TypeChecking

```
Se nel TE, facendo una lookup alla tabella dei simboli, trovo che id ha un tipo tau
allora nel TE riesco a verificare che id ha tipo tau.
```

$$\frac{\Gamma(id) = \tau}{\Gamma \vdash id : \tau}$$

```
Nel TE riesco a verificare che intconst è di tipo "integer".
```

$$\Gamma \vdash intconst : integer$$

```
Nel TE riesco a verificare che stringconst è di tipo "string".
```

$$\Gamma \vdash stringconst : string$$

```
Nel TE riesco a verificare che boolconst è di tipo "bool".
```

$$\Gamma \vdash boolconst : bool$$

```
Se nel TE stmt1 è del tipo "notype", e se nel TE stmt2 è del tipo "notype"
allora nel TE stmt1;stmt2 sarà del tipo "notype"
```

$$\frac{\Gamma \vdash stmt1 : notype \Gamma \vdash stmt2 : notype}{\Gamma \vdash stmt1; stmt2 : notype}$$

```
Nel TE se riesco a verificare che un certo numero di parametri di una funzione
sono di un determinato tipo tau, che la funzione restituisce un tipo tau
 e che nel TE ogni parametro è di tipo tau
allora la chiamata di funzione avente parametri e1,...,en è di tipo tau
```

$$\frac{\Gamma \vdash f : \tau \times ... \times \tau \rightarrow \tau \Gamma \vdash e_i : \tau_i^i \in 1...n}{\Gamma \vdash f(e_1, ..., e_n) : \tau}$$

```
Nel TE se riesco a verificare che un certo numero di parametri di una funzione
sono di un determinato tipo tau, che la funzione restituisce un tipo notype
e che nel TE ogni parametro è di tipo tau
allora la chiamata di funzione avente parametri e1,...,en è di tipo notype
```

$$\frac{\Gamma \vdash f : \tau \times ... \times \tau \rightarrow notype \Gamma \vdash e_i : \tau_i^i \in 1...n}{\Gamma \vdash f(e_1, ..., e_n) : notype}$$

```
Nel TE, facendo una lookup alla tabella dei simboli  e scopro che è di tipo taue e che, nel TE riesco a verificare
un'espressione e è di tipo tau, allora nel TE l'assegnazione tra un id ed una
espressione è di tipo notype.
```

$$\frac{\Gamma(id) = \tau \; \Gamma \vdash e : \tau}{\Gamma \vdash id := e : notype}$$

```
Nel TE esteso con id di un certo tipo tau, se verifico che stmtè di tipo notype
allora ne TE il avrò che id sarà di tipo tau insieme a stmt sarà di tipo notype.
```

$$\frac{\Gamma[id \to \tau] \vdash stmt : notype}{\Gamma \vdash \tau\ id\ ; stmt : notype}$$

```
Nel TE se riesco a verificare che una espressione è di tipo booleana e
nel TE riesco a verificare che block sia di tipo notype, allora
nel TE riesco a verificare che lo statement relativo al while sia di tipo notype
```

$$\frac{\Gamma \vdash e : boolean\ \Gamma \vdash block : notype}{\Gamma \vdash while\ e\ loop\ block\ end\ loop : notype}$$

```
Nel TE se riesco a verificare che una espressione e è di tipo boolean e che
nel TE un block è di tpo notype, allora
nel TE lo statement if è di tipo notype.
```

$$\frac{\Gamma \vdash e : boolean\ \ \Gamma \vdash block : notype}{\Gamma \vdash if\ e\ then\ block\ end\ if : notype}$$

```
Nel TE se riesco a verificare che una espressione è di tipo tau,
allora nel TE riesco a verificare che lo stetement di ritorno avrà tipo tau.
```

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash return\ e : notype}$$

```
Nel TE se riesco a verificare che una espressione è ti tipo tau, allora
nel TE riesco a verificare che lo stetement relativo alla write è di tipo notype.

(Le write della grammatica di MyFun comprendono diversi write. Il ragionamento è analogo.)
```

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash write\ e : notype}$$

```
Nel TE se facendo una lookup di id scopro che è di tipo tau e che nel TE riesco a
verificare che e è di tipo tau, allora la read sarà di tipo notype.

Questa stessa può essere ( read id) quindi è la stessa rule senza considerare
l'espressione.
```

$$\frac{\Gamma(id = \tau)\ \ \Gamma \vdash e : \tau}{\Gamma \vdash read\ id\ e : notype}$$

```
Nel TE se riesco a verificare che block ha tipo notype e nel TE stmt ha tipo notype
allora l'esle statement ha tipo notype.
```

$$\frac{\Gamma \vdash block : notype\ \ \Gamma \vdash stmt : notype}{\Gamma \vdash else\ block\ stmt : notype}$$

```
Nel TE se riesco a verificare che l'espressione e1 è di tipo tau con tau appartenente ad integer e real, e che nel TE
e2 è di tipo tau appartenente ad integer e real, allora nel TE l'addizione di due espressioni sarà di tipo tau appartenente integer o real.
```

```
(analogo concetto per la sottrazione, motiplicazione, divisione, potenza.
```

$$\frac{\Gamma \vdash e_1 : \tau \in \{integer, real\} \;\; \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \; plus \; e_2 : \tau}$$

```
Nel TE se riesco a verificare che l'espressione e1 è di tipo tau con tau appartenente ad integer e che nel TE
e2 è di tipo tau appartenente ad integer, allora nel TE l'addizione di due espressioni sarà di tipo tau appartenente ad integer.
```

$$\frac{\Gamma \vdash e_1 : \tau \in \{integer\} \;\; \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \; divint \; e_2 : \tau}$$

```
Nel TE se riesco a verificare che l'espressione e1 è di tipo tau e che nel TE
e2 è di tipo tau , allora nel TE la concatenazione di due espressioni sarà di tipo stringa.
```

$$\frac{\Gamma \vdash e_1 : \tau \;\; \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \; strconcat \; e_2 : string}$$

```
Nel TE se riesco a verificare che l'espressione e1 è di tipo tau e che nel TE
e2 è di tipo tau , allora nel TE l'addizione di due espressioni sarà di tipo bool.

La regola è analoga per: OR, NOT, GT, GE, LT, LE, EQ, NE.
```

$$\frac{\Gamma \vdash e_1 : \tau \;\; \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \; and \; e_2 : bool}$$

```
Nel TE esteso se faccio una lookup di id e scopro che è di tipo tau, allora
nel TE riesco a verificare che la dichiarazione del parametro è di tipo notype.

ParDecl -> type id
        | out type id;
```

$$\frac{\Gamma[id \rightarrow \tau]}{\Gamma \vdash \tau \; id : notype}$$

```
Nel TE esteso se faccio una lookup di id e scopro che è di tipo tau, allora
nel TE riesco a verificare che
```

$$\frac{\Gamma \vdash [id \rightarrow \tau]}{\Gamma \vdash out \; \tau \; id : notype}$$