Preuve d'algorithmes probabilistes

Christine Paulin

AVERROES-16 sept 2003/21 nov 2003

Plan

- Motivations
- L'approche de Joe Hurd avec HOL
  - Transformation monadique
  - Exemple de la primalité
  - Terminaison probabiliste
- L'approche transformation de prédicats

Prouver des algorithmes probabilistes

Approche classique

$$\{\text{Précondition}\} \quad \text{programme } \{\text{Postcondition}\}$$

Approche probabiliste

La post-condition est réalisée avec une certaine probabilité q.

Correction partielle

Si le programme répond vrai, la propriété P est satisfaite, sinon la propriété P est fausse avec probabilité q.

Vers une preuve mathématique ?

- Modélisation du programme :
  Tirage aléatoire d'une pièce, `random`,
  distribution uniforme sur $[0, 1]$ ...
- Espace de probabilité :
  Indépendance implicite des tirages (espace produit),
  le nombre de tirages, la distribution d'un tirage peut dépendre de l'exécution.
- Choix des propriétés à établir.

L'approche de Joe Hurd avec HOL

Références

– *Formal Verification of Probabilistic Algorithms*. PhD thesis, University of Cambridge, 2002.

– *Verification of the Miller-Rabin Probabilistic Primaly test*. J. of Logic and Algebraic Programming, special issue on Probabilistic Techniques for the Design and Analysis of Systems, Elsevier Science, 2003.

– *A formal approach to probabilistic termination.* In *TPHOLs'02*, 2002.

Programmes probabilistes dans HOL

Principe

Programme déterministe qui accède une suite infinie de tirages aléatoires de valeurs booléennes.

$$\mathbb{B}^\infty : \text{ensemble des suites infinies de booléens}$$

Un programme aléatoire prend en argument une suite $s \in \mathbb{B}^\infty$, consomme un segment initial de s pour calculer son résultat et renvoie la partie de la séquence non utilisée.

– Variable aléatoire $X : \alpha$ ↔ Fonction $\hat{X} : \mathbb{B}^\infty \to \alpha \times \mathbb{B}^\infty$
– Programme p de type $\alpha \to \beta$ ↔ Fonction $\hat{p} : \alpha \to \mathbb{B}^\infty \to \beta \times \mathbb{B}^\infty$

Transformation monadique

La suite infinie peut être vue comme une variable globale, transformée par effet de bord.

$$
\begin{aligned}
\hat{\alpha} &= \mathbb{B}^{\infty} \to \alpha \times \mathbb{B}^{\infty} \\
\texttt{unit} &: \alpha \to \hat{\alpha} \\
&= \textbf{fun}\ \texttt{a} \to \textbf{fun}\ \texttt{s} \to \texttt{(a,s)} \\
\texttt{bind} &: \hat{\alpha} \to (\alpha \to \hat{\beta}) \to \hat{\beta} \\
&= \textbf{fun}\ \texttt{x f} \to \textbf{fun}\ \texttt{s} \to \texttt{let (a,s')=x s in g a s'} \\
\texttt{flip} &: \widehat{\texttt{bool}} \\
&= \textbf{fun}\ \texttt{s} \to \texttt{(hd s,tl s)}
\end{aligned}
$$

Exemple

Programme qui tire n fois à pile ou face et renvoie le nombre de pile (valeur vraie).

```
let rec nbp n = if n=0 then 0
else let p = nbp (n-1) in if flip() then p+1 else p
```

```
let rec nbp n s = if n=0 then (0,s)
else let (p,s1) = nbp (n-1) s in let (b,s2) = flip s1
     in if b then (p+1,s2) else (p,s2)
```

```
let rec nbp n = if n=0 then unit 0
else bind (nbp (n-1))
    (fun p → bind flip
             (fun b → if b then unit (p+1) else unit p))
```

Théorie des probabilités

- Formalisation dans HOL
- Théorie de la mesure $\mathsf{Pr} : \mathcal{P}(\mathbb{B}^\infty) \to \mathbb{R}$
- Fonction partielle sur un ensemble $\mathcal{E}$ d'évènements clos par complément et union finie.
- Propriétés
  - $A, B \in \mathcal{E}, A \cap B = \emptyset \Rightarrow \mathsf{Pr}(A \cup B) = \mathsf{Pr}(A) + \mathsf{Pr}(B)$
  - $\mathsf{Pr}(\{s \mid \mathtt{fst}\ (\mathtt{nbp}\ n\ s) = r\}) = \begin{pmatrix} n \\ r \end{pmatrix} \frac{1}{2^n}$
- Abréviations

  $\forall^* s.\phi(s)$: ϕ presque sûrement, $\{s \mid \phi(s)\} \in \mathcal{E} \wedge \mathsf{Pr}(\{s \mid \phi(s)\}) = 1$

Programme probabiliste dans HOL

Un programme probabiliste `p` qui calcule une valeur de type α est modélisé dans HOL comme une fonction $\hat{p}$ de type

$$\hat{\alpha} = \mathbb{B}^{\infty} \to \alpha \times \mathbb{B}^{\infty}$$

La probabilité que `p` satisfasse une propriété Q est donnée par :

$$\mathsf{Pr}(\{s \mid Q(\texttt{fst}\ (\hat{p}\ s))\})$$

Condition pour que $f : \hat{\alpha}$ représente un programme probabiliste :

- $\{s \mid Q(\texttt{fst}\ (\hat{p}\ s))\}$ est mesurable $\in \mathcal{E}$
- Indépendance : la suite de valeurs booléennes est consommée
- Indépendance forte préservée par les opérations monadiques.

Test de primalité de p

- $p - 1 = 2^k.u \quad 1 < a < p$
- `test` $p\ a =$
  $a^{2^k u} \equiv 1(p) \wedge \forall j.0 \leq j \leq k.a^{2^j u} \equiv 1(p) \Rightarrow a^{2^{j-1} u} \equiv +/-1(p)$
- si p premier : $\forall a.0 < a < n \Rightarrow$ `test` $p\ a$
- si p est composite alors $|\{a|(\texttt{test}\ p\ a)\}| \leq \frac{(p-1)}{2}$

Program :

```
let rec miller_rabin p n =
if n = 0 then true
else let a = random (p-1) in
if test p a then miller_rabbin p (n-1) else false
```

Preuve dans HOL

- Théorie des groupes.
- Utilisation d'un générateur approché à la place de `random` pour garder une fonction totale.
- p est premier : $\forall n, s.\mathtt{miller\_rabin}\ p\ n\ s = \mathtt{true}$
- p est composé : $\forall t, s.\mathsf{Pr}(\mathtt{miller\_rabin}\ p\ n\ s = \mathtt{false}) \geq 1 - \frac{1}{2^n}$

Terminaison

Les programmes modélisées à l'aide des opérateurs monadiques sont transformés en fonctions totales.

Une fonction totale de type $\hat{\alpha}$ n'utilise qu'un nombre borné de tirages booléens, la probabilité de chaque valeur est de la forme $\frac{m}{2^n}$.

De nombreux programmes probabilistes ne peuvent pas s'exprimer ainsi : `random`(3) (probabilité $\frac{1}{3}$).

Modélisation de fonctions qui terminent avec probabilités 1.

Principe de la construction

Boucle probabiliste avec condition déterministe :

$$b : \alpha \to \mathbb{B}^{\infty} \to \alpha \times \mathbb{B}^{\infty} \qquad c : \alpha \to \mathbb{B}$$

Itération bornée :

```
let rec iter n a = if n=0 then unit a
else if c a then bind (b a) (iter (n-1)) else unit a
```

Arrêt : `stop n a= not (c (iter n a))`

Boucle :

```
let while a = if ∃n.stop n a then iter (μn.stop n a) a
              else arb
```

Raisonner sur les boucles

Propriétés

```
let while a = if c a then bind (b a) while else unit a
```

Terminaison presque sûre :

$$\forall a. \forall^* s. \exists n. (\texttt{stop}\ n\ a\ s)$$

La propriété de terminaison presque sûre permet d'établir l'indépendance forte de la fonction `while`.

Autres conditions de terminaison

Propriété équivalente :

$$\exists p > 0. \forall a. \mathsf{Pr}(\{s \mid \exists n. (\mathtt{stop}\ n\ a\ s)\} \geq p$$

Condition suffisante (variant f borné par N)

$$\exists p > 0. \forall a. c\ a \Rightarrow (f\ a) < N \wedge \mathsf{Pr}(\{s \mid f\ (\mathtt{fst}\ (b\ a\ s)) < f\ a\}) \geq p$$

Exemple : Bernouilli (p)

Tirer vrai avec probabilité p et faux avec probabilité $1 - p$.
L'idée est de considérer la séquence aléatoire $s = (s_i)_{i \geq 1}$ comme l'écriture binaire du nombre $\Sigma_{i=1}^{\infty} \frac{s_i}{2^i}$, ce nombre est inférieur à p avec probabilité p.

$$0 \leq \Sigma_{i=1}^{\infty} \frac{s_i}{2^i} \leq 1$$

$$\begin{array}{rcl} 0 + \Sigma_{i=2}^{\infty} \frac{s_i}{2^i} \leq p & \Leftrightarrow & \Sigma_{i=1}^{\infty} \frac{s_{i+1}}{2^i} \leq 2 * p \\ \frac{1}{2} + \Sigma_{i=2}^{\infty} \frac{s_i}{2^i} \leq p & \Leftrightarrow & \Sigma_{i=1}^{\infty} \frac{s_{i+1}}{2^i} \leq 2 * p - 1 \end{array}$$

Code

```
let rec bernouilli p =
let x = flip () in
 if p < 1/2
 then
    if x then false else bernouilli (2×p)
 else
    if x then bernouilli (2×p-1) else true
```

Itération avec α = `bool+int`

Une chance sur deux de terminer à chaque itération.

Exemple : marche aléatoire dimension 1

Variable entière n, si $n > 0$ n croit ou décroit aléatoirement.

Nombre d'étapes pour arriver à 0.

```
let rec walk n k =
if n=0 then k else
if flip () then walk (n+1) (k+1) else walk (n-1) (k+1)
```

Terminaison de la marche aléatoire

Analyse de $\pi_{i,j}$ probabilité en partant de i d'atteindre j.

$$\pi_{p+i,p} = \pi_{i,0} \qquad \pi_{i,0} = \pi_{i,1} \times \pi_{1,0} = \pi_{1,0}^{i}$$

$$\pi_{1,0} = \frac{1}{2}\pi_{2,0} + \frac{1}{2} = \frac{1}{2}\pi_{1,0}^{2} + \frac{1}{2}$$

Terminaison presque sûre

$$\pi_{1,0} = 1 = \pi_{i,0}$$

Conclusion

– Modélisation monadique simple
– Espace probabiliste clair, formalisé en HOL
– Possibilité de simulation en CAML des programmes
– Raisonnement formel
– Modélisation des programmes qui terminent presque sûrement possible mais plus complexe à mettre en œuvre
  – Codage de l'itération
  – Preuve de terminaison
  – Équation de définition

L'approche Transformateur de mesures

D. Kozen, G. Plotkin, C. Jones, C. Morgan . . .

– Imperative framework with `x :=random` or $\texttt{P} +_p \texttt{Q}$.

– A randomized program `P` is non-deterministic, its semantics is a set of states.

– Analyse the distribution of this set.

– Interpret randomized algorithm `P` as transformers of "measures" on the set of state S.

Axiomatic semantics

f and g are functions from S to $[0,1]$

$$f[P]g \Leftrightarrow \forall \mu. \int f \; d\mu \leq \int g \; dP(\mu)$$

Property ϕ on states : characteristic function $\mathbb{I}_\phi$.

$$\phi(s) \text{ true} \qquad \leftrightarrow \int \mathbb{I}_\phi \; d\mu = 1$$

State s : Dirac measure δ_s.

Interpretation of a functional randomized language

(First-order) functional ML + `random`

- A program $p : \tau$ represents a set of values of type τ
- A program $p : \tau$ corresponds to a distribution μ_p on τ
- When $Q : \tau \rightarrow \texttt{Prop}$, $\mathsf{Pr}(p \in Q) = \mu_p(Q)$

Monadic transformation

Computations	Functional values
$p : \tau$	$\widehat{p} : \widehat{\tau}$
	$?\mathsf{unit} : \tau \to \widehat{\tau}$ $?\mathsf{bind} : \widehat{\tau} \to (\tau \to \widehat{\sigma}) \to \widehat{\sigma}$ $?\widehat{\mathtt{random}} : \widehat{\tau}$
$\mathsf{let}\ x\ = a\ \mathsf{in}\ b$ $[x : \tau]t$ $(t\ u)$ $\mathsf{if}\ b\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2$	$(\mathsf{bind}\ \widehat{a}\ [x : \tau]\widehat{b})$ $[x : \tau]\widehat{t}$ $(\mathsf{bind}\ \widehat{u}\ \widehat{t})$ $(\mathsf{bind}\ \widehat{b}\ [x : \mathtt{bool}]\mathsf{if}\ x\ \mathsf{then}\ \widehat{e_1}\ \mathsf{else}\ \widehat{e_2})$

Which monad ?

Transform $p : \tau$ into a measure μ_p on τ of type $\widehat{\tau} \equiv (\tau \to [0,1]) \to [0,1]$

Kind of double negation, usual monadic transformation :

$$
\begin{array}{lcl}
\mathsf{unit} & : & \tau \to \widehat{\tau} \\
 & = & \mathbf{fun}\ x \to\ \mathbf{fun}\ (f : \tau \to [0,1]) \to\ (f\ x) \\
\mathsf{bind} & : & \widehat{\tau} \to (\tau \to \widehat{\sigma}) \to \widehat{\sigma} \\
 & = & \mathbf{fun}\ \mu_a\ \mu_b \to\ \mathbf{fun}\ (f : \sigma \to [0,1]) \\
 & & \to\ (\mu_a\ \mathbf{fun}\ (x : \tau) \to\ (\mu_b\ x\ f))
\end{array}
$$

Satisfy the expected properties $(\mathsf{bind}\ (\mathsf{unit}\ x)\ \mu) = (\mu\ x)$

Interpretation of randomized constructions

$$
\begin{array}{lcl}
\texttt{random}(n) & : & \widehat{\texttt{int}} \\
 & = & \textbf{fun } f \rightarrow \Sigma_{i=1}^{n}(f\ i)/n \\
\texttt{flip}() & : & \widehat{\texttt{bool}} \\
 & = & \textbf{fun } f \rightarrow \frac{1}{2}(f\ \texttt{true}) + \frac{1}{2}(f\ \texttt{false}) \\
e_1 +_p e_2 & : & \widehat{\tau} \\
 & = & \textbf{fun } f \rightarrow p \times (\widehat{e_1}\ f) + (1-p) \times (\widehat{e_2}\ f)
\end{array}
$$

Functional interpretation

Transform a functional program with `random` into a measure.

```
let rec prime_test p n =
if n = 0 then true
else if test p (random (p-1)) then prime_test p (n-1)
else false
let rec prime_test_fun p n =
if n = 0 then (unit true)
else
bind (bind (random (p-1)) [a](unit (test p a)))
     [b]if b then prime_test_fun p (n-1) else (unit false)
```

Functional computation

The functional program computes the probability for the output to satisfy a decidable property.

The program is wrong when it computes `true` while p is not prime.

```
let prime_correct p b =
  if b then if exact_prime p then 1. else 0. else 1.
```

Computing the probability

Probability for the program to behave properly after n iterations.

```
let evaluate p n = prime_test_fun p n (prime_correct p)
```

```
# evaluate 23 1;;
- : float = 1
# [evaluate 9 0; evaluate 9 1;evaluate 9 2;evaluate 9 3];;
- : float list = [0.; 0.75; 0.9375; 0.984375]
```

Inefficient : test all the cases for the value of `random`!!

General Recursion

A program that only terminates with probability one :

```
let rec fix_test x =
if flip () then fix_test (x+1) else x
```

```
# fix_test 1;;
- : int = 1
# fix_test 1;;
- : int = 2
# fix_test 1;;
- : int = 2
# fix_test 1;;
- : int = 3
# fix_test 1;;
- : int = 1
```

Functional interpretation

```
let rec fix_test_fun x =
bind flip() [b] if b then fix_test_fun (x+1) else (unit x)
# fix_test_fun 1 (fun n -> 1.);;
Stack overflow during evaluation (looping recursion?).
```

Needs to compute limits

General fixpoints

```
let rec f x = F f x
```

$$F : (\tau \to \widehat{\sigma}) \to \tau \to \widehat{\sigma}$$

Definitions :

$$f_0 \equiv [x][f]0 \qquad f_{n+1} \equiv (F\ f_n) \qquad \mathsf{fix}\ F \equiv \sqcup_{n\in\mathbb{N}}\, f_n$$

Properties :

$$f \leq g \Rightarrow (F\ f) \leq (F\ g) \qquad \Longrightarrow \qquad f_n \leq f_{n+1}$$

$$(F\ \sqcup_{n\in\mathbb{N}}\, f_n) = \sqcup_{n\in\mathbb{N}}(F\ f_n) \qquad \Longrightarrow \qquad (F\ (\mathsf{fix}\ F)) = (\mathsf{fix}\ F)$$

Properties of measures

$$\mu(f_1 + f_2) = \mu(f_1) + \mu(f_2) \qquad \mu(\lambda f) = \lambda\mu(f)$$

$$f \leq g \Rightarrow \mu(f) \leq \mu(g)$$

$$\mu(\mathbb{O}) = 0 \qquad \mu(\mathbb{I}) \leq 1$$

$$(\forall n. f_n \leq f_{n+1}) \Rightarrow \mu(\sqcup_{n\in\mathbb{N}}(f_n)) = \sqcup_{n\in\mathbb{N}}(\mu(f_n))$$

Specification of $[0, 1]$

Primitives

$$0 \text{ (false)} \qquad\qquad 1 \text{ (true)}$$

$$i_1 \leq i_2 \ (i_1 \Rightarrow i_2) \qquad i_1 \sqcup i_2 \ (i_1 \vee i_2) \qquad i_1 \sqcap i_2 \ (i_1 \wedge i_2)$$

$$i_1 + i_2 \qquad i_1 \times i_2$$

$$\sqcup_{n \in \mathbb{N}} i_n$$

Properties

$$\text{si } i_1, i_2 \in \{0, 1\} \text{ alors } \begin{cases} i_1 \sqcup i_2 = i_1 + i_2 - i_1 \times i_2 \\ i_1 \sqcap i_2 = i_1 \times i_2 \end{cases}$$

Extension to functions

Pointwise extension of operations from $[0,1]$ to $f : \alpha \rightarrow [0,1]$.

With $c : \alpha$

$\widetilde{c} : \alpha \rightarrow [0,1]$

$\widetilde{c} = \textbf{fun } a \ \rightarrow \textbf{if } a = x \textbf{ then } 1 \textbf{ else } 0$

Dirac measure

$\delta_c : \hat{\alpha} \equiv (\alpha \rightarrow [0,1]) \rightarrow [0,1]$

$\delta_c = \textbf{fun } f \ \rightarrow (f \ c)$

Doing proofs of programs

A well-typed randomized term : $p : \tau$

$\hat{p} : (\tau \rightarrow [0,1]) \rightarrow [0,1]$

Specifications : $P \in [0,1], Q : \tau \rightarrow [0,1]$

A correct program : $\widehat{p}\ \mathbf{pr}\ P \mid Q$

Interpretation :

$$P \leq (\widehat{p}\ Q)$$

Functional specifications

Specification of the input/output behavior :

$$\nu : \tau \rightarrow \widehat{\sigma} \quad P : \tau \rightarrow [0,1] \quad Q : \sigma \rightarrow [0,1]$$

Definition of ν **pr** $(P \rightarrow Q)$:

$$\forall x : \tau.(P\ x) \leq (\nu\ x\ Q) \qquad \equiv \qquad \forall x : \tau.(\nu\ x)\ \mathbf{pr}\ (P\ x) \mid Q$$

equivalent to

$$\forall \mu : \widehat{\tau}.(\mu\ P) \leq (\mu\ [x](\nu\ x\ Q))$$

Rule for application

What is needed for application.

$$\frac{f \ \mathbf{pr} \ P \rightarrow Q \qquad x \ \mathbf{pr} \ R \mid P}{(\mathsf{bind} \ x \ f) \ \mathbf{pr} \ R \mid Q}$$

$$\frac{\forall x : \tau . t \ \mathbf{pr} \ P \mid Q}{\mathsf{fun} \ x \rightarrow t \ \mathbf{pr} \ [x] P \rightarrow Q}$$

Some rules

$$\frac{e\ \mathbf{pr}\ P \mid Q \quad P' \leq P \quad Q \leq Q'}{e\ \mathbf{pr}\ P' \mid Q'}$$

$$\frac{}{\widehat{p}\ \mathbf{pr}\ (\widehat{p}\ \mathbb{I})Q \mid [x]Q}$$

$$\frac{\widehat{e}\ \mathbf{pr}\ P \mid Q \quad \widehat{e'}\ \mathbf{pr}\ P' \mid Q}{\widehat{\textsf{if}\ b\ \textsf{then}\ e\ \textsf{else}\ e'}\ \mathbf{pr}\ ((\widehat{b}\ \widetilde{\texttt{true}})P + (\widehat{b}\ \widetilde{\texttt{false}})P') \mid Q}$$

$$\frac{}{\widehat{\texttt{flip}}\ \mathbf{pr}\ (\frac{1}{2}(Q\ \texttt{true}) + \frac{1}{2}(Q\ \texttt{false})) \mid Q}$$

Rule for fixpoint

$P_i \leq P_{i+1}$

$$\frac{f\ \mathbf{pr}\ P_i \to Q \vdash t\ \mathbf{pr}\ P_{i+1} \to Q \quad P_0 \equiv \widetilde{0}}{\mathsf{fix}\ [f]t\ \mathbf{pr}\ (\sqcup_{i \in \mathbb{N}} P_i) \to Q}$$

Example

```
let rec f x = if flip then f (x+1) else x
```

Program terminates with probability one : $(\mu[x](\widehat{f}\ x\ \mathbb{I})) = 1$

Assume $\widehat{f}$ **pr** $P_i \to \mathbb{I} \equiv \forall\mu.(\mu\ P_i) \leq (\mu[x](\widehat{f}\ x\ \mathbb{I}))$

Prove

$$\textsf{fun}\ x \to (\widehat{\textsf{if}\ \texttt{flip}\ \textsf{then}}\ f(x+1)\ \textsf{else}\ x)\ \mathbf{pr}\ P_{i+1} \to \mathbb{I}$$

$$(\textsf{if}\ \texttt{flip}\ \widehat{\textsf{then}\ f}\ (x+1)\ \textsf{else}\ x)\ \mathbb{I}$$

$$= (\textsf{bind}\ \widehat{\texttt{flip}}\ [b]\textsf{if}\ b\ \textsf{then}\ f\ \widehat{(x+1)}\ \textsf{else}\ \widehat{x})\ \mathbb{I}$$

$$= \widehat{\texttt{flip}}\ [b]((\textsf{if}\ b\ \textsf{then}\ f\ \widehat{(x+1)}\ \textsf{else}\ \widehat{x})\ \mathbb{I})$$

$$= \frac{1}{2}(\widehat{f}\ (x+1)\ \mathbb{I}) + \frac{1}{2}(\textsf{unit}\ x\ \mathbb{I}) = \frac{1}{2}(\widehat{f}\ (x+1)\ \mathbb{I}) + \frac{1}{2}$$

Justification

Takes $P_{i+1} \equiv \frac{1}{2} P_i + \frac{1}{2} \quad Q \equiv \mathbb{I}$

$$
\begin{array}{rcll}
(\mu\ P_{i+1}) & = & (\mu'\ P_{i+1}) & \qquad (\mu'\ P) \equiv (\mu\ [x](P\ (x+1))) \\
& = & \dfrac{1}{2}(\mu'\ P_i) + \dfrac{1}{2} & \\
& \leq & \dfrac{1}{2}(\mu'\ [x](f\ x\ \mathbb{I})) + \dfrac{1}{2} & \\
& = & \dfrac{1}{2}(\mu\ [x](\widehat{f}\ (x+1)\ \mathbb{I})) + \dfrac{1}{2} & \\
& = & (\mu\ [x](\dfrac{1}{2}(\widehat{f}\ (x+1)\ \mathbb{I}) + \dfrac{1}{2})) &
\end{array}
$$

Example 2

Improve the precision :

$p_1, p_2 : \tau$

$Q : \tau \to \texttt{Prop}$

p_i satisfies Q with probability at least P_i.

Hypothesis : $(Q\ x) \vee (Q\ y) \Rightarrow (Q\ (\texttt{choice}\ x\ y))$

$$q \equiv \textsf{let}\ x\ = p_1\ \textsf{in}\ \textsf{let}\ y\ = p_2\ \textsf{in}\ (\texttt{choice}\ x\ y)$$

$\widetilde{Q} : \tau \to [0,1] \quad (\widetilde{Q}\ x) \in \{0,1\} \quad (\widetilde{Q}\ x) = 1 \Leftrightarrow (Q\ x)$

$\widehat{p_i}\ \mathbf{pr}\ P_i \mid \widetilde{Q}$

Result : $\widehat{q}\ \mathbf{pr}\ (P_1 + P_2 - P_1 P_2) \mid \widetilde{Q}$

Conclusion

- Mainly reformulation of known technics at a syntactic level
- More suitable for mechanical manipulation (?)
  - Library for measure manipulation
  - Non trivial examples
  - Partial automation of the proofs