


Vérification formelle d'algorithmes probabilistes

Tahina Ramananandro

27 juin - 25 août 2005

sous la direction de MM. Philippe Audebaud et Laurent Théry

(INRIA Marelle - Sophia Antipolis)

Sommaire

- Représentation d'un algorithme probabiliste
  - Les données aléa
- Mécanisation des preuves
  - Du programme à la formule Coq
  - De la formule à la preuve
- Application : simulations de lois simples
  - Loi géométrique

Représentation d'un algorithme probabiliste

Les données aléatoires

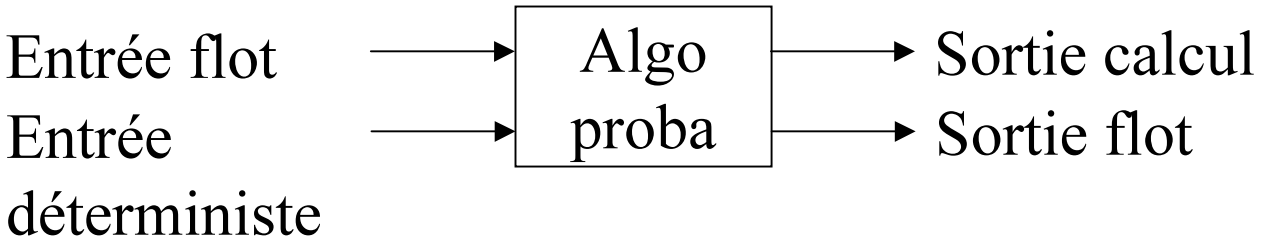
Données aléatoires

- Approche impérative
  - aléa = appel à un générateur aléatoire
  - appels 2 à 2 indépendants
  - données aléa toutes de même type

- Approche fonctionnelle
  - aléa = flot = suite de variables aléatoires
  - indépendantes
  - et identiquement distribuées

Algo proba avec les flots

- Algo proba = fonction déterministe de ses entrées déterministes **et du flot d’entrée** et retournant le flot de sortie après extractions et sa valeur « aléatoire »



- Pb : fonction peut utiliser flot sans extraire
  - N’apparaît pas dans le langage étudié

Représentation d'un algorithme probabiliste

Le langage λ_O

Langage fonctionnel proba : λ_O (Park, Pfenning, Thrun 2005)

- Termes : évaluation ne dépend pas des flots.
  - *Terme* ::= *vb* | *var* | **true** | **false** | **if** *Terme* **then** *Terme* **else** *Terme*
    | (*Terme, Terme*) | **fst** *Terme* | **snd** *Terme*
    | λ*var. Terme* | *Terme Terme* | **fix** *var* := *Terme*
    | **prob** *Expr* (distribution de probabilité abstraite)
- Expressions : calcul dépend des flots.
  - *Expr* ::= *Terme* | $\mathscr{S}$ (extraction d'un élément de flot)
    | **sample** *var* **from** *Terme* **in** *Expr* (échantillonnage de la distri de proba *Terme* pour utilisation dans *Expr*)

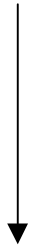
Réduction

- Sémantique à petits pas
  - Termes : appel par valeur, *évaluation* ne dépend pas du flot
  - Expressions : *calcul* dépend du flot

- Valeurs :
  - *Valeur ::= vb* | **true** | **false** | (*Valeur, Valeur*)
    | λ*var. Terme* | **prob** *Expr*

Calcul : extraction

$\mathscr{S}$ [flot x :: t]



x [flot t]

Calcul : SAMPLE

SAMPLE x FROM **m** IN e

SAMPLE x FROM PROB **ex** IN e [flot **s**]

SAMPLE x FROM PROB v IN e [flot t]

e [x := v] [flot **t**]

Calcul

- Le flot intervient
  - Sauf pour les termes, pour lesquels on utilise l’évaluation (par appel par valeur)
- A flot fixé :
  - Calcul déterministe
  - Au plus une expression peut être obtenue en un pas de calcul
- Le flot résultant est une queue du flot d’entrée, inutilisée par le programme

Exemple

- Flots de booléens
- Flot d'entrée : T, F, T, T, F, ...

  sample *x* **from prob** $\mathscr{S}$ **in** x = true
  sample *y* **from prob** $\mathscr{S}$ **in** y = false
  sample *z* **from if** *x* = *y* **then prob S else prob false in** z = false
  if *z* **then** *x* **else** *y*

- Résultat : false

Exemple

- Flots de booléens
- Flot d'entrée : T, T, F, T, F...

sample *x* **from prob** $\mathcal{S}$ **in** x = true
sample *y* **from prob** $\mathcal{S}$ **in** y = true
sample *z* **from if** *x* = *y* **then prob S else prob false in** z = false
if *z* **then** *x* **else** *y*

- Résultat : true

Modélisation d'un algo par un terme du λ_O

λp**. prob sample** x **from prob** $\mathscr{S}$ **in** $x \leq p$

- « Demander » les paramètres déterministes par des abstractions
  - L 'algo = corps le plus interne
  - Passage des paramètres par application
- Algorithme probabiliste : corps = terme **prob**
  - Mis en œuvre par un **sample** :
    - *expression*
    - *calcul dépend du flot*

Mécanisation des preuves

sur les algorithmes probabilistes

Raisonnement sur des programmes en λ_O

- On veut pouvoir exprimer et prouver des propriétés sur un programme.
- Exemple : loi de Bernoulli

  $\lambda p.$ **prob sample** x **from prob** $\mathscr{S}$ **in** $x \leq p$

  - On veut montrer que pour tout p, la probabilité que l'échantillonnage renvoie **true** soit p
- Problèmes :
  - Exprimer de telles propriétés en logique
  - Raisonner proba en Coq

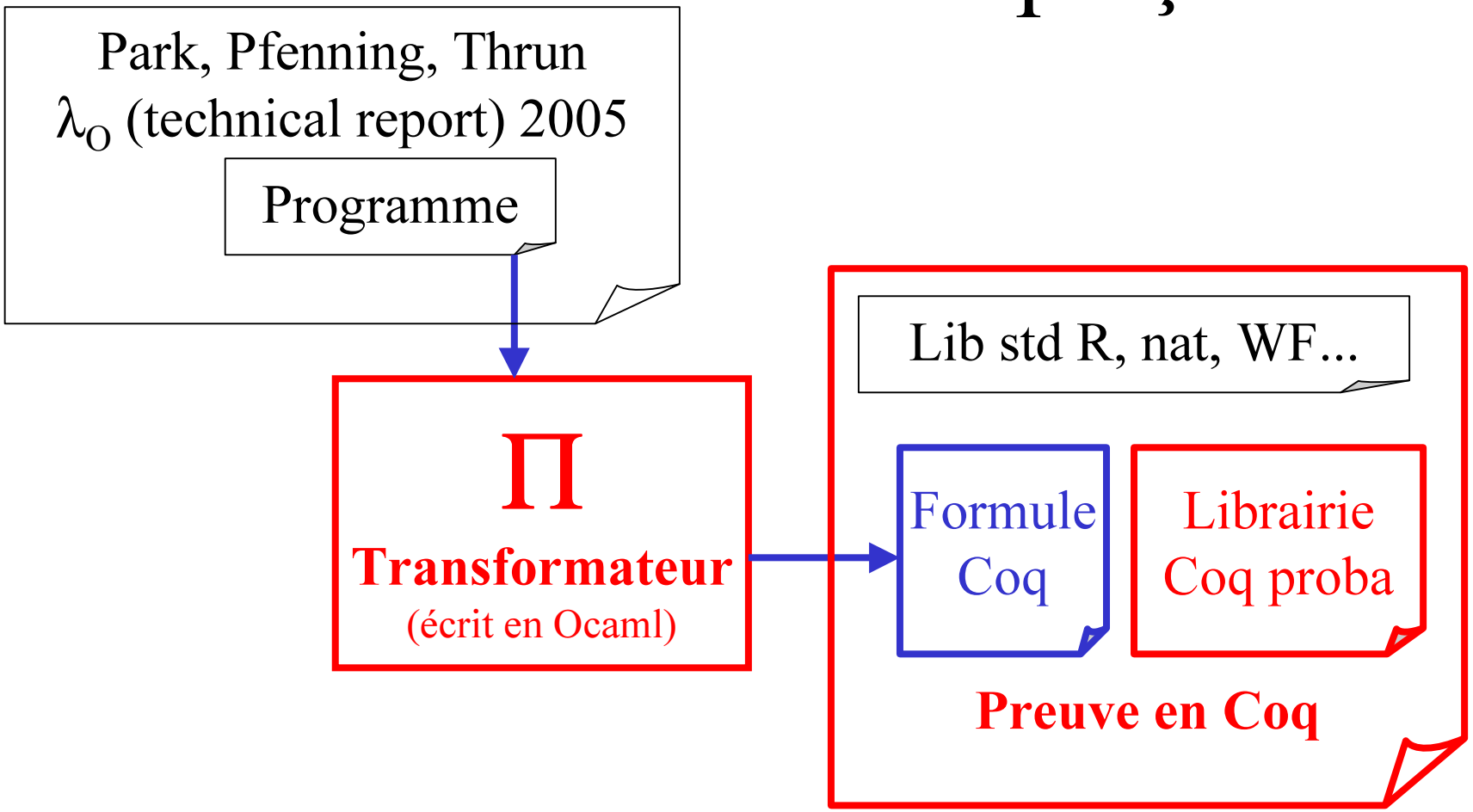
Aperçu

Park, Pfenning, Thrun
λ_O (technical report) 2005

Programme

Lib std R, nat, WF...

Aperçu


Park, Pfenning, Thrun
λ_O (technical report) 2005
Programme
Π
Transformateur
(écrit en Ocaml)
Lib std R, nat, WF...
Formule Coq
Librairie Coq proba
Preuve en Coq

Mécanisation des preuves

Du programme à la formule

Formules associées à un programme

- A chaque structure du langage on associe une formule logique (approche PROLOG)
  - Suivre la sémantique opérationnelle
  - On obtient une formule globale pour tout le programme

- Proba n'apparaît que dans le raisonnement sur les propriétés obtenues

Terminaison de points fixes

- Point fixe déterministe
  - fonction à un argument strictement décroissant selon un ordre bien fondé à préciser (exemple : *factorielle*)
- Point fixe probabiliste : **prob** récursif
  - exemple : « repeat *res* := *a* until *c*; return *res* »

```
fix boucle. prob
  sample res from prob a in
    sample y from
      if c res then prob res else boucle
    in y
```

Point fixe proba

- Idée : point fixe proba = point fixe déterministe sur le flot, rappel avec un sous-flot strictement plus petit suivant un ordre bien fondé
  - Préciser l'ordre bien fondé
  - Mais aussi un ensemble sur lequel on sait que l'algorithme termine
- Alors, le raisonnement probabiliste n'apparaît que plus tard.

Formule pour un point fixe

- Les deux cas se traitent de manière similaire.
  - Soit n le variant (argument récursif ou flot). Soit $<$ un ordre sur les variants. Alors, si l'algorithme termine pour les $n' < n$, alors il termine pour n.
  - Exhiber en Coq la preuve de bonne fondation de $<$ au moment de prouver que l'algorithme termine.
- Formule de correction : celle du corps

Mécanisation des preuves

De la formule à la preuve

Structure d'un raisonnement proba en Coq

- Déclaration des variables libres.
- *Hypothèses (et éventuellement lemmes) sur les variables libres*
- Hypothèse : formule générée par le transformateur Π.
- *Preuve de terminaison.*
- *Preuve de mesurabilité.*
- *Preuve finale (calcul proprement dit).*

Espace des flots

- Soit un espace de proba (E, p) sur les éléments.
- Tribu sur les flots : sigma-algèbre engendrée par les rectangles (produits finis d'ensembles E-mesurables).
- Mesure d'un rectangle : produit des p-mesures de ses ensembles constitutifs

Calculs de proba en Coq

- Soit calculer la mesure d'ensembles de flots d'entrée de tout le programme (proba globale)
  - On peut se ramener à un mesurable connu
  - On utilise aussi toutes les propriétés de la mesure (notamment l'additivité)

Calculs de proba en Coq

- Soit utiliser des formules de proba locales : calcul de proba sur un flot intermédiaire
  - Quand on utilise une loi auxiliaire
  - Valable que si l'on ne dispose d'aucune propriété sur ce flot intermédiaire
- Axiomes :
  - Probas locales de deux propriétés équivalentes sont égales.
  - Proba locale d'un événement est sa mesure

Applications :
simulations de lois simples

Cadre des simulations

- Types de base : réels, naturels (en plus des booléens)
  - avec les opérations arithmétiques élémentaires et même des fonctions comme Ln, Exp...
- Flots de réels distribués uniformément sur [0, 1]
  - Démonstrations en général plus simples
  - sauf peut-être au point zéro

Loi géométrique

```
λp.
  let berp = bernoulli p in
    fix geomp. prob
      sample b from berp in
        sample y from
          if b then
            prob 0
          else
            prob
              sample g from geomp in g + 1
        in y
```

Loi géométrique

- Récursivité probabiliste : variant flot.
  - Toute opération sur le flot doit être connue, même pour *bernoulli*.
  - Impossible d'utiliser des probas locales.

- Hypothèse *bernoulli* :
  - *bernoulli p* consomme 1 élément et renvoie **true** ssi $\leq p$.

Loi géométrique

- L'échantillonnage termine Si le flot est dans cet ensemble G :
  - on tombe sur un élément $\leq p$ en un nb fini d'extractions
- Définir un ordre < bien fondé sur cet ensemble tel que :
  - si un flot de cet ensemble commence par un élément $> p$, il est plus grand que sa queue

Loi géométrique

- Pour tout n, échantillon n proba « géométrique » $(1 - p)^n p$.
  - On se ramène aux G_n : ensembles des flots commençant par n éléments $>p$ puis un $\leq p$.
  - Ils sont mesurables
  - Proba par induction sur n

Conclusion

Conclusion

- Raisonnement sur les algo proba = raisonnement déterministe à aléa fixé + proba sur les aléas fournis
  - Aléa abstrait par défaut est rendu concret
- Approche λ_O assez nouvelle
  - Algo proba surtout implantés en impératif
- Problèmes majeurs
  - Optimisations possibles pour la formule Π.
  - Automatisation du raisonnement Coq avec Ltac
  - Approche détachée de Why : localité ?
  - Extraction ???

Remerciements (ordre aléatoire)

- MM. Laurent Théry et Philippe Audebaud
  - Pour leur soutien sans faille.
- Tous les membres des équipes Marelle et Everest
  - Pour l'ambiance chaleureuse ici à Sophia.
- Ma famille, bien sûr !
  - Dont ma mère, qui est venue deux semaines exprès à Antibes
  - ma tante et mes cousins de Digne-les-Bains.