Typage

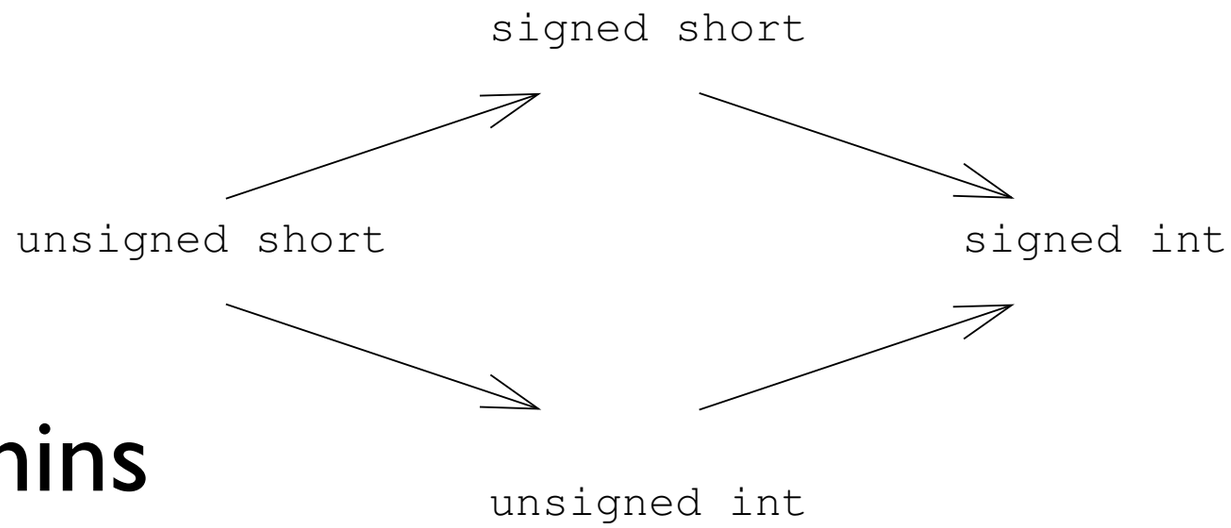
... et plus spécialement de ML

Types

- Typage à la **compilation** (ML, Pascal, C)
  vs. à **l'exécution** (Lisp, Python)
- Typage **fort** (ML, Pascal) vs. **faible** (C)
- Polymorphisme **ad hoc** (C, Java)
  vs. **paramétrique** (ML)
- **Vérification** (C) vs.
  inférence de types (ML)

Conversions (C)

- Un graphe de conversions fixe
- Recherche de chemins dans ce graphe



Typage de ML

- Typage **fort**
- Types **inférés**: pas la peine d'annoter les variables avec leur type
- Typage **polymorphe**: une expression peut avoir plusieurs types en même temps

Syntaxe de pureML

- pureML ~ $\mathrm{PCF_V}$, mais sans annotations de types

$s, t, u, v, \ldots$	$::=$	$x, y, z, \ldots$	variables
	$\mid$	$\dot{n}$	constantes entières
	$\mid$	uv	application
	$\mid$	$\texttt{letrec}\ f(x) = u\ \texttt{in}\ v$	fonction récursive
	$\mid$	$\texttt{let}\ x = u\ \texttt{in}\ v$	définition locale
	$\mid$	$u \dot{+} v$	addition
	$\mid$	$\dot{-} u$	opposé
	$\mid$	$\texttt{if}\ u = 0\ \texttt{then}\ v\ \texttt{else}\ w$	conditionnelle.

Inférence de types

- Il existe un **algorithme** décidant si un programme est typable,
- et reconstruisant un **typage principal** si c'est le cas
- On commence par le cas plus simple de *monomorphic pure ML*

Types (monomorphes)

- Les **types** contiennent des variables de types

$$
\begin{array}{rcll}
\sigma, \tau, \ldots & ::= & \alpha, \beta, \ldots & \text{variables de type} \\
& | & \texttt{int} & \text{type des entiers} \\
& | & \sigma \to \tau & \text{type fonction.}
\end{array}
$$

- Les variables de type seront (ici) des **inconnues** de types.

Monomorphic pure ML

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\,(Var :_{\text{simple}}) \qquad \frac{}{\Gamma \vdash \dot{n} : \texttt{int}}\,(\texttt{int} :)$$

$$\frac{\Gamma \vdash u : \sigma \to \tau \quad \Gamma \vdash v : \sigma}{\Gamma \vdash uv : \tau}\,(App :) \qquad \frac{\Gamma \vdash u : \sigma \quad \Gamma, x : \sigma \vdash v : \tau}{\Gamma \vdash \texttt{let}\ x = u\ \texttt{in}\ v : \tau}\,(Let :_{\text{simple}})$$

$$\frac{\Gamma, f : \sigma \to \tau, x : \sigma \vdash u : \tau \quad \Gamma, f : \sigma \to \tau \vdash v : \lambda}{\Gamma \vdash \texttt{letrec}\ f(x) = u\ \texttt{in}\ v : \lambda}\,(Letrec :_{\text{simple}})$$

$$\frac{\Gamma \vdash u : \texttt{int} \quad \Gamma \vdash v : \texttt{int}}{\Gamma \vdash u \dot{+} v : \texttt{int}}\,(+ :) \qquad \frac{\Gamma \vdash u : \texttt{int}}{\Gamma \vdash \dot{-} u : \texttt{int}}\,(- :)$$

$$\frac{\Gamma \vdash u : \texttt{int} \quad \Gamma \vdash v : \tau \quad \Gamma \vdash w : \tau}{\Gamma \vdash \texttt{if}\ u = 0\ \texttt{then}\ v\ \texttt{else}\ w : \tau}\,(\texttt{if} :)$$

L'algorithme de Hindley

- On crée une variable de type α_t pour chaque sous-expression *t* du prog u_0 à typer
- On pose les équations (transp. suivant)
- On résout par **unification**.

Equations de typage

— *pour les variables, rien ;*
— *pour les constantes* $\dot{n}$*, l'équation* $\alpha_{\dot{n}} \doteq \texttt{int}$ *;*
— *pour les applications* uv*, l'équation* $\alpha_u \doteq \alpha_v \to \alpha_{uv}$ *;*
— *pour les expressions de la forme* $\texttt{let}\ x = u\ \texttt{in}\ v$*, les équations* $\alpha_x \doteq \alpha_u$ *et* $\alpha_{\texttt{let}\ x=u\ \texttt{in}\ v} \doteq \alpha_v$ *;*
— *pour les expressions de la forme* $\texttt{letrec}\ f(x) = u\ \texttt{in}\ v$*, les équations* $\alpha_f \doteq \alpha_x \to \alpha_u$ *et* $\alpha_{\texttt{letrec}\ f(x)=u\ \texttt{in}\ v} \doteq \alpha_v$ *;*
— *pour les expressions* $u \dot{+} v$*, les équations* $\alpha_u \doteq \texttt{int}$*,* $\alpha_v \doteq \texttt{int}$ *et* $\alpha_{u \dot{+} v} \doteq \texttt{int}$ *;*
— *pour les expressions* $\dot{-} u$*, les équations* $\alpha_u \doteq \texttt{int}$ *et* $\alpha_{\dot{-} u} \doteq \alpha_u$ *;*
— *pour les expressions* $\texttt{if}\ u = 0\ \texttt{then}\ v \texttt{else} w$*, les équations* $\alpha_u \doteq \texttt{int}$*,* $\alpha_{\texttt{if}\ u=0\ \texttt{then}\ v\texttt{else}w} \doteq \alpha_u$ *et* $\alpha_{\texttt{if}\ u=0\ \texttt{then}\ v\texttt{else}w} \doteq \alpha_v$*.*

Exemple

$$\texttt{letrec}\ f(x) = x\ \texttt{in}\ \texttt{letrec}\ g(h) = h(h\dot{3})\ \texttt{in}\ gf$$

(pour u_0 lui-même :) $\alpha_{u_0} \doteq \alpha_{\texttt{letrec}\ g(h)=h(h\dot{3})\ \texttt{in}\ gf}$ $\alpha_f \doteq \alpha_x \to \alpha_x$

(pour le sous-terme $\texttt{letrec}\ g(h) = h(h3)\ \texttt{in}\ gf$:) $\alpha_{\texttt{letrec}\ g(h)=h(h\dot{3})\ \texttt{in}\ gf} \doteq \alpha_{gf}$ $\alpha_g \doteq \alpha_h \to \alpha_{h(h\dot{3})}$

(pour le sous-terme $h(h\dot{3})$:) $\alpha_h \doteq \alpha_{h\dot{3}} \to \alpha_{h(h\dot{3})}$

(pour le sous-terme $h\dot{3}$:) $\alpha_h \doteq \alpha_{\dot{3}} \to \alpha_{h\dot{3}}$

(pour le sous-terme $\dot{3}$:) $\alpha_{\dot{3}} \doteq \texttt{int}$

(pour le sous-terme gf) $\alpha_g \doteq \alpha_f \to \alpha_{gf}$.

Termes

- $t ::= x$ (variables)
  $\mid f(t_1, \ldots, t_n)$ (applications)
  où f/n pris parmi une signature Σ
- Ce sont juste des arbres finis…
- Types = termes sur la signature `int`/0, $\rightarrow$/2

Substitutions

- $\theta = [x_1 \mapsto t_1, \ldots, x_n \mapsto t_n]$
  spécifie le remplacement de chaque x_i par t_i
- Formellement, θ :Var $\rightarrow$ Term telle que $\theta(x)=x$ pour toute var. x sauf un nombre fini
- Application de substitution $M\theta$:
  - $x_i\,\theta = t_i$
  - $x\theta = x$ si $x \neq x_i$ pour tout i
  - $f(t_1, \ldots, t_n)\theta = f(t_1\theta, \ldots, t_n\theta)$

Unification

- Résolution de systèmes finis d'équations

$$s_i = t_i \qquad (1 \leq i \leq n)$$

dans l'algèbre des termes

- Solution: θ telle que $s_i\theta = t_i\theta$ pour tout i
- Existence d'une solution: décidable
(et même en temps polynomial)
- S'il existe une solution, il existe un **mgu**
(définition: plus tard)

Unification (naïve)

$$
\begin{array}{lrcl}
(\textbf{Dec}) & (E \cup \{f(s_1,\cdots,s_m) \doteq f(t_1,\cdots,t_m)\},\theta) & \to & (E \cup \{s_1 \doteq t_1,\cdots,s_m \doteq t_m\},\theta) \\
(\textbf{DecFail}) & (E \cup \{f(s_1,\cdots,s_m) \doteq g(t_1,\cdots,t_n)\},\theta) & \to & \mathsf{Fail} \quad \text{si } f \neq g \\
(\textbf{Triv}) & (E \cup \{x \doteq x\},\theta) & \to & (E,\theta) \\
(\textbf{Bind}) & (E \cup \{x \doteq t\},\theta) & \to & (E[x := t],\theta[x := t]) \\
 & & & \text{si } t \neq x, x \text{ pas libre dans } t \\
(\textbf{Bind}') & (E \cup \{t \doteq x\},\theta) & \to & (E[x := t],\theta[x := t]) \\
 & & & \text{si } t \neq x, x \text{ pas libre dans } t \\
(\textbf{Check}) & (E \cup \{x \doteq t\},\theta) & \to & \mathsf{Fail} \\
 & & & \text{si } t \neq x, x \text{ libre dans } t \\
(\textbf{Check}') & (E \cup \{t \doteq x\},\theta) & \to & \mathsf{Fail} \\
 & & & \text{si } t \neq x, x \text{ libre dans } t
\end{array}
$$

- (*E* multi-ensemble~ens. avec multiplicités~listes mod permutation)
- **Thm:** correction, termination.

Composition

- **Defn:** θθ' est l'unique substitution telle que pour tout terme *t*, *t*(θθ')=(*t*θ)θ'. Explicitement: θθ'(*x*) = θ(*x*)θ'
- Ex: [*x*:=*f*(*y*)] [*y*:=*a*] = [*x*:=*f*(*a*), *y*:=*a*]
- **Lemme**: (θθ')θ'' = θ(θ'θ'')

mgu

- **Defn:** $\theta \leq \theta'$ (θ plus générale que θ') ssi
  il existe θ'' / $\theta' = \theta\theta''$

- Ex: $[x:=f(y)] \leq [x:=f(a), y:=a]$

- **Defn:** Un **mgu** de E est
  une substitution θ telle que:
  — θ **unifie** E: pour tout $(s=t) \in E$, $s\theta = t\theta$
  — pour tout unificateur θ' de E, $\theta \leq \theta'$

Correction

- **Defn:** une **solution** de (E, θ) est θθ'
  où θ' unifie E. **Fail** n'a pas de solution.
- **Thm** (correction): si $St \rightarrow^* St'$ alors St et St' ont le même ensemble de solutions.
- Il suffit de le montrer lorsque $St \rightarrow St'$ (+ récurrence)

Correction

- **Defn:** une solution de (E, θ) est θθ' où θ' unifie E. **Fail** n'a pas de solution.
- **Lemme**: si $St \rightarrow St'$ alors St et St' ont le même ensemble de solutions.
  - Cas (Bind): $St=(E \cup \{x=t\}, \theta)$, $St'=(E[x:=t], \theta[x:=t])$, x non libre dans t, $x \neq t$.
  - Si θ' unifie $E \cup \{x=t\}$, soit θ"=restr. de θ' à Var–$\{x\}$ $[x:=t]\theta'' = \theta'$ [pourquoi?]; et θ" unifie $E[x:=t]$
  - Si θ" unifie $E[x:=t]$, $\theta' = [x:=t]\theta''$ unifie E [ok?]

Correction

- **Defn:** une solution de (E, θ) est θθ' où θ' unifie E. **Fail** n'a pas de solution.
- **Lemme**: si $St \rightarrow St'$ alors St et St' ont le même ensemble de solutions.
  - Cas (Check): $St=(E \cup \{x=t\}, \theta)$, $St'=$ **Fail**,
    x libre dans t, $x \neq t$.
  - Il suffit de montrer qu'aucune θ' n'unifie $E \cup \{x=t\}$, en fait $x\theta'$ ne peut pas être égal à $t\theta'$ [pourquoi?]

Correction

- **Defn:** une **solution** de (E, θ) est θθ'
  où θ' unifie E. **Fail** n'a pas de solution.
- **Thm** (correction): si $St \rightarrow^* St'$ alors St et St' ont le même ensemble de solutions.
- **Corl:** si (E, []) $\rightarrow^*$ (∅, θ) alors θ est un **mgu** de E; si (E, []) $\rightarrow^*$ **Fail** alors E n'a pas d'unificateur.

Terminaison

- **Thm:** toute suite de réductions → est finie.
  - $|s{=}t| = |s| + |t|$, $|E| = \Sigma |s{=}t|$, $(s{=}t)$ in E, $|\mathbf{Fail}| = 0$.
  - (Bind), (Bind') font décroître #vars strictement
  - Les autres règles préservent #vars ou le font décroître, et font décroître $|E|$.

Complexité

- Cet algo. est en temps **exponentiel**
  $E = \{x_1 = f(x_2,x_2), x_2 = f(x_3,x_3), \ldots, x_{n-1} = f(x_n,x_n)\}$
- On peut obtenir un algo.
  en temps **polynomial** en:
  — représentant θ en forme **triangulaire**,
  e.g. $[x_1 := f(x_2,x_2)][x_2 := f(x_3,x_3)]\ldots[x_{n-1} := f(x_n,x_n)]$
  — n'effectuant **aucune** application de substitution

En temps polynomial

$$
\begin{array}{lrcl}
(\mathbf{Dec}) & (E \cup \{f(s_1,\cdots,s_m) \doteq f(t_1,\cdots,t_m)\},\vartheta) & \to & (E \cup \{s_1 \doteq t_1,\cdots,s_m \doteq t_m\},\vartheta) \\
(\mathbf{DecFail}) & (E \cup \{f(s_1,\cdots,s_m) \doteq g(t_1,\cdots,t_n)\},\vartheta) & \to & \mathsf{Fail} \quad \text{si } f \neq g \\
(\mathbf{Triv}) & (E \cup \{x \doteq x\},\vartheta) & \to & (E,\vartheta) \\
(\mathbf{LazyRep}) & (E \cup \{x \doteq t\},\vartheta) & \to & (E \cup \{u \doteq t\},\vartheta) \\
 & & & \text{si } t \neq x, bound(\vartheta,x) = \texttt{Some}\ u \\
(\mathbf{LazyRep'}) & (E \cup \{t \doteq x\},\vartheta) & \to & (E \cup \{t \doteq u\},\vartheta) \\
 & & & \text{si } t \neq x, bound(\vartheta,x) = \texttt{Some}\ u \\
(\mathbf{Bind}) & (E \cup \{x \doteq t\},\vartheta) & \to & (E,\vartheta;[x := t]) \\
 & & & \text{si } t \neq x, bound(\vartheta,x) = \texttt{None} \text{ et non } occ(\vartheta,t,x) \\
(\mathbf{Bind'}) & (E \cup \{t \doteq x\},\vartheta) & \to & (E,\vartheta;[x := t]) \\
 & & & \text{si } t \neq x, bound(\vartheta,x) = \texttt{None} \text{ et non } occ(\vartheta,t,x) \\
(\mathbf{Check}) & (E \cup \{x \doteq t\},\vartheta) & \to & \mathsf{Fail} \\
 & & & \text{si } t \neq x, bound(\vartheta,x) = \texttt{None} \text{ et } occ(\vartheta,t,x) \\
(\mathbf{Check'}) & (E \cup \{t \doteq x\},\vartheta) & \to & \mathsf{Fail} \\
 & & & \text{si } t \neq x, bound(\vartheta,x) = \texttt{None} \text{ et } occ(\vartheta,t,x)
\end{array}
$$

- Voir Exercise 4 (prog1_sem3.pdf)

ML monomorphe

- **Thm.** Etant donné un terme M, on peut:
  — décider si M **typable** (en ML monomorphe)
  — si oui, en fournir un **typage le plus général** $\Gamma \vdash M : \tau$
    (les autres sont $\Gamma\theta, \Delta \vdash M : \tau\theta$)
    en **temps polynomial**.

Typage de ML

- Typage **fort**
- Types **inférés**: pas la peine d'annoter les variables avec leur type

- Typage **polymorphe**: une expression peut avoir plusieurs types en même temps

Types et schémas

- Les **types** contiennent des variables de types

$$\begin{array}{rcll} \sigma, \tau, \ldots & ::= & \alpha, \beta, \ldots & \text{variables de type} \\ & | & \texttt{int} & \text{type des entiers} \\ & | & \sigma \to \tau & \text{type fonction.} \end{array}$$

- Les **schémas** de types quantifient ces variables

$$\forall \vec{\alpha} \cdot \sigma$$

- Double rôle des variables: inconnues/quantifiées

Typage

$$\frac{}{\Gamma, x : \forall \alpha_1, \cdots, \alpha_n \cdot \sigma \vdash x : \sigma[\alpha_1 := \tau_1, \cdots, \alpha_n := \tau_n]}\ (Var :) \qquad \frac{}{\Gamma \vdash \dot{n} : \texttt{int}}\ (\texttt{int} :)$$

$$\frac{\Gamma \vdash u : \sigma \to \tau \quad \Gamma \vdash v : \sigma}{\Gamma \vdash uv : \tau}\ (App :) \qquad \frac{\Gamma \vdash u : \sigma \quad \Gamma, x : \forall \vec{\alpha} \cdot \sigma \vdash v : \tau}{\Gamma \vdash \texttt{let}\ x = u\ \texttt{in}\ v : \tau}\ (Let :)$$

où $\vec{\alpha} \subseteq \text{ftv}(\sigma) \setminus \text{ftv}(\Gamma)$

$$\frac{\Gamma, f : \forall \cdot \sigma \to \tau, x : \forall \cdot \sigma \vdash u : \tau \quad \Gamma, f : \forall \vec{\alpha} \cdot \sigma \to \tau \vdash v : \lambda}{\Gamma \vdash \texttt{letrec}\ f(x) = u\ \texttt{in}\ v : \lambda}\ (Letrec :)$$

où $\vec{\alpha} \subseteq \text{ftv}(\sigma \to \tau) \setminus \text{ftv}(\Gamma)$

$$\frac{\Gamma \vdash u : \texttt{int} \quad \Gamma \vdash v : \texttt{int}}{\Gamma \vdash u \dot{+} v : \texttt{int}}\ (+ :) \qquad \frac{\Gamma \vdash u : \texttt{int}}{\Gamma \vdash \dot{-} u : \texttt{int}}\ (- :)$$

$$\frac{\Gamma \vdash u : \texttt{int} \quad \Gamma \vdash v : \tau \quad \Gamma \vdash w : \tau}{\Gamma \vdash \texttt{if}\ u = 0\ \texttt{then}\ v\ \texttt{else}\ w : \tau}\ (\texttt{if} :)$$

Renommage

- $\forall \underline{\alpha}.\sigma \equiv \forall \underline{\beta}.\tau$ ssi on obtient l'un par l'autre par renommage bijectif entre $\underline{\alpha}$ et $\underline{\beta}$
- On étend cette notion aux contextes Γ
- **Lemme.** Si $\Gamma \vdash u{:}\boldsymbol{\tau}$ est dérivable, et $\Gamma \equiv \Gamma'$, alors $\Gamma' \vdash u{:}\boldsymbol{\tau}$ est dérivable par une preuve de même taille

Substitutions et typage

- **Lemme.** Si $\Gamma \vdash u{:}\tau$ est dérivable, alors pour toute substitution θ, $\Gamma\theta \vdash u{:}\tau\theta$ est dérivable aussi.

- **Preuve.** Par récurrence sur la dérivation. Attention aux conditions de bord sur (Let:) et (Letrec:)!
  (On utilise le lemme précédent.)

Sém. à grands pas

$$\frac{}{E \vdash x \Rightarrow E(x)} \; (Var) \quad \text{si } x \in \text{dom } E \qquad \frac{}{E \vdash \dot{n} \Rightarrow \dot{n}} \; (Cst)$$

$$\frac{E \vdash u \Rightarrow \langle \texttt{rec}\, f(x) = u', E' \rangle \quad E \vdash v \Rightarrow V \quad E'[f \mapsto \langle \texttt{rec}\, f(x) = u', E' \rangle, x \mapsto V] \vdash u' \Rightarrow V'}{E \vdash uv \Rightarrow V'} \; (App)$$

$$\frac{E \vdash u \Rightarrow V \quad E[x \mapsto V] \vdash v \Rightarrow V'}{E \vdash \texttt{let}\, x = u\, \texttt{in}\, v \Rightarrow V'} \; (Let) \qquad \frac{E[f \mapsto \langle \texttt{rec}\, f(x) = u, E \rangle] \vdash v \Rightarrow V}{E \vdash \texttt{letrec}\, f(x) = u\, \texttt{in}\, v \Rightarrow V} \; (Letrec)$$

$$\frac{E \vdash u \Rightarrow \dot{n}_1 \quad E \vdash v \Rightarrow \dot{n}_2}{E \vdash u \dot{+} v \Rightarrow \dot{n}} \; (\dot{+}) \quad \text{où } n = n_1 + n_2 \qquad \frac{E \vdash u \Rightarrow \dot{n}}{E \vdash \dot{-}u \Rightarrow \dot{-}n} \; (\dot{-})$$

$$\frac{E \vdash u \Rightarrow \dot{0} \quad E \vdash v \Rightarrow V}{E \vdash \texttt{if}\, u = 0\, \texttt{then}\, v\, \texttt{else}\, w \Rightarrow V} \; (\texttt{if}_0) \qquad \frac{E \vdash u \Rightarrow \dot{n} \quad E \vdash w \Rightarrow V}{E \vdash \texttt{if}\, u = 0\, \texttt{then}\, v\, \texttt{else}\, w \Rightarrow V} \; (\texttt{if}_1) \quad \text{si } n \neq 0$$

Invariance du typage

$$\frac{}{\triangleright \dot{n} : \mathtt{int}}\,(Val : \mathtt{int}) \qquad \frac{\triangleright E : \Gamma \quad \Gamma, f : \forall \cdot \sigma \to \tau, x : \forall \cdot \sigma \vdash u : \tau}{\triangleright \langle \mathtt{rec}\, f(x) = u, E \rangle : \sigma \to \tau}\,(Val : Fun)$$

- $\triangleright E : \Gamma$ ssi dom $E \subseteq$ dom Γ et $\triangleright E(x) : \Gamma(x)$ pour tout $x \in$ dom E
- **Thm:** si $E \vdash u \Rightarrow V, \Gamma \vdash u{:}\boldsymbol{\tau}$ et $\triangleright E : \Gamma$

  alors $\triangleright V : \boldsymbol{\tau}$
- **Prf.** Récurrence sur la 1ère dérivation.

Tests de types

$$\frac{E \vdash u \Rightarrow V}{E \vdash uv \Rightarrow \mathsf{Wrong}}\,(App_{\mathsf{Wrong}}) \quad \text{si } V \text{ pas une clôture}$$

$$\frac{E \vdash u \Rightarrow V}{E \vdash \mathtt{if}\, u = 0\, \mathtt{then}\, v\, \mathtt{else}\, w \Rightarrow \mathsf{Wrong}}\,(\mathtt{if}_{\mathsf{Wrong}}) \quad \text{si } V \text{ pas un entier}$$

$$\frac{E \vdash u \Rightarrow V_1 \quad E \vdash v \Rightarrow V_2}{E \vdash u \dot{+} v \Rightarrow \mathsf{Wrong}}\,(\dot{+}_{\mathsf{Wrong}}1) \quad \text{si } V_1 \text{ pas un entier}$$

$$\frac{E \vdash u \Rightarrow V_1 \quad E \vdash v \Rightarrow V_2}{E \vdash u \dot{+} v \Rightarrow \mathsf{Wrong}}\,(\dot{+}_{\mathsf{Wrong}}2) \quad \text{si } V_2 \text{ pas un entier}$$

$$\frac{E \vdash u \Rightarrow V}{E \vdash \dot{-} u \Rightarrow \mathsf{Wrong}}\,(\dot{-}_{\mathsf{Wrong}}) \quad \text{si } V \text{ pas un entier}$$

Thm: si $\Gamma \vdash u{:}\boldsymbol{\tau}$ et $\triangleright\, E : \Gamma$ alors

non $(E \vdash u \Rightarrow \mathsf{Wrong})$