.

Subcase E-MatchGuardTrue: $t_1 = v \quad g_i =$ `true` $\quad t' = e_i$

From the assumptions of the T-MatchExpr case, we have $\Gamma \vdash e_i : T_i$ and $T_i <: T = \text{lub}(T_1, \ldots, T_n)$. Therefore, by rule T-Sub, we have $\Gamma \vdash t' : T$.

Subcase E-MatchGuardFalse: $t' =$ `match` v `as` $x \ldots$ `end`

Similar to the E-MatchMatched subcase.

Subcase E-MatchNext: $t' =$ `match` v `as` $x \ldots$ `end`

Similar to the E-MatchMatched subcase.

Subcase E-MatchExhausted: $t_1 = v \quad t' =$ `error`$(\ldots)$

If $t \to t'$ is derived using E-MatchExhausted, then t' results in an error.

Case T-TryElse: $t =$ `try` t_1 `else` $t_2 \quad t_1 : T_1 \quad t_2 : T_2$

If the last rule in derivation is T-TryElse, then we know from the form of this rule that t must have the form `try` t_1 `else` t_2, for some t_1 and t_2. We must also have subderivations with conditions t_1 :

T, $t_2 : T$, and $T = \text{lub}(T_1, T_2)$. From the evaluation rules, there are three rules by which $t \to t'$ can be derived: E-TryElseStep, E-TryElseNoThrow, and E-TryElseThrow. Proceed by cases.

Subcase E-TryElseStep: $t_1 \to t_1' \quad t' = \texttt{try}\ t_1'\ \texttt{else}\ t_2$
From the assumptions of the T-TryElse case, we have a subderivation with conclusion $t_1 : T_1$. By the induction hypothesis, $t_1' : T_1$. Since $t_2 : T_2$ and $T = \text{lub}(T_1, T_2)$, rule T-TryElse yields $\Gamma \vdash t' : T$.

Subcase E-TryElseNoThrow: $t_1 = v_1 \quad t' = v_1$
If $t \to t'$ is derived using E-TryElseNoThrow, then t_1 is a value v_1. From the assumptions of the T-TryElse case, we have $v_1 : T_1$ and $T_1 <: T = \text{lub}(T_1, T_2)$. Rule T-Sub yields $t' : T$.

Subcase E-TryElseThrow: $t_1 = \texttt{throw}\ v \quad t' = t_2$
If $t \to t'$ is derived using E-TryElseThrow, then t_1 must be a throw statement. The resulting term t' is t_2, and from the assumptions of the T-TryElse case we know $t_2 : T_2$ and $T_2 <: T = \text{lub}(T_1, T_2)$. Rule T-Sub yields $t' : T$.