Getting stuck

If a term t makes no sense, we introduce no rule to define its evaluation, so there is no t' such that $t \rightsquigarrow t'$ Example: consider this top-level expression:

if (5) 3 **else** 7

the expression 5 cannot be evaluated further and is a constant, but there are no rules for when condition of \mathbf{if} is a number constant; there are only rules for boolean constants.

Such terms, that are not constants and have no applicable rules, are called **stuck**, because no further steps are possible.

Stuck terms indicate errors. Type checking is a way to detect them **statically**, without trying to (dynamically) execute a program and see if it will get stuck or produce result.

Type Rules: Program

「トナ:で

After the definition of operational semantics, we define type rules (also inductively). Given initial program (e, t) define

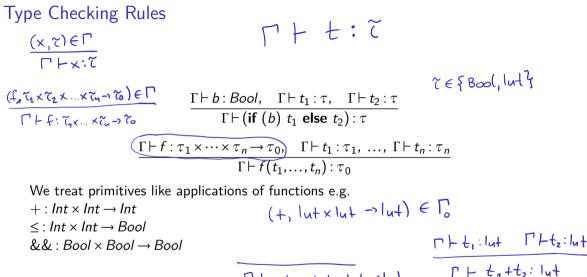
$$\Gamma_0 = \{ (f, \tau_1 \times \cdots \times \tau_n \to \tau_0) \mid (f, _, (\tau_1, \ldots, \tau_n), t_f, \tau_0) \in e \}$$

We say program type checks iff:

(1) the top-level expression type checks:

and

$$\begin{array}{c}
\Gamma_{0} \vdash t:\tau \\ \Gamma_{0} \oplus \Gamma_{1} = \{(x,\tau) \mid (x,\tau) \in \Gamma_{1} \lor (x,\tau) \in \Gamma_{1} \lor (x,\tau) \in \Gamma_{1} \lor (x,\tau) \in \Gamma_{1} \lor (x,\tau) \in \Gamma_{2} \lor (x,\tau) \lor (x,\tau) \in \Gamma_{2} \lor (x,\tau) \lor (x,\tau) \to \Gamma_{2} \lor (x,\tau) \lor (x$$



TH +: IntxInt-)lut

Soundness through progress and preservation

Soundness theorem: *if program type checks, its evaluation does not get stuck.* Proof uses the following two lemmas (a common approach):

progress: if a program type checks, it is not stuck: if

then either t is a constant (execution is done) or there exists t' such that t → t'
preservation: if a program type checks and makes one → step, then the result again type checks in our simple system: it type checks and has the same type: if

 $\Gamma \vdash t : \tau$

 $\Gamma \vdash t : \tau$

and $t \rightsquigarrow t'$ then

$$\mathsf{T}\vdash t':\tau$$

Proof of progress and preservation - case of if

We prove conjunction of progress and preservation by induction on term t such that $\Gamma \vdash t : \tau$. The operational semantics defines the non-error cases of an interpreter, which enables case analysis. Consider **if**. By type checking rules, **if** can only type check if its condition b type checks and has type Bool. By inductive hypothesis and progress either b is constant or it can be reduced to a b'. If it is constant one of these rules apply (so we get progress): $t = if(b) t_1 e^{t} e^{t} t_2$

 $\begin{array}{c} \Gamma \vdash b: Bool \\ \Gamma \vdash t_{1}: \tau \\ \Gamma \vdash b': Bool \end{array} \qquad \overline{(\text{if } (true) \ t_{1} \ \text{else} \ t_{2}) \leadsto t_{1}}$

[b~ b'] [F b': Bool

 $\overrightarrow{\Gamma} \vdash (if(b) \not t_1 else \not t_2) : \overrightarrow{\tau} \quad (if(false) t_1 else t_2) \rightsquigarrow t_2$

and the result, by type rule for **if**, has type τ (preservation). If b' is not constant, the assumption of the rule

$$b \leadsto b'$$

(if (b)
$$t_1$$
 else t_2) \rightsquigarrow (if (b') t_1 else t_2)

applies, so t also makes progress. By preservation IH, b' also has type Bool, so the entire expression can be typed as τ re-using the type derivations for t_1 and t_2 .

Progress and preservation - user defined functions

Following the cases of operational semantics, either all arguments of a function have been evaluated to a constant, or some are not yet constant.

If they are not all constants, the case is as for the condition of **if**, and we establish progress and preservation analogously.

Otherwise rule

$$f(c_1,\ldots,c_n) \leadsto t_f[x_1 := c_1,\ldots,x_n := c_n]$$

progress V

(*)

applies, so progress is ensured. For preservation, we need to show $f(c_1, ..., c_N) : \tilde{c}_p$ $\Gamma \vdash t_f[x_1 := c_1, ..., x_n := c_n] : \tau$

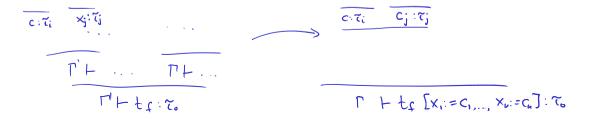
where $e(f) = ((x_1, ..., x_n), (\tau_1, ..., \tau_n), t_f, \tau_0)$ and t_f is the body of f. According to type rules $\tau = \tau_0$ and $\Gamma \vdash c_i : \tau_i$.

Progress and preservation - substitution and types

Function f definition type checks, so $\Gamma' \vdash (t_f)$: τ_0 where $\Gamma' = \Gamma \oplus \{(x_1, \tau_1), \dots, (x_n, \tau_n)\}$. Consider the type derivation tree for t_f and replace each use of $\Gamma' \vdash x_i : \tau_i$ with $\Gamma \vdash c_i : \tau_i$. The result is a type derivation for (*):

$$\Gamma \vdash t_f[x_1 := c_1, \dots, x_n := c_n] : \tau \tag{(*)}$$

Therefore, the preservation holds in this case as well.



Progress and preservation - substitution and types

Function f definition type checks, so $\Gamma' \vdash t_f : \tau_0$ where $\Gamma' = \Gamma \oplus \{(x_1, \tau_1), \dots, (x_n, \tau_n)\}$. Consider the type derivation tree for t_f and replace each use of $\Gamma' \vdash x_i : \tau_i$ with $\Gamma \vdash c_i : \tau_i$. The result is a type derivation for (*):

$$\Gamma \vdash t_f[x_1 := c_1, \dots, x_n := c_n] : \tau \tag{(*)}$$

Therefore, the preservation holds in this case as well.

Exercise: prove the above step that replacing variables with constants of the same type transforms term that has type derivation with type τ into a term that again has a derivation with type τ . Is there a more general statement?