Principes d'induction: 1ers exemples de preuves dans le domaine du logiciel certifié

Joëlle Despeyroux

INRIA Sophia-Antipolis

Module
"Méthodes formelles et fiabilité du logiciel"

Dea d'informatique, Nice, automne 2000

Exemple: Imp

$$
\begin{array}{ll}
exp\ \ e ::= & x \mid n \mid e + e' \mid e - e' \mid \\
& true \mid false \mid e = e' \mid e > e' \mid e \wedge e' \mid e \vee e' \mid \neg e \\
stmt\ s ::= & skip \mid x := e \mid s; s' \mid \\
& if\ e\ then\ s\ else\ s' \mid while\ e\ do\ s
\end{array}
$$

sortes $\mathrm{var}, \mathrm{exp}, \mathrm{stmt}$

sous sortes $\mathrm{var} \subset \mathrm{exp}$

constructeurs

$$
\begin{array}{llll}
\mathrm{var} & : & \mathrm{Ident} & \rightarrow \mathrm{var} \\
\mathrm{int} & : & \mathrm{Integer} & \rightarrow \mathrm{exp} \\
\mathrm{true}, \mathrm{false} & : & & \rightarrow \mathrm{exp} \\
\mathrm{plus}, \mathrm{minus} & : & \mathrm{exp} \times \mathrm{exp} & \rightarrow \mathrm{exp} \\
\mathrm{eq}, \mathrm{gt}, \mathrm{and}, \mathrm{or} & : & \mathrm{exp} \times \mathrm{exp} & \rightarrow \mathrm{exp} \\
\mathrm{not} & : & \mathrm{exp} & \rightarrow \mathrm{exp} \\
\mathrm{skip} & : & & \rightarrow \mathrm{stmt} \\
\mathrm{assign} & : & \mathrm{var} \times \mathrm{exp} & \rightarrow \mathrm{stmt} \\
\mathrm{seq} & : & \mathrm{stmt} & \rightarrow \mathrm{stmt} \\
\mathrm{if} & : & \mathrm{exp} \times \mathrm{stmt} \times \mathrm{stmt} & \rightarrow \mathrm{stmt} \\
\mathrm{while} & : & \mathrm{exp} \times \mathrm{stmt} & \rightarrow \mathrm{stmt}
\end{array}
$$

Sémantique statique

$types: \quad t ::= integer \mid boolean$

typage:

$Judgement : env \vdash exp : type;\ env \vdash stmt;$

$int: \ \Gamma \vdash int\ n : integer$

$var: \ \Gamma \vdash var\ x : t \quad (\Gamma(x) = t)$

$$eq: \ \frac{\Gamma \vdash e : t \quad \Gamma \vdash e' : t}{\Gamma \vdash (e = e') : boolean}$$

$$assign: \ \frac{\Gamma \vdash x : t \quad \Gamma \vdash e : t}{\Gamma \vdash x := e}$$

$$seq: \ \frac{\Gamma \vdash s \quad \Gamma \vdash s'}{\Gamma \vdash s; s'}$$

$$if: \ \frac{\Gamma \vdash e : boolean \quad \Gamma \vdash s \quad \Gamma \vdash s'}{\Gamma \vdash if\ e\ then\ s\ else\ s'}$$

$$while: \ \frac{\Gamma \vdash e : boolean \quad \Gamma \vdash s}{\Gamma \vdash while\ e\ do\ s}$$

Sémantique dynamique

$values: \quad v ::= n \mid True \mid False$

évaluation:

$Judgement : env \vdash exp \hookrightarrow value;\ env \vdash stmt \hookrightarrow env;$

$int: \ \rho \vdash int\ n \hookrightarrow n$

$var: \ \rho \vdash var\ x \hookrightarrow v \quad (\rho(x) = v)$

$$eq_t: \ \frac{\rho \vdash e \hookrightarrow v \quad \rho \vdash e' \hookrightarrow v}{\rho \vdash e = e' \hookrightarrow True}$$

$$eq_f: \ \frac{\rho \vdash e \hookrightarrow v \quad \rho \vdash e' \hookrightarrow v'}{\rho \vdash e = e' \hookrightarrow False} \quad v \neq v'$$

$skip: \ \rho \vdash skip \hookrightarrow \rho$

$$assign: \ \frac{\rho \vdash e \hookrightarrow v}{\rho \vdash x := e \hookrightarrow \rho[x = v]}$$

$$seq: \ \frac{\rho \vdash s \hookrightarrow \rho' \quad \rho' \vdash s' \hookrightarrow \rho''}{\rho \vdash s; s' \hookrightarrow \rho''}$$

Sémantique dynamique (suite)

$$if_t : \frac{\rho \vdash e \hookrightarrow True \quad \rho \vdash s \hookrightarrow \rho'}{\rho \vdash if\ e\ then\ s\ else\ s' \hookrightarrow \rho'}$$

$$if_f : \frac{\rho \vdash e \hookrightarrow False \quad \rho \vdash s' \hookrightarrow \rho'}{\rho \vdash if\ e\ then\ s\ else\ s' \hookrightarrow \rho'}$$

$$while_t : \frac{\rho \vdash e \hookrightarrow True \quad \rho \vdash s \hookrightarrow \rho' \quad \rho' \vdash while\ e\ do\ s \hookrightarrow \rho''}{\rho \vdash while\ e\ do\ s \hookrightarrow \rho''}$$

$$while_f : \frac{\rho \vdash e \hookrightarrow False}{\rho \vdash while\ e\ do\ s \hookrightarrow \rho}$$

Programmes infinis

Def. Les *formes normales approximatives* d'un langage sont les valeurs auxquelles on ajoute $\perp$.

Def. Ordre partiel sur les formes normales approximatives: $\forall t, t \leq \perp,\ \ c_i \leq c_j \Rightarrow i = j,\ \ (t, u) \leq (t', u') \Rightarrow t \leq t'$ et $u \leq u'$.

Extension de la sémantique: On ajoute un axiome de la forme: $\rho \vdash e \hookrightarrow \perp$, par ex, pour l'évaluation de Imp.

Def. α est une *forme normale approximative* d'un terme t ssi

– α est une forme normale approximative et

– $\vdash t \hookrightarrow \alpha$ est valide.

Thm à démontrer (non-déterminisme apparent):

$\vdash p \hookrightarrow \alpha\ \&\ \vdash p \hookrightarrow \beta \Rightarrow \exists\gamma,\ \gamma \succeq \alpha,\ \gamma \succeq \beta\ s.t.\ \vdash p \hookrightarrow \gamma$

Corollaire: La suite α_n tq $\vdash p \hookrightarrow \alpha_n$ admet une limite.

Une preuve très simple en SN

Thm : $s; skip \equiv s$ i.e. $(\forall \rho, \rho', s.\ \rho \vdash s; skip \hookrightarrow \rho' \Leftrightarrow \rho \vdash s \hookrightarrow \rho')$

Inversion de la règle *seq* : $\dfrac{\rho \vdash s \hookrightarrow \rho' \quad \rho' \vdash s' \hookrightarrow \rho''}{\rho \vdash s; s' \hookrightarrow \rho''}$

Lemme *inv_seq* :
$\forall \rho, \rho'', s.\ (\rho \vdash s; s' \hookrightarrow \rho'')\ \ \exists \rho'.(\rho \vdash s \hookrightarrow \rho')\ \&\ (\rho' \vdash s' \hookrightarrow \rho'')$.
Preuve.immédiate: il y a une seule règle applicable à *seq*: *seq*.

Preuve du thm:

$(\Rightarrow)$

Hypothèse: $\rho \vdash s; skip \hookrightarrow \rho'\ \ [H]$.

Preuve:

Par le lemme *inv_seq*, appliqué à H, on a:
$\exists \rho_1.(\rho \vdash s \hookrightarrow \rho_1)\ \ [H_1]\ \&\ (\rho_1 \vdash skip \hookrightarrow \rho')\ \ [H_2]$.

La seule règle applicable à *skip* est *skip*, d'où $\rho' = \rho_1\ \ [H_3]$.

D'où la conclusion $\rho \vdash s \hookrightarrow \rho'$, par H_1 et H_3.

$(\Leftarrow)$

Hypothèse: $\rho \vdash s \hookrightarrow \rho'\ \ [H]$.

Preuve:

La conclusion $\rho \vdash s; skip \hookrightarrow \rho'$ est immédiate par application de la règle *seq* à H et à l'instance de la règle *skip*:
$\rho' \vdash skip \hookrightarrow \rho'$.

La même preuve présentée différemment

Thm : $s; skip \equiv s$
i.e. $(\forall \rho, \rho', s.\ \rho \vdash s; skip \hookrightarrow \rho' \Leftrightarrow \rho \vdash s \hookrightarrow \rho')$

Preuve:

$(\Rightarrow)$

Hypothèse: $\rho \vdash s; skip \hookrightarrow \rho'$.

Preuve:

$$\dfrac{\rho \vdash s \hookrightarrow \rho_1 \quad \rho_1 \vdash skip \hookrightarrow \rho' = \rho_1}{\rho \vdash s; skip \hookrightarrow \rho'}$$

$\Rightarrow \rho \vdash s \hookrightarrow \rho'$

$(\Leftarrow)$

Hypothèse: $\rho \vdash s \hookrightarrow \rho'$.

Preuve:

$$\dfrac{\rho \vdash s \hookrightarrow \rho' \quad \rho' \vdash skip \hookrightarrow \rho'}{\rho \vdash s; skip \hookrightarrow \rho'}$$

La présentation de la preuve est plus concise, mais difficile à relire. Idéale pour faire des preuves sur le brouillon. La forme précédente (forme textuelle) servant de document de référence.

Induction mathématique

Un entier est:

- 0
- S d'un entier

  et peut seulement être obtenu par cette définition récursive.

Soit P une propriété sur les entiers.

Principe d'induction mathématique sur les entiers:

$$\begin{array}{l}(P(0)\ \& \\ (\forall m \in nat.\ P(m) \Rightarrow P(S(m))) \\ \Rightarrow \forall n \in nat.\ P(n)\end{array}$$

Cas particulier $(P(m) = \forall k < m.Q(k))$:

$$\begin{array}{l}(\forall m \in nat.\ (\forall k < m.\ Q(k)) \Rightarrow Q(m)) \\ \Rightarrow \forall n \in nat.\ Q(n)\end{array}$$

Induction structurelle

C'est l'induction sur la structure d'une expression.

Exemple:

$exp \quad e \ ::= x \mid e + e' \mid e * e'$

Soit P une propriété sur exp.

Principe d'induction structurelle sur exp:

$$\begin{array}{l}(\forall x \in var.\ P(x))\ \& \\ (\forall e, e' \in exp.\ P(e)\ \&\ P(e') \Rightarrow P(e + e'))\ \& \\ (\forall e, e' \in exp.\ P(e)\ \&\ P(e') \Rightarrow P(e * e')) \\ \Rightarrow \forall e \in exp.\ P(e)\end{array}$$

Une preuve par induction structurelle

Thm. La sémantique dynamique de Imp\while est déterministe.

Où Imp\while est Imp moins le constructeur while.

$$\forall \rho, e, v, v'.\ (\rho \vdash e \hookrightarrow v)\ \&\ (\rho \vdash e \hookrightarrow v') \Rightarrow v = v'$$

Preuve par induction sur la structure de e.

$$P(e) = \forall \rho, v, v'.\ (\rho \vdash e \hookrightarrow v)\ \&\ (\rho \vdash e \hookrightarrow v') \Rightarrow v = v'$$

Cas $s; s'$.

Hypothèses: $(\rho \vdash s; s' \hookrightarrow \rho'')$ $[H]$ et $(\rho \vdash s; s' \hookrightarrow \rho''_1)$ $[H']$.

Preuve:

Par inversion de la règle *seq*, appliquée à H et H', on a:
$\exists \rho'.\ tq.\ (\rho \vdash s \hookrightarrow \rho')$ $[H_1]$ & $(\rho' \vdash s' \hookrightarrow \rho'')$ $[H_2]$ et

$\exists \rho'_1.\ tq.\ (\rho \vdash s \hookrightarrow \rho'_1)$ $[H'_1]$ & $(\rho'_1 \vdash s' \hookrightarrow \rho''_1)$ $[H'_2]$.

Par hypothèse d'induction sur s $(P(s))$ appliquée à H_1 et H'_1, on obtient: $\rho' = \rho'_1$.

Par hypothèse d'induction sur s' $(P(s'))$ appliquée à H_2 et H'_2, on obtient la conclusion: $\rho'' = \rho''_1$.

[...]

Cas while. Il faut utiliser un principe d'induction plus fort: l'induction sur la structure d'une preuve.

Induction bien fondée

Def. Une **relation bien fondée** est une relation binaire $<$ sur un ensemble A, tq il n'existe pas de chaîne infinie décroîssante $\cdots < a_i < \cdots < a_1 < a_0$.
Alternativement, on peut la définir par la propriété:

Prop. Soit $<$ une relation binaire sur un ensemble A. $<$ est une relation bien fondée ssi tout sous-ensemble Q de A admet un élément minimum, i.e. un n tel que: $\forall m \in Q,\ m < n \Rightarrow m \notin Q$.
Principe d'induction sur une relation bien fondée:

Def. Soit $<$ une relation bien fondée sur un ensemble A. Soit P une propriété sur les éléments de A. On a:
$(\forall x \in A.\ (\forall y \in A.\ y < x \Rightarrow P(y)) \Rightarrow P(x)) \equiv \forall a \in A.\ P(a)$.
Exemples:

- l'induction mathématique est une induction bien fondée sur l'ensemble des entiers, avec la relation successeur.
- l'induction sur une expression d'un langage L est l'induction bien fondée sur l'ensemble des expressions de L, muni de la relation "est une sous-expression de".

Pour montrer qu'un programme termine, on peut montrer que son exécution fait décroître une valeur dans un ensemble bien fondé (i.e. muni d'une relation bien fondée).

Induction sur la structure d'une preuve

"Induction on derivations" en anglais.
Exemple:

Set *eval* is
Judgement : $env \vdash exp \hookrightarrow value$
int : $\rho \vdash int\ n \hookrightarrow n$
var : $\rho \vdash var\ x \hookrightarrow v \quad (\rho(x) = v)$

$$plus : \frac{\rho \vdash e \hookrightarrow n \quad \rho \vdash e' \hookrightarrow n'}{\rho \vdash e + e' \hookrightarrow m} \ (n + n') = m$$

end *eval*

Une preuve p de $(\rho \vdash e \hookrightarrow v)$, notée $p : (\rho \vdash e \hookrightarrow v)$, est:

- une instance de l'axiome *int*: $\rho \vdash int\ n \hookrightarrow n$
- une instance de l'axiome *var*: $\rho \vdash var\ x \hookrightarrow v \quad (\rho(x) = v)$
- un arbre de preuve de la forme:

$$\frac{\overset{p_1}{\rho \vdash e \hookrightarrow n} \qquad \overset{p_2}{\rho \vdash e' \hookrightarrow n'}}{\rho \vdash e + e' \hookrightarrow m} \ (n + n') = m$$

  où $p_1 : (\rho \vdash e \hookrightarrow n)$ et $p_2 : (\rho \vdash e' \hookrightarrow n')$
  et peut seulement être obtenue par cette définition récursive.

Induction sur la structure d'une preuve (suite)

Soit P une propriété sur les preuves p de $(\rho \vdash e \hookrightarrow v)$.

Principe d'induction sur la structure de la preuve p:

$$(\forall p\ (\forall p'\ \textit{sous preuve de}\ p.\ P(p') \Rightarrow P(p)))$$
$$\Rightarrow \forall p\ P(p)$$

Exemple, pour *eval*:

$(\forall \rho, n.\ P(p : (\rho \vdash int\ n \hookrightarrow n)))$ &
$(\forall \rho, x, v\ tq\ \rho(x) = v.\ P(p : (\rho \vdash var\ x \hookrightarrow v)))$ &
$(\forall \rho, e, n, m.\ \forall p_1 : (\rho \vdash e \hookrightarrow n).\ \forall e', n'.\ \forall p_2 : (\rho \vdash e' \hookrightarrow n').$
$\quad P(p_1)\ \&\ P(p_2)\ \&\ (n + n') = m \Rightarrow P(p : (\rho \vdash e + e' \hookrightarrow m)))$
$\Rightarrow \forall p : (\rho \vdash e \hookrightarrow v).\ P(p)$

Une preuve par induction sur la preuve

Thm. La sémantique dynamique de Imp est déterministe.
$\forall \rho, v, v'.\ (\rho \vdash e \hookrightarrow v)\ \&\ (\rho \vdash e \hookrightarrow v') \Rightarrow v = v'$

Preuve par (double) induction sur la structure de la preuve des hypotheses $(\rho \vdash e \hookrightarrow v)$ et $(\rho \vdash e \hookrightarrow v')$.

$P(p, p') = \forall \rho, v, v'.\ p : (\rho \vdash e \hookrightarrow v)\ \&\ p' : (\rho \vdash e \hookrightarrow v') \Rightarrow v = v'$

Cas seq.

Hypothèses:

$$\frac{\rho \vdash s \hookrightarrow \rho' \quad \rho' \vdash s' \hookrightarrow \rho''}{\rho \vdash s; s' \hookrightarrow \rho''}\ \&\ \frac{\rho \vdash s \hookrightarrow \rho'_1 \quad \rho'_1 \vdash s' \hookrightarrow \rho''_1}{\rho \vdash s; s' \hookrightarrow \rho''_1}$$

Preuve:

Par hypothèse d'induction sur les preuves de $(\rho \vdash s \hookrightarrow \rho')$ et de $(\rho \vdash s \hookrightarrow \rho'_1)$, on a: $\rho' = \rho'_1$.

Par hypothèse d'induction sur les preuves de $(\rho' \vdash s' \hookrightarrow \rho'')$ et de $(\rho'_1 \vdash s' \hookrightarrow \rho''_1)$, on a: $\rho'' = \rho''_1$.

Cas eq_t/eq_f.

Hypothèses:

$$\frac{\rho \vdash e \hookrightarrow v \quad \rho \vdash e' \hookrightarrow v}{\rho \vdash e = e' \hookrightarrow True}\ \&\ \frac{\rho \vdash e \hookrightarrow v \quad \rho \vdash e' \hookrightarrow v'}{\rho \vdash e = e' \hookrightarrow False}\quad v \neq v'$$

Preuve:

Par hypothèse d'induction sur les preuves de $(\rho \vdash e' \hookrightarrow v)$ et de $(\rho \vdash e' \hookrightarrow v')$, on a: $v = v'$. Contradiction.

Une preuve par induction sur la preuve (suite)

Cas $while_t/while_t$.

Hypothèses:

$$\frac{\rho \vdash e \hookrightarrow True \quad \rho \vdash s \hookrightarrow \rho' \quad \rho' \vdash while\ e\ do\ s \hookrightarrow \rho''}{\rho \vdash while\ e\ do\ s \hookrightarrow \rho''}\ \&$$

$$\frac{\rho \vdash e \hookrightarrow True \quad \rho \vdash s \hookrightarrow \rho'_1 \quad \rho'_1 \vdash while\ e\ do\ s \hookrightarrow \rho''_1}{\rho \vdash while\ e\ do\ s \hookrightarrow \rho''_1}$$

Preuve:

Par hypothèse d'induction sur les preuves de $(\rho \vdash s \hookrightarrow \rho')$ et de $(\rho \vdash s \hookrightarrow \rho'_1)$, on a: $\rho' = \rho'_1$.

Par hypothèse d'induction sur les preuves de $(\rho' \vdash while\ e\ do\ s \hookrightarrow \rho'')$ et de $(\rho'_1 \vdash while\ e\ do\ s \hookrightarrow \rho''_1)$, on a: $\rho'' = \rho''_1$.

[...]

Preuve d'une règle d'inférence

$$\text{Thm.}\ \frac{H_1 \cdots H_n}{C}$$

On prouve la validité de la règle d'inférence par induction sur la structure des preuves des hypothèses.
Pour chaque preuve des hypothèses H_i, il faut exiber une preuve de la conclusion C.

Exemple:
Pour prouver la correction de la compilation d'un langage source l dans un langage cible (par exemple un langage machine) l', on devra prouver deux règles d'inférences du type:

$$\frac{\rho \vdash^{l} e \hookrightarrow v \qquad \overline{\rho} \vdash^{ll'} e \hookrightarrow c}{\vec{\rho}, s \vdash^{l'} c \hookrightarrow t(v), s}$$

$$\frac{\overline{\rho} \vdash^{ll'} e \hookrightarrow c \qquad \vec{\rho}, s \vdash^{l'} c \hookrightarrow t(v), s}{\rho \vdash^{l} e \hookrightarrow v}$$

où les systèmes l, l' et ll' spécifient respectivement l'exécution des langages l et l' et le compilateur.
t spécifie la traduction des valeurs du language source dans les valeurs du langage cible.
Les environnements ρ, $\overline{\rho}$ et $\vec{\rho}$ se correspondent d'une manière plus ou moins simple selon les langages.