

TYPE SAFETY: PROGRESS & PRESERVATION

Alex Kavvos

Reading: PFPL, §6.

The **statics** and the **dynamics** play well together: we show that **well-typed programs do not go wrong**.

Theorem 1 (Type safety).

1. (Preservation) If $\vdash e : \tau$ and $e \mapsto e'$ then $\vdash e' : \tau$.
2. (Progress) If $\vdash e : \tau$ then either e val or $e \mapsto e'$ for some e' .

Therefore **closed, well-typed terms** behave well under reduction:

1. their type is preserved under evaluation, and
2. if they're not done evaluating, transitions will continue to take place.

1 Preservation

Preservation is the statement that types are preserved under evaluation. This is a central **safety** property of type systems: it shows that a step-by-step computation preserves the kind of value that is being computed.

Theorem 2 (Preservation). If $\vdash e : \tau$ and $e \mapsto e'$ then $\vdash e' : \tau$.

Proof. By induction on the derivation of $e \mapsto e'$. We show the most difficult case, namely that of D-LET.

Case(D-LET). Suppose that the reduction $e \mapsto e'$ is of the form

$$\frac{}{\text{let}(e_1; x. e_2) \mapsto e_2[e_1/x]} \text{D-LET}$$

We know that $\vdash \text{let}(e_1; x. e_2) : \tau$. By **inversion** there must exist σ such that $\vdash e_1 : \sigma$ and $x : \sigma \vdash e_2 : \tau$. By the **substitution lemma** (Lecture 4) we obtain $\vdash e_2[e_1/x] : \tau$, which is what we wanted to prove.

□

2 Progress

Progress is the statement that if a well-typed program is not done computing (is a value), then there is a step of computation it may take. It is a central **liveness** property of type systems: it shows that a computation will continue to evolve until it produces a useful result (if ever!).

First, we need to characterise the values of each type. The following lemma follows ‘by inspection.’

Lemma 3 (Canonical forms). Suppose e val.

1. If $\vdash e : \text{Num}$ then $e = \text{num}[n]$ for some $n \in \mathbb{N}$.
2. If $\vdash e : \text{Str}$ then $e = \text{str}[s]$ for some $s \in \Sigma^*$.

Theorem 4 (Progress). If $\vdash e : \tau$ then either e val or $e \mapsto e'$ for some e' .

Proof. By induction on the derivation of $\vdash e : \tau$. We only show the case for PLUS.

Case(PLUS). Suppose that the derivation is of the form

$$\frac{\frac{\vdots}{\vdash e_1 : \text{Num}} \quad \frac{\vdots}{\vdash e_2 : \text{Num}}}{\vdash \text{plus}(e_1; e_2) : \text{Num}} \text{PLUS}$$

e_1 is a closed, well-typed term with a ‘smaller’ derivation, so the **induction hypothesis** applies to it. Hence, either e_1 val, or there exists e'_1 such that $e_1 \mapsto e'_1$. We consider each case separately.

- Suppose e_1 val. We then apply the **induction hypothesis** to e_2 , and obtain the same two cases for e_2 .
 - Suppose e_2 val. Then, by the canonical forms lemma (Lemma 3) we have that $e_1 = \text{num}[n_1]$ and $e_2 = \text{num}[n_2]$ for some $n_1, n_2 \in \mathbb{N}$. Then the reduction rule D-PLUS applies to the term $\text{plus}(e_1; e_2) = \text{plus}(\text{num}[n_1]; \text{num}[n_2])$, and we have $\text{plus}(\text{num}[n_1]; \text{num}[n_2]) \mapsto \text{num}[n_1 + n_2]$.
 - Suppose there exists e'_2 so that $e_2 \mapsto e'_2$. Then we can construct a derivation

$$\frac{e_1 \text{ val} \quad \frac{\vdots}{e_2 \mapsto e'_2}}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e_1; e'_2)} \text{D-PLUS-2}$$

so that $\text{plus}(e_1; e_2)$ in fact steps to $\text{plus}(e_1; e'_2)$ according to the dynamics.

- Suppose there exists e'_1 so that $e_1 \mapsto e'_1$. Then we can construct a derivation

$$\frac{\frac{\vdots}{e_1 \mapsto e'_1}}{\text{plus}(e_1; e_2) \mapsto \text{plus}(e'_1; e_2)} \text{D-PLUS-1}$$

so that $\text{plus}(e_1; e_2)$ in fact steps to $\text{plus}(e'_1; e_2)$ according to the dynamics.

In each case of this exhaustive analysis, there always exists a term to which $\text{plus}(e_1; e_2)$ steps (if well-typed).

□