

Traduction des langages

Typage

Objectif :

- Définir la nouvelle structure d'arbre obtenue après la passe de typage
- Définir les actions à réaliser par la passe de typage

1 Rappel : le typage du langage Rat

1.1 Grammaire du langage RAT

- | | |
|--|-------------------------------|
| 1. $PROG' \rightarrow PROG \$$ | 20. $TYPE \rightarrow int$ |
| 2. $PROG \rightarrow FUN PROG$ | 21. $TYPE \rightarrow rat$ |
| 3. $FUN \rightarrow TYPE id (DP) BLOC$ | 22. $E \rightarrow id (CP)$ |
| 4. $PROG \rightarrow id BLOC$ | 23. $E \rightarrow [E / E]$ |
| 5. $BLOC \rightarrow \{ IS \}$ | 24. $E \rightarrow num E$ |
| 6. $IS \rightarrow I IS$ | 25. $E \rightarrow denom E$ |
| 7. $IS \rightarrow \Lambda$ | 26. $E \rightarrow id$ |
| 8. $I \rightarrow TYPE id = E ;$ | 27. $E \rightarrow true$ |
| 9. $I \rightarrow id = E ;$ | 28. $E \rightarrow false$ |
| 10. $I \rightarrow const id = entier ;$ | 29. $E \rightarrow entier$ |
| 11. $I \rightarrow print E ;$ | 30. $E \rightarrow (E + E)$ |
| 12. $I \rightarrow if E BLOC else BLOC$ | 31. $E \rightarrow (E * E)$ |
| 13. $I \rightarrow while E BLOC$ | 32. $E \rightarrow (E = E)$ |
| 14. $I \rightarrow return E ;$ | 33. $E \rightarrow (E < E)$ |
| 15. $DP \rightarrow \Lambda$ | 34. $E \rightarrow (E)$ |
| 16. $DP \rightarrow TYPE id DP2$ | 35. $CP \rightarrow \Lambda$ |
| 17. $DP2 \rightarrow , TYPE id DP2$ | 36. $CP \rightarrow E CP2$ |
| 18. $DP2 \rightarrow \Lambda$ | 37. $CP2 \rightarrow , E CP2$ |
| 19. $TYPE \rightarrow bool$ | 38. $CP2 \rightarrow \Lambda$ |

1.2 Jugements de typage du langage RAT

Axiomes

- | | |
|---|--------------------------------|
| $\frac{x \in \sigma \quad \sigma(x) = \tau}{\sigma \vdash x : \tau}$ | — $\sigma \vdash true : bool$ |
| Si x a été ajouté plusieurs fois à σ , $\sigma(x)$ renvoie le dernier type associé à x . | — $\sigma \vdash false : bool$ |
| | — $\sigma \vdash entier : int$ |

Expression

| | |
|---|--|
| — $\frac{\sigma \vdash E_1 : int \quad \sigma \vdash E_2 : int}{\sigma \vdash [E_1 / E_2] : rat}$ | — $\frac{\sigma \vdash E_1 : int \quad \sigma \vdash E_2 : int}{\sigma \vdash (E_1 * E_2) : int}$ |
| — $\frac{\sigma \vdash E : rat}{\sigma \vdash num E : int}$ | — $\frac{\sigma \vdash E_1 : rat \quad \sigma \vdash E_2 : rat}{\sigma \vdash (E_1 * E_2) : rat}$ |
| — $\frac{\sigma \vdash E : rat}{\sigma \vdash denom E : int}$ | — $\frac{\sigma \vdash E_1 : int \quad \sigma \vdash E_2 : int}{\sigma \vdash (E_1 = E_2) : bool}$ |
| — $\frac{\sigma \vdash E : \tau}{\sigma \vdash (E) : \tau}$ | — $\frac{\sigma \vdash E_1 : bool \quad \sigma \vdash E_2 : bool}{\sigma \vdash (E_1 = E_2) : bool}$ |
| — $\frac{\sigma \vdash E_1 : int \quad \sigma \vdash E_2 : int}{\sigma \vdash (E_1 + E_2) : int}$ | — $\frac{\sigma \vdash E_1 : int \quad \sigma \vdash E_2 : int}{\sigma \vdash (E_1 < E_2) : bool}$ |
| — $\frac{\sigma \vdash E_1 : rat \quad \sigma \vdash E_2 : rat}{\sigma \vdash (E_1 + E_2) : rat}$ | — On se limitera à ces signatures. |

Structures de contrôle

| |
|---|
| — $\frac{\sigma \vdash E : bool \quad \sigma, \tau_r \vdash BLOC_1 : void \quad \sigma, \tau_r \vdash BLOC_2 : void}{\sigma, \tau_r \vdash if E BLOC_1 else BLOC_2 : void, []}$ |
| — $\frac{\sigma \vdash E : bool \quad \sigma, \tau_r \vdash BLOC : void}{\sigma, \tau_r \vdash while E BLOC : void, []}$ |

Déclaration / affectation

| |
|---|
| — $\frac{\sigma \vdash TYPE : \tau \quad \sigma \vdash E : \tau}{\sigma, \tau_r \vdash TYPE id = E : void, [id, \tau]}$ |
| — $\frac{\sigma \vdash id : \tau \quad \sigma \vdash E : \tau}{\sigma, \tau_r \vdash id = E : void, []}$ |

Autres instructions

| |
|---|
| — $\frac{}{\sigma, \tau_r \vdash const id = entier : void, [id, int]}$ |
| — $\frac{\sigma \vdash E : \tau}{\sigma, \tau_r \vdash print E : void, []}$ |

Déclaration de fonction

| |
|--|
| — $\frac{A \quad B \quad C}{\sigma \vdash TYPE id (DP) BLOC : void, [id, \tau_p \rightarrow \tau_r]}$ |
| — A : $\sigma \vdash TYPE : \tau_r$ |
| — B : $\sigma \vdash DP : \tau_p, \sigma_p \quad (DP = TYPE_{E_1} id_1, \dots, TYPE_{E_n} id_n)$ |
| — C : $(id, \tau_p \rightarrow \tau_r) :: \sigma_p @ \sigma, \tau_r \vdash BLOC : void$ |
| — $\frac{\sigma \vdash TYPE_1 : \tau_1 \quad \dots \quad TYPE_n : \tau_n}{\sigma \vdash TYPE_1 id_1, \dots, TYPE_n id_n : \tau_1 \times \dots \times \tau_n, [(id_1, \tau_1); \dots; (id_n, \tau_n)]}$ |

Appel de fonction

$$\begin{array}{l} \text{---} \frac{\sigma \vdash id : \tau_1 \rightarrow \tau_2 \quad E_1, \dots, E_n : \tau_1}{\sigma \vdash id (E_1, \dots, E_n) : \tau_2} \\ \text{---} \frac{\sigma \vdash E_1 : \tau_1 \quad \dots \quad E_n : \tau_n}{\sigma \vdash E_1, \dots, E_n : \tau_1 \times \dots \times \tau_n} \end{array}$$

Retour de fonction

$$\text{---} \frac{\sigma \vdash E : \tau_r}{\sigma, \tau_r \vdash return E : void, []}$$

Suite d'instructions

$$\begin{array}{l} \text{---} \frac{\sigma, \tau_r \vdash IS : void, \sigma'}{\sigma, \tau_r \vdash \{IS\} : void} \quad \frac{\sigma, \tau_r \vdash IS : void, \sigma'}{\sigma, \tau_r \vdash BLOC : void} \quad (BLOC \text{ et } \{IS\} \text{ sont la même chose}) \\ \text{---} \frac{\sigma, \tau_r \vdash I : void, \sigma' \quad \sigma' @ \sigma, \tau_r \vdash IS : void, \sigma''}{\sigma, \tau_r \vdash I IS : void, \sigma'' @ \sigma'} \\ \text{---} \sigma, \tau_r \vdash : void \end{array}$$

Le programme

$$\begin{array}{l} \text{---} \frac{\sigma \vdash FUN : void, \sigma' \quad \sigma' @ \sigma \vdash PROG : void, \sigma''}{\sigma \vdash FUN PROG : void, \sigma'' @ \sigma'} \\ \text{---} \frac{\sigma, void \vdash BLOC : void}{\sigma \vdash id BLOC : void} \end{array}$$

2 Passe de typage

Nous rappelons qu'un compilateur fonctionne par passes, chacune d'elle réalisant un traitement particulier (gestion des identifiants, typage, placement mémoire, génération de code, ...). Chaque passe parcourt, et potentiellement modifie, l'AST.

La seconde passe est une passe de typage. C'est elle qui vérifie la conformité des types déclarés et associe aux identifiants leurs informations de type.

2.1 Structure de l'AST après la passe de typage

La passe de typage réalise des vérifications de type qui nécessitent une mise à jour des informations de type de identificateurs. Elle prépare également les passes suivantes, par exemple en choisissant la "version" de l'opérateur à utiliser en cas de surcharge des opérateurs.

▷ **Exercice 1** Définir la structure de l'AST après la passe de typage.

2.2 Actions à réaliser lors de la passe de typage

▷ **Exercice 2** Définir les actions à réaliser lors de la passe de typage.