asm.js

Dave Herman, Luke Wagner, and Alon Zakai

November 14, 2012

1 Abstract syntax

$$\begin{array}{rcl} b, e, f, g, x, y, z & \in & \textit{Identifier} \\ \texttt{arguments}, \texttt{eval} & \notin & \textit{Identifier} \end{array}$$

$$\begin{array}{rcl} P & ::= & \texttt{function}\ [g]([e[,b]])\ \texttt{\{ "use asm";}\ \overline{imp_x}\ \overline{fn_f}\ \overline{\texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}}\ exp\ \texttt{\}} \\ imp_x & ::= & \texttt{var}\ x\ \texttt{=}\ e.y\texttt{;} \\ & | & \texttt{var}\ x\ \texttt{=}\ \texttt{new}\ e.y(b)\texttt{;} \\ exp & ::= & \texttt{return}\ f\texttt{;} \\ & | & \texttt{return \{}\ \overline{x\texttt{:}f}\ \texttt{\};} \\ fn_f & ::= & \texttt{function}\ f(\overline{x})\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \overline{\texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}}\ ss\ \texttt{\}} \end{array}$$

$$\begin{array}{rcl} s & ::= & \texttt{\{}\ ss\ \texttt{\}} \\ & | & e\texttt{;} \\ & | & \texttt{;} \\ & | & \texttt{if}\ (e)\ s \\ & | & \texttt{if}\ (e)\ s\ \texttt{else}\ s \\ & | & \texttt{return}\ [e]\texttt{;} \\ & | & \texttt{while}\ (e)\ s \\ & | & \texttt{do}\ s\ \texttt{while}\ (e)\texttt{;} \\ & | & \texttt{for}\ ([e]\texttt{;}\ [e]\texttt{;}\ [e])\ s \\ & | & \texttt{switch}\ (e)\ \texttt{\{}\ \overline{c}\ [d]\ \texttt{\}} \\ & | & \texttt{break}\ [lab]\texttt{;} \\ & | & \texttt{continue}\ [lab]\texttt{;} \\ & | & lab\texttt{:}\ s \end{array}$$

$$ss ::= \overline{s}$$

$$\begin{array}{rcl} c & ::= & \texttt{case}\ v\texttt{:}\ ss \\ d & ::= & \texttt{default:}\ ss \\ cd & ::= & c \mid d \end{array}$$

$$\kappa_x ::= \texttt{\~{}\~{}}x \mid \texttt{+}x \mid x\texttt{|0} \mid x\texttt{>>>0}$$

$$
\begin{array}{rcl}
v & ::= & r \mid n \\
\\
e & ::= & v \\
 & \mid & \mathit{lval} \\
 & \mid & \mathit{lval}\ \texttt{=}\ e \\
 & \mid & f\texttt{(}\overline{e}\texttt{)} \\
 & \mid & \mathit{unop}\ e \\
 & \mid & e\ \mathit{binop}\ e \\
 & \mid & e\ \texttt{?}\ e\ \texttt{:}\ e \\
 & \mid & \texttt{(}\overline{e}\texttt{)} \\
\\
\mathit{unop} & ::= & \texttt{+} \mid \texttt{~} \mid \texttt{!} \\
\\
\mathit{binop} & ::= & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \\
 & \mid & \texttt{|} \mid \texttt{\&} \mid \texttt{\^{}} \mid \texttt{<<} \mid \texttt{>>} \mid \texttt{>>>} \\
 & \mid & \texttt{<} \mid \texttt{<=} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{!=} \mid \texttt{==} \\
\mathit{lval} & ::= & x \mid x\texttt{[(}e\ \texttt{\&}\ m\texttt{)}\ \texttt{>>}\ n\texttt{]}
\end{array}
$$

2 Type rules

$$
\begin{array}{rcl}
\sigma, \tau & ::= & \texttt{bit} \mid \texttt{double} \mid \texttt{int} \mid \texttt{signed} \mid \texttt{unsigned} \mid \texttt{boolish} \mid \texttt{intish} \mid \texttt{void} \mid \texttt{unknown} \\
\rho & ::= & \tau \mid \texttt{view}^n_\tau \mid \texttt{imul} \mid \texttt{function} \mid (\overline{\sigma}) \to \tau \\
\omega & ::= & ((\overline{\sigma}) \to \tau) \wedge \ldots \wedge ((\overline{\sigma'}) \to \tau')
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{type}(\texttt{\~{}\~{}}X) & = & \texttt{int} \\
\mathit{type}(\texttt{+}X) & = & \texttt{double} \\
\mathit{type}(n) & = & \texttt{int} \\
\mathit{type}(r) & = & \texttt{double} \\
\mathit{type}(X\texttt{|0}) & = & \texttt{signed} \\
\mathit{type}(X\texttt{>>>0}) & = & \texttt{unsigned}
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{constant} & <: & \texttt{signed}, \texttt{unsigned} \\
\texttt{signed}, \texttt{unsigned} & <: & \texttt{int}, \texttt{extern} \\
\texttt{bit}, \texttt{int} & <: & \texttt{boolish} \\
\texttt{double} & <: & \texttt{extern} \\
\texttt{unknown}, \texttt{int} & <: & \texttt{intish}
\end{array}
$$

$$
\begin{array}{rcl}
M(\texttt{imul}) & : & \texttt{imul} \\
M(\texttt{ceil}), M(\texttt{sin}), M(\texttt{cos}) & : & (\texttt{double}) \to \texttt{double}
\end{array}
$$

$$
\begin{array}{rcl}
A(\texttt{Uint8Array}), A(\texttt{Int8Array}) & = & \texttt{view}^{8}_{\texttt{int}} \\
A(\texttt{Uint16Array}), A(\texttt{Int16Array}) & = & \texttt{view}^{16}_{\texttt{int}} \\
A(\texttt{Uint32Array}), A(\texttt{Int32Array}) & = & \texttt{view}^{32}_{\texttt{int}} \\
A(\texttt{Float32Array}) & = & \texttt{view}^{32}_{\texttt{double}} \\
A(\texttt{Float64Array}) & = & \texttt{view}^{64}_{\texttt{double}}
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{+}, \texttt{-} & : & (\texttt{double}, \texttt{double}) \to \texttt{double} \\
 & & \wedge\ (\texttt{int}, \texttt{int}) \to \texttt{intish} \\
\texttt{*} & : & (\texttt{double}, \texttt{double}) \to \texttt{double} \\
\texttt{/}, \texttt{\%} & : & (\texttt{double}, \texttt{double}) \to \texttt{double} \\
 & & \wedge\ (\texttt{signed}, \texttt{signed}) \to \texttt{intish} \\
 & & \wedge\ (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{intish} \\
 & & \\
\texttt{|}, \texttt{\&}, \texttt{\^{}}, \texttt{<<}, \texttt{>>} & : & (\texttt{intish}, \texttt{intish}) \to \texttt{signed} \\
\texttt{>>>} & : & (\texttt{intish}, \texttt{intish}) \to \texttt{unsigned} \\
 & & \\
\texttt{<}, \texttt{<=}, \texttt{>}, \texttt{>=}, \texttt{==}, \texttt{!=} & : & (\texttt{signed}, \texttt{signed}) \to \texttt{bit} \\
 & & \wedge\ (\texttt{unsigned}, \texttt{unsigned}) \to \texttt{bit} \\
 & & \wedge\ (\texttt{double}, \texttt{double}) \to \texttt{bit} \\
 & & \\
\texttt{+} & : & (\texttt{intish}) \to \texttt{double} \\
\texttt{\~{}} & : & (\texttt{intish}) \to \texttt{signed} \\
\texttt{!} & : & (\texttt{boolish}) \to \texttt{bit}
\end{array}
$$

$$
\begin{array}{rcl}
\Delta & ::= & \{\overline{x : \rho}\} \\
\Gamma & ::= & \{\overline{x : \tau}\}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{breaks}(\overline{s}) & = & \bigcup_i \mathit{breaks}(s_i) \\
\mathit{breaks}(\texttt{\{}\ ss\ \texttt{\}}) & = & \mathit{breaks}(ss) \\
\mathit{breaks}(\texttt{if (}e\texttt{)}\ s) & = & \mathit{breaks}(s) \\
\mathit{breaks}(\texttt{if (}e\texttt{)}\ s_1\ \texttt{else}\ s_2) & = & \mathit{breaks}(s_1) \cup \mathit{breaks}(s_2) \\
\mathit{breaks}(\texttt{while (}e\texttt{)}\ s) & = & \mathit{breaks}(s) - \{\epsilon\} \\
\mathit{breaks}(\texttt{do}\ s\ \texttt{while (}e\texttt{);}) & = & \mathit{breaks}(s) - \{\epsilon\} \\
\mathit{breaks}(\texttt{for (}[e_1]\texttt{; }[e_2]\texttt{; }[e_3]\texttt{)}\ s) & = & \mathit{breaks}(s) - \{\epsilon\} \\
\mathit{breaks}(\texttt{break;}) & = & \{\epsilon\} \\
\mathit{breaks}(\texttt{break}\ lab\texttt{;}) & = & \{lab\} \\
\mathit{breaks}(lab\texttt{:}\ s) & = & \mathit{breaks}(s) - \{lab\} \\
\mathit{breaks}(\texttt{switch (}e\texttt{) \{}\ \overline{cd}\ \texttt{\}}) & = & \bigcup_i \mathit{breaks}(cd_i) - \{\epsilon\} \\
\mathit{breaks}(s)\ (\text{otherwise}) & = & \emptyset \\
\mathit{breaks}(\texttt{case}\ v\texttt{:}\ ss) & = & \mathit{breaks}(ss) \\
\mathit{breaks}(\texttt{default:}\ ss) & = & \mathit{breaks}(ss)
\end{array}
$$

Program checking $\boxed{\vdash P\ \textbf{ok}}$

[T-Program]

$$\frac{\overline{x}, \overline{y}, \overline{f}, [g], [e], [b] \text{ distinct} \qquad \forall y.\Delta(y) = \mathit{type}(v) \qquad \forall i.[e]; [b]; \Delta \vdash \mathit{imp}_x\ \textbf{ok} \qquad \forall i.\Delta \vdash \mathit{fn}_f\ \textbf{ok} \qquad \forall i.\Delta \vdash \mathit{exp}\ \textbf{ok}}{\vdash \texttt{function}\ [g]\texttt{(}[e[, b]]\texttt{)}\ \texttt{\{ "use asm";}\ \overline{\mathit{imp}_x}\ \overline{\mathit{fn}_f}\ \overline{\texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}}\ \mathit{exp}\ \texttt{\}}\ \textbf{ok}}$$

Import checking $\boxed{[e]; [b]; \Delta \vdash \mathit{imp}\ \textbf{ok}}$

[T-ImportStd]

$$\frac{\Delta(x) = M(y)}{e; [b]; \Delta \vdash \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;}\ \textbf{ok}}$$

[T-ImportFFI]

$$\frac{y \notin \mathit{dom}(M), \mathit{dom}(A) \qquad \Delta(x) = \texttt{function}}{e; [b]; \Delta \vdash \texttt{var}\ x\ \texttt{=}\ e\texttt{.}y\texttt{;}\ \textbf{ok}}$$

[T-View]

$$\frac{\Delta(x) = \texttt{view}^n_{A(y)}}{e; b; \Delta \vdash \texttt{var}\ x\ \texttt{=}\ \texttt{new}\ e\texttt{.}y\texttt{(}b\texttt{);}\ \textbf{ok}}$$

Function checking $\boxed{\Delta \vdash \mathit{fn}\ \textbf{ok}}$

[T-Function]

$$\frac{\overline{x}, \overline{y}\ \text{distinct} \qquad \Delta(f) = (\overline{\sigma}) \to \tau \qquad \overline{\sigma} = \overline{\mathit{type}(\kappa_x)} \qquad \Delta; \{\overline{x : \sigma}, \overline{y : \mathit{type}(v)}\}; f \vdash \mathit{ss}\ \textbf{ok} \qquad \tau \neq \texttt{void} \Rightarrow \mathit{returns}(\mathit{ss})}{\Delta \vdash \texttt{function}\ f\texttt{(}\overline{x}\texttt{)}\ \texttt{\{}\ \overline{x\ \texttt{=}\ \kappa_x\texttt{;}}\ \overline{\texttt{var}\ \overline{y\ \texttt{=}\ v}\texttt{;}}\ \mathit{ss}\ \texttt{\}}\ \textbf{ok}}$$

Export checking $\boxed{\Delta \vdash \mathit{exp}\ \textbf{ok}}$

[T-Singleton]

$$\frac{\Delta(f) = (\overline{\sigma}) \to \tau \qquad \tau <: \texttt{extern}}{\Delta \vdash \texttt{return}\ f\texttt{;}\ \textbf{ok}}$$

[T-Module]

$$\frac{\forall f.\Delta(f) = (\overline{\sigma}) \to \tau \wedge \tau <: \texttt{extern}}{\Delta \vdash \texttt{return \{}\ \overline{x\texttt{:}f}\ \texttt{\};}\ \textbf{ok}}$$

$\mathit{returns}(\overline{s})$
  if $\mathit{returns}(s_m) \wedge \forall i < m.\mathit{breaks}(s_m) = \emptyset$
  for some m
$\mathit{returns}(\texttt{\{}\ \mathit{ss}\ \texttt{\}})$
  if $\mathit{returns}(\mathit{ss})$
$\mathit{returns}(\texttt{if (}e\texttt{)}\ s_1\ \texttt{else}\ s_2)$
  if $\mathit{returns}(s_1) \wedge \mathit{returns}(s_2)$
$\mathit{returns}(\texttt{do}\ s\ \texttt{while (}e\texttt{);})$
  if $\mathit{returns}(s)$
$\mathit{returns}(\texttt{switch (}e\texttt{) \{}\ \overline{cd}\ \texttt{\}})$
  if $\mathit{returns}(cd_n) \wedge \forall i.\mathit{breaks}(cd_i) = \emptyset$
$\mathit{returns}(\texttt{case}\ v\texttt{:}\ \mathit{ss})$
  if $\mathit{returns}(\mathit{ss})$
$\mathit{returns}(\texttt{default:}\ \mathit{ss})$
  if $\mathit{returns}(\mathit{ss})$

Statement list checking $\boxed{\Delta; \Gamma; f \vdash ss \ \mathbf{ok}}$

$$\text{[T-Statements]}\ \frac{\forall i. \Delta; \Gamma; f \vdash s_i \ \mathbf{ok}}{\Delta; \Gamma; f \vdash \overline{s} \ \mathbf{ok}}$$

Statement checking $\boxed{\Delta; \Gamma; f \vdash s \ \mathbf{ok}}$

$$\text{[T-Block]}\ \frac{\Delta; \Gamma; f \vdash ss \ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{\{ } ss \texttt{ \}} \ \mathbf{ok}} \qquad \text{[T-ExprStmt]}\ \frac{\Delta; \Gamma \vdash e : \sigma}{\Delta; \Gamma; f \vdash e\texttt{;} \ \mathbf{ok}} \qquad \text{[T-EmptyStatement]}\ \frac{}{\Delta; \Gamma; f \vdash \texttt{;} \ \mathbf{ok}}$$

$$\text{[T-If]}\ \frac{\Delta; \Gamma \vdash e : \texttt{boolish} \quad \Delta; \Gamma; f \vdash s \ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{if (}e\texttt{)} \ s \ \mathbf{ok}} \qquad \text{[T-IfElse]}\ \frac{\Delta; \Gamma \vdash e : \texttt{boolish} \quad \Delta; \Gamma; f \vdash s_1 \ \mathbf{ok} \quad \Delta; \Gamma; f \vdash s_2 \ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{if (}e\texttt{)} \ s_1 \ \texttt{else} \ s_2 \ \mathbf{ok}}$$

$$\text{[T-ReturnVar]}\ \frac{\Delta(f) = (\overline{\sigma}) \to \tau \quad \Delta; \Gamma \vdash x : \tau}{\Delta; \Gamma; f \vdash \texttt{return} \ x\texttt{;} \ \mathbf{ok}} \qquad \text{[T-ReturnExpr]}\ \frac{\Delta(f) = (\overline{\sigma}) \to \tau \quad \Delta; \Gamma \vdash e : \tau \quad \mathit{type}(e) <: \tau}{\Delta; \Gamma; f \vdash \texttt{return} \ e\texttt{;} \ \mathbf{ok}}$$

$$\text{[T-ReturnVoid]}\ \frac{\Delta(f) = (\overline{\sigma}) \to \texttt{void}}{\Delta; \Gamma; f \vdash \texttt{return;} \ \mathbf{ok}}$$

$$\text{[T-While]}\ \frac{\Delta; \Gamma \vdash e : \texttt{boolish} \quad \Delta; \Gamma; f \vdash s \ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{while (}e\texttt{)} \ s \ \mathbf{ok}} \qquad \text{[T-DoWhile]}\ \frac{\Delta; \Gamma; f \vdash s \ \mathbf{ok} \quad \Delta; \Gamma \vdash e : \texttt{boolish}}{\Delta; \Gamma; f \vdash \texttt{do} \ s \ \texttt{while (}e\texttt{);} \ \mathbf{ok}}$$

$$\text{[T-For]}\ \frac{[\Delta; \Gamma \vdash e_1 : \sigma_1] \quad [\Delta; \Gamma \vdash e_2 : \texttt{boolish}] \quad [\Delta; \Gamma \vdash e_3 : \sigma_3] \quad \Delta; \Gamma; f \vdash s \ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{for (}[e_1]\texttt{;} \ [e_2]\texttt{;} \ [e_3]\texttt{)} \ s \ \mathbf{ok}}$$

$$\text{[T-Break]}\ \frac{}{\Delta; \Gamma; f \vdash \texttt{break} \ [lab]\texttt{;} \ \mathbf{ok}} \qquad \text{[T-Continue]}\ \frac{}{\Delta; \Gamma; f \vdash \texttt{continue} \ [lab]\texttt{;} \ \mathbf{ok}}$$

$$\text{[T-Label]}\ \frac{\Delta; \Gamma; f \vdash s \ \mathbf{ok}}{\Delta; \Gamma; f \vdash lab\texttt{:} \ s \ \mathbf{ok}}$$

$$\text{[T-Switch]}\ \frac{\Delta; \Gamma \vdash e : \sigma \quad \sigma <: \texttt{extern} \quad \forall i. \mathit{type}(v_i) <: \sigma \quad \forall i. \Delta; \Gamma; f \vdash ss_i \ \mathbf{ok} \quad [\Delta; \Gamma; f \vdash ss \ \mathbf{ok}]}{\Delta; \Gamma; f \vdash \texttt{switch (}e\texttt{) \{} \ \overline{\texttt{case} \ v_i\texttt{:} \ ss_i} \ [\texttt{default:} \ ss] \ \texttt{\}} \ \mathbf{ok}}$$

Case checking $\boxed{\Delta; \Gamma; f \vdash cd\ \mathbf{ok}}$

$$\frac{\Delta; \Gamma; f \vdash ss\ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{case}\ v\texttt{:}\ ss\ \mathbf{ok}}\ [\text{T-Case}] \qquad \frac{\Delta; \Gamma; f \vdash ss\ \mathbf{ok}}{\Delta; \Gamma; f \vdash \texttt{default:}\ ss\ \mathbf{ok}}\ [\text{T-Default}]$$

$$(\Delta \cdot \Gamma)(x) = \begin{cases} \Gamma(x) & \text{if } x \in dom(\Gamma) \\ \Delta(x) & \text{otherwise} \end{cases}$$

Expression checking $\boxed{\Delta; \Gamma \vdash e : \tau}$

$$\frac{-2^{31} \leq n < 2^{32}}{\Delta; \Gamma \vdash n : \texttt{constant}}\ [\text{T-Constant}] \qquad \frac{}{\Delta; \Gamma \vdash r : \texttt{double}}\ [\text{T-Double}]$$

$$\frac{(\Delta \cdot \Gamma)(x) = \tau}{\Delta; \Gamma \vdash x : \tau}\ [\text{T-VarRef}] \qquad \frac{\Delta; \Gamma \vdash e : \tau \qquad \tau <: (\Delta \cdot \Gamma)(x)}{\Delta; \Gamma \vdash x\ \texttt{=}\ e : \tau}\ [\text{T-Assign}]$$

$$\frac{m = 2^k - 1 \qquad (\Delta \cdot \Gamma)(x) = \texttt{view}^n_\tau \qquad \Delta; \Gamma \vdash e : \texttt{intish}}{\Delta; \Gamma \vdash x\texttt{[(}e\ \texttt{\&}\ m\texttt{)}\ \texttt{>>}\ n/8\texttt{]} : \tau}\ [\text{T-Load}]$$

$$\frac{m = 2^k - 1 \qquad (\Delta \cdot \Gamma)(x) = \texttt{view}^n_\tau \qquad \Delta; \Gamma \vdash e_1 : \texttt{intish} \qquad \Delta; \Gamma \vdash e_2 : \tau}{\Delta; \Gamma \vdash x\texttt{[(}e_1\ \texttt{\&}\ m\texttt{)}\ \texttt{>>}\ n/8\texttt{]}\ \texttt{=}\ e_2 : \tau}\ [\text{T-Store}]$$

$$\frac{(\Delta \cdot \Gamma)(f) = \texttt{imul} \qquad \forall i. \Delta; \Gamma \vdash e_i : \texttt{intish}}{\Delta; \Gamma \vdash f\texttt{(}e_1, e_2\texttt{)} : \texttt{signed}}\ [\text{T-IMul}]$$

$$\frac{(\Delta \cdot \Gamma)(f) = (\overline{\sigma}) \to \tau \qquad \forall i. \Delta; \Gamma \vdash e_i : \sigma_i}{\Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \tau}\ [\text{T-FunCall}]$$

$$\frac{(\Delta \cdot \Gamma)(f) = \texttt{function} \qquad \forall i. \Delta; \Gamma \vdash e_i : \texttt{extern}}{\Delta; \Gamma \vdash f\texttt{(}\overline{e}\texttt{)} : \texttt{unknown}}\ [\text{T-FFI}]$$

$$\frac{\Delta; \Gamma \vdash e_1 : \texttt{boolish} \qquad \Delta; \Gamma \vdash e_2 : \tau \qquad \Delta; \Gamma \vdash e_3 : \tau}{\Delta; \Gamma \vdash e_1\ \texttt{?}\ e_2\ \texttt{:}\ e_3 : \tau}\ [\text{T-Conditional}]$$

$$\frac{\forall i \leq n. \Delta; \Gamma \vdash e_i : \tau_i}{\Delta; \Gamma \vdash \texttt{(}\overline{e}\texttt{)} : \tau_n}\ [\text{T-Paren}]$$

$$\frac{unop : \_ \wedge (\sigma) \to \tau \wedge \_ \qquad \Delta; \Gamma \vdash e : \sigma}{\Delta; \Gamma \vdash unop\ e : \tau}\ [\text{T-Unop}]$$

$$\frac{binop : \_ \wedge (\sigma_1, \sigma_2) \to \tau \wedge \_ \qquad \Delta; \Gamma \vdash e_1 : \sigma_1 \qquad \Delta; \Gamma \vdash e_2 : \sigma_2}{\Delta; \Gamma \vdash e_1\ binop\ e_2 : \tau}\ [\text{T-Binop}]$$

$$\frac{\Delta; \Gamma \vdash e : \sigma \qquad \sigma <: \tau}{\Delta; \Gamma \vdash e : \tau}\ [\text{T-Sub}] \qquad \frac{\Delta; \Gamma \vdash e : \texttt{double}}{\Delta; \Gamma \vdash \texttt{~~}e : \texttt{signed}}\ [\text{T-Cast}]$$