Type Safety

CS 4301/6371: Advanced Programming Languages

Kevin W. Hamlen

March 21, 2024

Objective

Recall: Important characteristics of a static semantics:

- Catches all (or most) stuck states before runtime (type-safety)
- Deterministic (otherwise can't implement it!)
- Try not to classify programmer-desired functionalities as type-errors

Today: Formally define and prove the first one (type-safety).

Type-safety

Definition (well-typed)

A command c is *well-typed* if there exists Γ' such that $\perp \vdash c : \Gamma'$ is derivable.

Theorem (type-safety)

If c is well-typed and $\langle c, \perp \rangle \to_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Recall that we previously defined two kinds of states:

- Final states: $\langle \texttt{skip}, \sigma \rangle$, $\langle n, \sigma \rangle$, $\langle \texttt{true}, \sigma \rangle$, $\langle \texttt{false}, \sigma \rangle$
- Stuck states: Non-final state from which no step is derivable

How to prove this? Recall that we have no judgments for $\to_n$ so we'd like to remove that with a trivial $\mathbb{N}$-induction like we did in the proof of semantic equivalence.

Attempt #1

Theorem (type-safety)

If c is well-typed and $\langle c, \bot \rangle \rightarrow_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Proof

Assume c is well-typed. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \langle c, \bot \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$. ...

Q: If we prove this, then does it prove the theorem?

Attempt #1

Theorem (type-safety)

If c is well-typed and $\langle c, \bot\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \langle c, \bot\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$. ...

Q: If we prove this, then does it prove the theorem?
A: No! Fails to prove that $\langle c_2, \sigma_2\rangle$ is not a stuck state, since:

- σ_2 might not be $\bot$, and
- c_2 might not be well-typed

How to fix?

Attempt #2

Theorem (type-safety)

If c is well-typed and $\langle c, \perp \rangle \rightarrow_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Proof

Assume c is well-typed and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$ where c_2 is well-typed. ...

Q: If we prove this, then does it prove the theorem?

Attempt #2

Theorem (type-safety)

If c is well-typed and $\langle c, \bot \rangle \rightarrow_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Proof

Assume c is well-typed and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$ where c_2 is well-typed. ...

Q: If we prove this, then does it prove the theorem?
A: Yes, but there's a bigger problem: It's not true!

Attempt #2

Theorem (type-safety)

If c is well-typed and $\langle c, \bot\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed. ...

Q: If we prove this, then does it prove the theorem?
A: Yes, but there's a bigger problem: It's not true!
Example: $\langle \texttt{int x;x := } 2, \bot\rangle \rightarrow_1 ?$

Attempt #2

Theorem (type-safety)

If c is well-typed and $\langle c, \bot\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed. ...

Q: If we prove this, then does it prove the theorem?
A: Yes, but there's a bigger problem: It's not true!
Example: $\langle \texttt{int x;x := }2, \bot\rangle \rightarrow_1 \langle \texttt{skip;x := }2, \bot\rangle$

Generalizing Well-typedness

Solution: Generalize the definition of well-typedness.

Definition (well-typed): Command c is well-typed in context Γ if there exists Γ' such that $\Gamma \vdash c : \Gamma'$ is derivable.

Attempt #3

Theorem (type-safety)

If c is well-typed and $\langle c, \perp \rangle \rightarrow_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$ where c_2 is well-typed in Γ_2. ...

Q: Is this one sufficient (and true)?

Attempt #3

Theorem (type-safety)

If c is well-typed and $\langle c, \perp\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed in Γ_2. ...

Q: Is this one sufficient (and true)?
A: Still not true, but for a different reason. Can you spot the problem?

Attempt #3

Theorem (type-safety)

If c is well-typed and $\langle c, \perp\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed in Γ_2. ...

Q: Is this one sufficient (and true)?
A: Still not true, but for a different reason. Can you spot the problem?

Example: Suppose $c = (\texttt{x} := \texttt{x} + 2)$ and $\Gamma = \{(\texttt{x}, (\mathit{int}, T))\}$ and $\sigma = \{(\texttt{x}, T)\}$. Note that $\Gamma \vdash c : \Gamma$ so it's well-typed. But $\langle c, \sigma\rangle \rightarrow_1$?

Attempt #3

Theorem (type-safety)

If c is well-typed and $\langle c, \bot \rangle \rightarrow_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$ where c_2 is well-typed in Γ_2. ...

Q: Is this one sufficient (and true)?
A: Still not true, but for a different reason. Can you spot the problem?

Example: Suppose $c = (\texttt{x} := \texttt{x} + 2)$ and $\Gamma = \{(\texttt{x}, (int, T))\}$ and $\sigma = \{(\texttt{x}, T)\}$. Note that $\Gamma \vdash c : \Gamma$ so it's well-typed. But $\langle c, \sigma \rangle \rightarrow_1 \langle \texttt{x} := \texttt{true} + 2, \sigma \rangle$.

Solution: Need to somehow stipulate that Γ and σ "match".

Modeling Relation

Definition (models): A typing context Γ *models* a store σ (written $\Gamma \models \sigma$) if for all $v \in \Gamma^{\leftarrow}$,

- if $\Gamma(v) = (int, T)$ then $\sigma(v) \in \mathbb{Z}$, and
- if $\Gamma(v) = (bool, T)$ then $\sigma(v) \in \{T, F\}$.

(Note that if $\Gamma(v) = (\tau, F)$ or $v \notin \Gamma^{\leftarrow}$ then we impose no obligation on σ.)

Attempt #4

Theorem (type-safety)

If c is well-typed and $\langle c, \perp\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given such that $\Gamma \models \sigma$. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed in Γ_2 and $\Gamma_2 \models \sigma_2$. ...

Q: Is this one sufficient (and true)? *(please, please, please, ...)*

Attempt #4

Theorem (type-safety)

If c is well-typed and $\langle c, \bot\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given such that $\Gamma \models \sigma$. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed in Γ_2 and $\Gamma_2 \models \sigma_2$. ...

Q: Is this one sufficient (and true)? *(please, please, please, ...)*
A: For some languages this would be enough, but our language has one more feature that makes this false: local scopes.

Example: $\langle \texttt{if true then int x else skip;bool x}, \sigma\rangle \rightarrow_1 ?$

Attempt #4

Theorem (type-safety)

If c is well-typed and $\langle c, \bot \rangle \rightarrow_n \langle c', \sigma' \rangle$ (where $n \geq 0$), then $\langle c', \sigma' \rangle$ is not a stuck state.

Proof

Assume c is well-typed in Γ, and let $\sigma \in \Sigma$ be given such that $\Gamma \models \sigma$. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \Gamma_2, \langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$ where c_2 is well-typed in Γ_2 and $\Gamma_2 \models \sigma_2$. ...

Q: Is this one sufficient (and true)? *(please, please, please, ...)*
A: For some simple languages this would be enough, but our language has one more feature that makes this false:

Example: $\langle \texttt{if true then int x else skip;bool x}, \sigma \rangle \rightarrow_1 \langle \texttt{int x;bool x}, \sigma \rangle$

Intermediate States

General problem: Most real language have small-step semantics that pass through *intermediate states* that are invalid at the original source level.

Example from Java: $\langle obj.field, \sigma\rangle \rightarrow_1 \langle \boxed{\text{value of } obj}.field, \sigma\rangle$

In SIMPL, our intermediate states are local scopes introduced by `if` and `while` commands.

Solution: Extend the static semantics to include extra rules that type-check intermediate states. Since programmers are not allowed to write such states (syntax error), the new rules have no effect on them.

Adding Explicit Scoping

New syntax for these intermediate states:

$$c ::= \cdots \mid \texttt{\{}c_1\texttt{\}}$$

They do nothing at runtime:

$$\frac{\langle c, \sigma\rangle \to_1 \langle c', \sigma'\rangle}{\langle \texttt{\{}c\texttt{\}}, \sigma\rangle \to_1 \langle \texttt{\{}c'\texttt{\}}, \sigma'\rangle}$$

$$\frac{}{\langle \texttt{\{skip\}}, \sigma\rangle \to_1 \langle \texttt{skip}, \sigma\rangle}$$

But we can introduce them when reducing conditionals and loops:

$$\frac{}{\langle \texttt{if true then } c_1 \texttt{ else } c_2, \sigma\rangle \to_1 \langle \texttt{\{}c_1\texttt{\}}, \sigma\rangle}$$

$$\frac{}{\langle \texttt{if false then } c_1 \texttt{ else } c_2, \sigma\rangle \to_1 \langle \texttt{\{}c_2\texttt{\}}, \sigma\rangle}$$

$$\frac{}{\langle \texttt{while } e \texttt{ do } c, \sigma\rangle \to_1 \langle \texttt{if } e \texttt{ then } (\texttt{\{}c\texttt{\};while } e \texttt{ do } c) \texttt{ else skip}, \sigma\rangle}$$

Typing Stacks

Scopes can be nested, so we now need a stack of typing contexts:

$$\frac{\Gamma_2, \ldots \vdash c : \Gamma'}{\Gamma_1, \Gamma_2, \ldots \vdash \{c\} : \Gamma_1}$$

Definition (typing context stacks): A typing context stack $\overrightarrow{\Gamma}$ is a non-empty, finite sequence $\Gamma_1, \ldots, \Gamma_n$ of typing contexts satisfying $\Gamma_n \preceq \cdots \preceq \Gamma_1$.

Definition (subtype): Context Γ_1 is a *subtype* of context Γ_2 (written $\Gamma_1 \preceq \Gamma_2$) if for all $(v, (\tau, p)) \in \Gamma_2$, there exists $q \in \{T, F\}$ such that

- $\Gamma_1(v) = (\tau, q)$ and
- $p \Rightarrow q$.

Intuition: Γ_1 is the outermost context, and outer contexts are "more restrictive" ($\succeq$) than inner ones (fewer declared/initialized variables).

See online notes for complete static semantics.

Attempt #5

Theorem (type-safety)

If c is well-typed and $\langle c, \bot\rangle \rightarrow_n \langle c', \sigma'\rangle$ (where $n \geq 0$), then $\langle c', \sigma'\rangle$ is not a stuck state.

Proof

Assume c is well-typed in $\overrightarrow{\Gamma}$, and let $\sigma \in \Sigma$ be given such that $\overrightarrow{\Gamma} \models \sigma$. We will prove that either $c = \texttt{skip}$ or $\exists c_2, \sigma_2, \overrightarrow{\Gamma_2}, \langle c, \sigma\rangle \rightarrow_1 \langle c_2, \sigma_2\rangle$ where c_2 is well-typed in $\overrightarrow{\Gamma_2}$ and $\overrightarrow{\Gamma_2} \models \sigma_2$. ...

This (finally!) works!

Progress & Preservation

Easier to break it up into four lemmas:

Lemma 1 (Progress of expressions)

If $\Gamma \vdash e : \tau$ and $\Gamma \models \sigma$ then either $\langle e, \sigma \rangle$ is final or $\exists e', \sigma', \langle e, \sigma \rangle \rightarrow_1 \langle e', \sigma' \rangle$.

Lemma 2 (Progress of commands)

If $\overrightarrow{\Gamma} \vdash e : \Gamma'$ and $\overrightarrow{\Gamma} \models \sigma$ then either $\langle c, \sigma \rangle$ is final or $\exists c', \sigma', \langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$.

Lemma 3 (Preservation of expressions)

If $\Gamma \vdash e : \tau$ and $\Gamma \models \sigma$ and $\langle e, \sigma \rangle \rightarrow_1 \langle e', \sigma' \rangle$, then $\Gamma \vdash e' : \tau$ and $\Gamma \models \sigma'$.

Lemma 4 (Preservation of commands)

If $\overrightarrow{\Gamma} \vdash c : \Gamma'$ and $\overrightarrow{\Gamma} \models \sigma$ and $\langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$, then $\exists \overrightarrow{\Gamma_2}, \overrightarrow{\Gamma_2} \vdash c_2 : \Gamma'$ and $\overrightarrow{\Gamma_2} \models \sigma_2$ and $\Gamma_2 \preceq \Gamma$.

Preservation is also called *subject reduction*.

Proving Progress & Preservation

Practice Problem: See if you can prove any cases of these lemmas.

Suggested approaches:

- Prove progress lemmas by structural induction on derivation $\mathcal{D}$ of
  - $\Gamma \vdash e : \tau$ (for expressions), or
  - $\overrightarrow{\Gamma} \vdash c : \Gamma'$ (for commands).
- Prove preservation lemmas by structural induction on derivation $\mathcal{D}$ of
  - $\langle e, \sigma \rangle \rightarrow_1 \langle e', \sigma' \rangle$ (for expressions), or
  - $\langle c, \sigma \rangle \rightarrow_1 \langle c_2, \sigma_2 \rangle$ (for commands).