Erreurs étape B :

1. Règles communes aux trois passes de vérifications contextuelles :

- **Règle (0.1):**

$$\begin{array}{rl} \textbf{identifier} \downarrow env\_exp \uparrow def & \\ \quad \rightarrow \quad \underline{\texttt{Identifier}} \uparrow name & \\ \textbf{\textit{affectation}} \quad def := env\_exp(name) & \end{array} \tag{0.1}$$

  - raison : opération partielle env_exp(name)
  - message : « identificateur non déclaré »

- **Règle (0.2) :**

$$\begin{array}{rl} \textbf{type} \downarrow env\_types \uparrow type & \\ \quad \rightarrow \quad \underline{\texttt{Identifier}} \uparrow name & \\ \textbf{\textit{condition}} \quad (\_, type) \triangleq env\_types(name) & \end{array} \tag{0.2}$$

  - raison : opération partielle env_types(name)
  - message : « identificateur de type non déclaré »

2. Grammaire attribuée spécifiant la passe 1:

- **Règle (1.3) :**

$$\begin{array}{rl} \textbf{decl\_class} \downarrow env\_types \uparrow \{name \mapsto (\underline{\text{class}}(super, \{\}), \underline{\text{type\_class}}(name))\} \oplus env\_types & \\ \quad \rightarrow \quad \underline{\texttt{DeclClass}}[& \\ \qquad \underline{\texttt{Identifier}} \uparrow name & \\ \qquad \underline{\texttt{Identifier}} \uparrow super & \\ \qquad \textbf{LIST\_DECL\_FIELD} & \\ \qquad \textbf{LIST\_DECL\_METHOD}] & \\ \quad] & \\ \textbf{\textit{condition}} \quad env\_types(super) = (\underline{\text{class}}(\_), \_) & \end{array} \tag{1.3}$$

  - raison : opération paratielle [L'identificateur super doit être un identificateur de classe préalablement déclaré]
  - message : "identificateur non déclaré "
  - raison : condition
  - message : " identificateur de classe attendu"
  - raison : opération union disjointe
  - message : "classe ou type déjà déclaré "

3. Grammaire attribuée spécifiant la passe 2 :

- **Règle (2.3) :**

$$
\begin{array}{rl}
\textbf{decl\_class} \downarrow env\_types \uparrow env\_types_r & (2.3)\\
\quad \rightarrow & \underline{\texttt{DeclClass}}[\\
& \quad \underline{\texttt{Identifier}} \uparrow name\\
& \quad \underline{\texttt{Identifier}} \uparrow super\\
& \quad \textbf{list\_decl\_field} \downarrow env\_types \downarrow super \downarrow name \uparrow env\_exp_f\\
& \quad \textbf{list\_decl\_method} \downarrow env\_types \downarrow super \uparrow env\_exp_m\\
&]\\
\textbf{\textit{condition}} & (\underline{\text{class}}(\_, env\_exp\_super), \_) \triangleq env\_types(super)\\
\textbf{\textit{affectation}} & new\_def := (\underline{\text{class}}(super, (env\_exp_f \oplus env\_exp_m)/env\_exp\_super),\\
& \qquad \underline{\text{type\_class}}(name))\\
& env\_types_r := \{name \mapsto new\_def\}/env\_types
\end{array}
$$

- raison : opération partielle env_types(super)
- message : "en fait erreur interne : on doit pas arriver ici "
- raison : condition
- message : " identificateur de classe attendu"
- raison : opération partielle union disjointe
- message : "un nom de méthode redéclare un nom de champ "

- **Règle (2.4) :**

$$
\begin{array}{rl}
\textbf{list\_decl\_field} \downarrow env\_types \downarrow super \downarrow class \uparrow env\_exp_r & (2.4)\\
\quad \rightarrow & \{env\_exp_r := \{\}\}\\
& [\ (\textbf{decl\_field} \downarrow env\_types \downarrow super \downarrow class \uparrow env\_exp\\
& \quad \{env\_exp_r := env\_exp_r \oplus env\_exp\})^*\ ]
\end{array}
$$

- raison : opération partielle union disjointe .
- message : "nom d'attribut déjà déclaré ".

- **Règle (2.5) :**

$$
\begin{array}{rl}
\textbf{decl\_field} \downarrow env\_types \downarrow super \downarrow class \uparrow \{name \mapsto (\underline{\text{field}}(visib, class), type)\} & (2.5)\\
\quad \rightarrow & \underline{\texttt{DeclField}} \uparrow visib[\\
& \quad \textbf{type} \downarrow env\_types \uparrow type\\
& \quad \underline{\texttt{Identifier}} \uparrow name\\
& \quad \textbf{INITIALIZATION}\\
&]\\
\textbf{\textit{condition}} & type \neq \underline{\text{void}}\ \texttt{et},\\
& \texttt{si} \begin{cases} (\underline{\text{class}}(\_, env\_exp\_super), \_) \triangleq env\_types(super)\\ \texttt{et}\ env\_exp\_super(name)\ \text{est défini} \end{cases}\\
& \texttt{alors}\ env\_exp\_super(name) = (\underline{\text{field}}(\_, \_), \_).
\end{array}
$$

- raison : condition type != void .
- message : "le type attendu doit étre différent de void".
- raison : condition et opération partielle env_types(super)
- message : "en fait erreur interne : on doit pas arriver ici "

- raison :si le champ est deja définir dans le super class ---> elle doit étre de type field
- message : "le type attendu est field "

- **Règle (2.6) :**

$$\begin{array}{l}\textbf{list\_decl\_method}\ \downarrow env\_types\ \downarrow super\ \uparrow env\_exp_r \\ \qquad \rightarrow \quad \{env\_exp_r := \{\}\} \\ \qquad\quad [\ (\textbf{decl\_method}\ \downarrow env\_types\ \downarrow super\ \uparrow env\_exp \\ \qquad\qquad \{env\_exp_r := env\_exp_r \oplus env\_exp\})^*\ ]\end{array} \tag{2.6}$$

- raison : opération partielle union disjointe
- message : "méthode déja déclaré".

- **Règle (2.7) :**

$$\begin{array}{l}\textbf{decl\_method}\ \downarrow env\_types\ \downarrow super\ \uparrow \{name \mapsto (\underline{\text{method}}(sig), type)\} \\ \qquad \rightarrow \quad \texttt{DeclMethod}\ [\\ \qquad\qquad \textbf{type}\ \downarrow env\_types\ \uparrow type \\ \qquad\qquad \texttt{Identifier}\ \uparrow name \\ \qquad\qquad \textbf{list\_decl\_param}\ \downarrow env\_types\ \uparrow sig \\ \qquad\qquad \textbf{METHOD\_BODY} \\ \qquad\quad] \\ \qquad \textit{condition} \quad \texttt{si} \begin{cases} (\underline{\text{class}}(\_, env\_exp\_super), \_) \triangleq env\_types(super) \\ \texttt{et}\ env\_exp\_super(name)\ \text{est défini} \end{cases} \\ \qquad\qquad\quad \texttt{alors} \begin{cases} (\underline{\text{method}}(sig_2), type_2) \triangleq env\_exp\_super(name) \\ \texttt{et}\ sig = sig_2 \\ \texttt{et}\ \underline{\text{subtype}}(env\_types, type, type_2) \end{cases}\end{array} \tag{2.7}$$

- raison : condition et opération partielle env_types(super)
- message : "en fait erreur interne : on doit pas arriver ici "
- raison : condition
- message : " identificateur de classe attendu"
- Si une méthode est redéfinie alors :
  - raison : sig = sig2;
  - message : "une méthode redéfinie doit avoire la méme signature"
  - raison : subtype(env_types, type, type2)
  - message : "une méthode redéfinie doit avoir pour type de retour un sous-type du type de retour de la méthode héritée"

- **Règle (2.9) :**

$$\begin{array}{l}\textbf{decl\_param}\ \downarrow env\_types\ \uparrow type \\ \qquad \rightarrow \quad \texttt{DeclParam}[\ \textbf{type}\ \downarrow env\_types\ \uparrow type\ \texttt{Identifier}\ \uparrow \_\ ] \\ \qquad \textit{condition} \quad type \neq \underline{\text{void}}\end{array} \tag{2.9}$$

- raison :type != void
- message : "le type attendu doit étre différent de void"

4. Grammaire attribuée spécifiant la passe 3 :

- **Règle (3.5) :**

$$\begin{array}{lll} \textbf{decl\_class}\ \downarrow env\_types & & (3.5)\\ \quad \rightarrow & \underline{\texttt{DeclClass}}\ [& \\ & \quad \underline{\texttt{Identifier}}\ \uparrow class & \\ & \quad \underline{\texttt{Identifier}}\ \uparrow \_ & \\ & \quad \textbf{list\_decl\_field}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class & \\ & \quad \textbf{list\_decl\_method}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class & \\ &] & \\ \textit{condition} & (\underline{\text{class}}(\_, env\_exp), \_) \triangleq env\_types(class) & \end{array}$$

  - raison : opération partielle env_types(super)
  - message : "en fait erreur interne : on doit pas arriver ici "
  - raison : condition
  - message : " identificateur de classe attendu"

- **Règle (3.12) :**

$$\begin{array}{lll} \textbf{list\_decl\_param}\ \downarrow env\_types\ \uparrow env\_exp_r & & (3.12)\\ \quad \rightarrow & \{env\_exp_r := \{\}\} & \\ & [\ (\textbf{decl\_param}\ \downarrow env\_types\ \uparrow env\_exp & \\ & \quad \{env\_exp_r := env\_exp_r \oplus env\_exp\})^*\ ] & \end{array}$$

  - raison : opération union disjointe env_expr ⊕ env_exp
  - message : " paramètre deja déclarée "

- **Règle (3.17) :**

$$\begin{array}{lll} \textbf{decl\_var}\ \downarrow env\_types\ \downarrow env\_exp\_sup\ \downarrow env\_exp\ \downarrow class\ \uparrow \{name \mapsto (\underline{\text{var}}, type)\} \oplus env\_exp & & (3.17)\\ \quad \rightarrow & \underline{\texttt{DeclVar}}[& \\ & \quad \textbf{type}\ \downarrow env\_types\ \uparrow type & \\ & \quad \underline{\texttt{Identifier}}\ \uparrow name & \\ & \quad \textbf{initialization}\ \downarrow env\_types\ \downarrow env\_exp/env\_exp\_sup\ \downarrow class\ \downarrow type & \\ &] & \\ \textit{condition} & type \neq \underline{\text{void}} & \end{array}$$

  - raison : opération union disjointe
  - message : " variable deja déclarée "
  - raison : condition type != void
  - message : "le type attendu doit étre différent de void"

- **Règle (3.24) :**

$$\begin{array}{rl} \rightarrow & \underline{\texttt{Return}}\ [\ \textbf{rvalue}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class\ \downarrow return\ ] \\ \textbf{\textit{condition}} & return \neq \underline{\text{void}} \end{array} \tag{3.24}$$

- raison : condition return != void
- message : " paramètre deja déclarée "

- **Règle (3.28) :**

$$\begin{array}{rl} \textbf{rvalue}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class\ \downarrow type_1 & \\ \rightarrow & \textbf{expr}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class\ \uparrow type_2 \\ \textbf{\textit{condition}} & \text{assign\_compatible}(env\_types, type_1, type_2) \end{array} \tag{3.28}$$

- raison : condition assign_compatible(env_types, type1, type2)
- message : "cette affection n'est pas cohérente "

- **Règle (3.29) :**

$$\begin{array}{rl} \textbf{condition}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class & \\ \rightarrow & \textbf{expr}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class\ \uparrow \underline{\text{boolean}} \end{array} \tag{3.29}$$

- raison : Filtrage d'un attribut synthétisé en partie droite : doit étre boolean .
- message : "type attendu est boolean".

- **Règle (3.31) :**

$$\begin{array}{rl} \textbf{exp\_print}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class & \\ \rightarrow & \textbf{expr}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class\ \uparrow type \\ \textbf{\textit{condition}} & type = \underline{\text{int}}\ \text{ou}\ type = \underline{\text{float}}\ \text{ou}\ type = \underline{\text{string}} \end{array} \tag{3.31}$$

- raison : condition type = int ou type = float ou type = string
- message : "le type attendu int , float , string "

- **Règle (3.39) :**

$$\begin{array}{rl} \rightarrow & \underline{\texttt{Cast}}\ [\\ & \quad \textbf{type}\ \downarrow env\_types\ \uparrow type \\ & \quad \textbf{expr}\ \downarrow env\_types\ \downarrow env\_exp\ \downarrow class\ \uparrow type_2 \\ &] \\ \textbf{\textit{condition}} & \text{cast\_compatible}(env\_types, type_2, type) \end{array} \tag{3.39}$$

- raison : condition cast_compatible(env_types, type2, type)
- message : "casting non compatible "

- **Règle (3.42) :**

$$\begin{array}{rl} \rightarrow & \underline{\texttt{New}}\ [\ \textbf{type}\ \downarrow env\_types\ \uparrow type\ ] \\ \textbf{\textit{condition}} & type = \underline{\text{type\_class}}(\_) \end{array} \tag{3.42}$$

- raison : condition type = type_class(__)
- message : "le type attendu : type_class"

- **Règle (3.43) :**

$$\begin{array}{rl} \rightarrow & \underline{\texttt{This}} \\ \textbf{\textit{condition}} & class \neq 0 \\ \textbf{\textit{affectation}} & type := \underline{\text{type\_class}}(class) \end{array} \quad (3.43)$$

- raison : condition class!=0
- "class non définie"

- **Règle (3.65) :**

$$\begin{array}{rl} \rightarrow & \underline{\texttt{Selection}}\ [\\ & \quad \textbf{expr} \downarrow env\_types \downarrow env\_exp \downarrow class \uparrow \underline{\text{type\_class}}(class_2) \\ & \quad \textbf{field\_ident} \downarrow env\_exp_2 \uparrow \underline{\text{public}} \uparrow\_ \uparrow type \\ &] \\ \textbf{\textit{condition}} & (\underline{\text{class}}(\_, env\_exp_2), \_) \triangleq env\_types(class_2) \end{array} \quad (3.65)$$

- raison : opération partielle env_types(class2)
- message : "en fait erreur interne : on doit pas arriver ici "
- raison : condition
- message : " identificateur de classe attendu"
- raison : Filtrage d'un attribut synthétisé en partie droite: field_ident ↓env_exp2 ↑public ↑__ ↑type
- message : "accès pas possible : doit étre public "

- **Règle (3.66) :**

$$\begin{array}{rl} \rightarrow & \underline{\texttt{Selection}}\ [\\ & \quad \textbf{expr} \downarrow env\_types \downarrow env\_exp \downarrow class \uparrow \underline{\text{type\_class}}(class_2) \\ & \quad \textbf{field\_ident} \downarrow env\_exp_2 \uparrow \underline{\text{protected}} \uparrow class\_field \uparrow type \\ &] \\ \textbf{\textit{condition}} & (\underline{\text{class}}(\_, env\_exp_2), \_) \triangleq env\_types(class_2) \\ & \texttt{et}\ \text{subtype}(env\_types, \underline{\text{type\_class}}(class_2), \underline{\text{type\_class}}(class)) \\ & \texttt{et}\ \text{subtype}(env\_types, \underline{\text{type\_class}}(class), \underline{\text{type\_class}}(class\_field)) \end{array} \quad (3.66)$$

- raison : opération partielle env_types(class2)
- message : "en fait erreur interne : on doit pas arriver ici "
- raison : condition
- message : " identificateur de classe attendu"
- raison : Filtrage d'un attribut synthétisé en partie droite: field_ident ↓env_exp2 ↑protected ↑class_field ↑type
- message : "visibilité attendu : protected "
- raison : subtype(env_types,type_class(class2),type_class(class))
- message : "le type de class2 doit étre un sous type de classe"

- raison : subtype(env_types,type_class(class),type_class(class_field))
- message : "le type de class doit étre un sous de type de class field "

- **Règle (3.67) , (3.68) , (3.69) :**

$$\begin{aligned}
&\textbf{lvalue_ident}\ \downarrow env_exp\ \uparrow type && (3.67)\\
&\quad\rightarrow\ \textbf{identifier}\ \downarrow env_exp\ \uparrow(\underline{\text{field}}(_,_), type)\\
&\quad\rightarrow\ \textbf{identifier}\ \downarrow env_exp\ \uparrow(\underline{\text{param}}, type) && (3.68)\\
&\quad\rightarrow\ \textbf{identifier}\ \downarrow env_exp\ \uparrow(\underline{\text{var}}, type) && (3.69)
\end{aligned}$$

- raison : Filtrage d'un attribut synthétisé en partie droite
- message :"nature attendu : champ , paramètre , varibale

- **Règle (3.70) :**

$$\begin{aligned}
&\textbf{field_ident}\ \downarrow env_exp\ \uparrow visib\ \uparrow class\ \uparrow type && (3.70)\\
&\quad\rightarrow\ \textbf{identifier}\ \downarrow env_exp\ \uparrow(\underline{\text{field}}(visib, class), type)
\end{aligned}$$

- raison : Filtrage d'un attribut synthétisé en partie droite
- message : "type attendu : field "

- **Règle (3.71) :**

$$\begin{aligned}
&\textbf{method_call}\ \downarrow env_types\ \downarrow env_exp\ \downarrow class\ \uparrow type && (3.71)\\
&\quad\rightarrow\ \underline{\texttt{MethodCall}}\ [\\
&\qquad\textbf{expr}\ \downarrow env_types\ \downarrow env_exp\ \downarrow class\ \uparrow\underline{\text{type_class}}(class_2)\\
&\qquad\textbf{method_ident}\ \downarrow env_exp_2\ \uparrow sig\ \uparrow type\\
&\qquad[\ \textbf{rvalue_star}\ \downarrow env_types\ \downarrow env_exp\ \downarrow class\ \downarrow sig\ ]\\
&\quad]\\
&\textbf{\textit{condition}}\quad (\underline{\text{class}}(_, env_exp_2), _) \triangleq env_types(class_2)
\end{aligned}$$

- raison : opération partielle env_types(class2)
- message : "en fait erreur interne : on doit pas arriver ici "
- raison : condition
- message : " identificateur de classe attendu"

- **Règle (3.72) :**

$$\begin{aligned}
&\textbf{method_ident}\ \downarrow env_exp\ \uparrow sig\ \uparrow type && (3.72)\\
&\quad\rightarrow\ \textbf{identifier}\ \downarrow env_exp\ \uparrow(\underline{\text{method}}(sig), type)
\end{aligned}$$

- raison : Filtrage d'un attribut synthétisé en partie droite
- message : "type attendu : méthode "

- **Règle (3.73) :**

$$\begin{aligned}
&\textbf{rvalue_star}\ \downarrow env_types\ \downarrow env_exp\ \downarrow class\ \downarrow[\,] && (3.73)\\
&\quad\rightarrow\ \varepsilon
\end{aligned}$$

- raison : Filtrage d’un attribut hérité en partie gauche
- message : "signature attendu : []"