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:

$$\textbf{if } (5)\ 3 \textbf{ else } 7$$

the expression 5 cannot be evaluated further and is a constant, but there are no rules for when condition of **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

$\Gamma \vdash t : \tau$

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:

$$\Gamma_0 \vdash t : \tau$$

$\Gamma_0 \oplus \Gamma_1 = \{(x,\tau) \mid (x,\tau) \in \Gamma_1 \vee ((x,\tau) \in \Gamma_0 \wedge \neg \exists \tau'.(x,\tau') \in \Gamma_1)\}$

and
(2) each function body type checks:

$$\Gamma_0 \overset{\cup}{\oplus} \{(x_1, \tau_1), \ldots, (x_n, \tau_n)\} \vdash \underbrace{t_f}_{\text{body}} : \underbrace{\tau_0}_{\text{result type}}$$

for each $(f, (x_1, \ldots, x_n), (\tau_1, \ldots, \tau_n), t_f, \tau_0) \in e$

Type Checking Rules

$$\Gamma \vdash t : \tau$$

$$\frac{(x, \tau) \in \Gamma}{\Gamma \vdash x : \tau}$$

$$\tau \in \{Bool, Int\}$$

$$\frac{(f, \tau_1 \times \tau_2 \times \ldots \times \tau_n \to \tau_0) \in \Gamma}{\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \to \tau_0}$$

$$\frac{\Gamma \vdash b : Bool, \quad \Gamma \vdash t_1 : \tau, \quad \Gamma \vdash t_2 : \tau}{\Gamma \vdash (\textbf{if } (b) \ t_1 \textbf{ else } t_2) : \tau}$$

$$\frac{\Gamma \vdash f : \tau_1 \times \cdots \times \tau_n \to \tau_0, \quad \Gamma \vdash t_1 : \tau_1, \ \ldots, \ \Gamma \vdash t_n : \tau_n}{\Gamma \vdash f(t_1, \ldots, t_n) : \tau_0}$$

We treat primitives like applications of functions e.g.

$+ : Int \times Int \to Int$

$\leq : Int \times Int \to Bool$

$\&\& : Bool \times Bool \to Bool$

$$(+, Int \times Int \to Int) \in \Gamma_0$$

$$\frac{}{\Gamma \vdash + : Int \times Int \to Int}$$

$$\frac{\Gamma \vdash t_1 : Int \quad \Gamma \vdash t_2 : Int}{\Gamma \vdash t_1 + t_2 : Int}$$

Soundness through progress and preservation

$\Gamma \vdash t : \tau$

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

  G

  $$\Gamma \vdash t : \tau$$

  then either t is a constant (execution is done) or there exists t' such that $t \rightsquigarrow t'$

- S

  preservation: if a program type checks and makes one $\rightsquigarrow$ step,
  then the result again type checks
  in our simple system: it type checks *and has the same type*: if

  $$\Gamma \vdash t : \tau$$

  and $t \rightsquigarrow t'$ then

  $$\Gamma \vdash t' : \tau$$

$t_0 : \tau \overset{G,S}{\rightsquigarrow} t_1 : \tau \overset{G,S}{\rightsquigarrow} t_2 : \tau \rightsquigarrow c : \tau$

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):

$\rightsquigarrow t' : \tau$

$t \equiv \text{if}(b)\ t_1 \text{ else } t_2$

$$\frac{\Gamma \vdash b : \text{Bool} \quad \Gamma \vdash b' : \text{Bool} \quad \Gamma \vdash t_1 : \tau \quad \Gamma \vdash t_2 : \tau}{\Gamma \vdash (\text{if}\ (b')\ t_1 \text{ else } t_2) : \tau}$$

$$\overline{(\textbf{if}\ (\mathit{true})\ t_1\ \textbf{else}\ t_2) \rightsquigarrow t_1}$$

$$\overline{(\textbf{if}\ (\mathit{false})\ t_1\ \textbf{else}\ t_2) \rightsquigarrow t_2}$$

$\boxed{b \rightsquigarrow b'}$

$\Gamma \vdash b' : \text{Bool}$

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

$$\frac{b \rightsquigarrow b'}{(\textbf{if}\ (b)\ t_1\ \textbf{else}\ t_2) \rightsquigarrow \underline{(\textbf{if}\ (b')\ t_1\ \textbf{else}\ t_2)}}$$

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

$$\frac{}{f(c_1,\ldots,c_n) \rightsquigarrow t_f[x_1 := c_1,\ldots,x_n := c_n]}$$

progress ✓

applies, so progress is ensured. For preservation, we need to show

$f(c_1,\ldots,c_n) : \tau_0$

$$\Gamma \vdash t_f[x_1 := c_1,\ldots,x_n := c_n] : \tau \qquad (*)$$

where $e(f) = ((x_1,\ldots,x_n),(\tau_1,\ldots,\tau_n),t_f,\tau_0)$ 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 $\Gamma' \vdash t_f : \tau_0$ where $\Gamma' = \Gamma \oplus \{(x_1, \tau_1), \ldots, (x_n, \tau_n)\}$. 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 $(*)$:

$$\Gamma \vdash t_f[x_1 := c_1, \ldots, x_n := c_n] : \tau \qquad (*)$$

Therefore, the preservation holds in this case as well.

$$\dfrac{\dfrac{\dfrac{}{c : \tau_i} \quad \dfrac{}{x_j : \tau_j} \;\cdots}{\Gamma' \vdash \ldots} \qquad \dfrac{\cdots}{\Gamma \vdash \ldots}}{\Gamma' \vdash t_f : \tau_0} \longrightarrow \dfrac{\dfrac{\dfrac{}{c : \tau_i} \quad \dfrac{}{c_j : \tau_j}}{}}{\Gamma \vdash t_f [x_1 := c_1, \ldots, x_n := c_n] : \tau_0}$$

Progress and preservation - substitution and types

Function f definition type checks, so $\Gamma' \vdash t_f : \tau_0$ where $\Gamma' = \Gamma \oplus \{(x_1, \tau_1), \ldots, (x_n, \tau_n)\}$. 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 $(*)$:

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

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?