Bach Overview

NICTA

Key ideas:

- Solve problems using symbolic manipulation of formulae – reason using the axiomatic method within a logical system.
- Probability distributions are functions; e.g. $(gaussian\ \mu\ \Sigma) = \lambda x.\frac{1}{\sqrt{(2\pi)^k|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$.
- Distributions over logical statements are higher-order functions: Use higher-order logic.
- Need to supply the system with the right theorems and identities to perform useful reasoning.

Bach reasoning system:

- Implements higher-order logic (with modal extensions, but these can be ignored if desired).
- Two reasoning mechanisms:

  Equational Reasoning Efficient, but very incomplete, reasoning mechanism similar to that used in the Haskell programming language. This is the main reasoning mechanism used in Bach. We have one relevant extension over Haskell – the ability to match inside lambda abstractions. This extension is very useful for probabilistic programming.

  Theorem Proving More complete, but slower and more complex, reasoning mechanism.

Why Higher-Order?

Natural: Distributions over logical statements are higher-order functions.

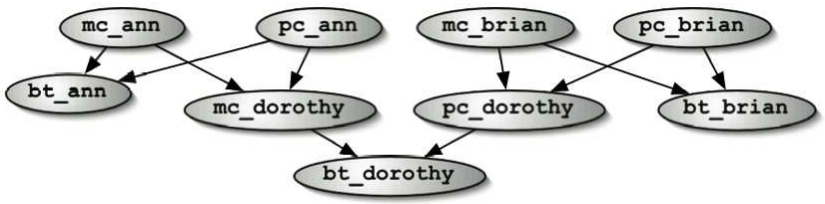
Common: IBAL and Church are both full programming languages with all the incompleteness issues of higher-order logics.

Syntactic Sugar: There are two sets of semantics for higher-order logic, Standard Semantics and Henkin Semantics. With just the equational reasoning inference mechanism used (mostly) in Bach, these are equivalent. Under Henkin semantics, higher-order logic is equivalent to first-order logic, but with a more useful syntax for probabilistic reasoning.

See also [Far08].

Bach for Graphical Models

Example: inferring blood types from maternal and paternal chromosomes [KD07; FGKP99].



$$mc : Chromosome \rightarrow Chromosome \rightarrow (Density\ Chromosome)$$
$$(mc\ A\ A\ z) = if\ (z = A)\ then\ 0.98\ else\ 0.01$$
$$(mc\ B\ A\ z) = if\ (z = B)\ then\ 0.98\ else\ 0.01$$
$$\ldots \text{ etc}$$

$$pc : Chromosome \rightarrow Chromosome \rightarrow (Density\ Chromosome)$$
$$(pc\ A\ A\ z) = if\ (z = A)\ then\ 0.98\ else\ 0.01$$
$$(pc\ B\ A\ z) = if\ (z = A)\ then\ 0.98\ else\ 0.01$$
$$\ldots \text{ etc}$$

$$bt : Chromosome \rightarrow Chromosome \rightarrow (Density\ BloodType)$$
$$(bt\ A\ A\ z) = if\ (z = A)\ then\ 0.97\ else\ 0.01$$
$$(bt\ B\ A\ z) = if\ (z = AB)\ then\ 0.97\ else\ 0.01$$
$$\ldots \text{ etc}$$

Let mc_x, pc_x and bt_x denote the relevant variables. We use a, b, d as shorthands for $ann, brian$ and $dorothy$. We define the joint as follows:

$$joint : Density\ (Chromosome \times Chromosome \times Chromosome \times Chromosome \times Chromosome \times Chromosome \times BloodType \times BloodType \times BloodType)$$

$$(joint\ (mc_a, pc_a, mc_b, pc_b, mc_d, pc_d, bt_a, bt_b, bt_d)) = (1/3)^4 \times (mc\ mc_a\ pc_a\ mc_d) \times (pc\ mc_b\ pc_b\ pc_d) \times (bt\ mc_a\ pc_a\ bt_a) \times (bt\ mc_b\ pc_b\ bt_b) \times (bt\ mc_d\ pc_d\ bt_d).$$

Note that the functions describing the conditional probabilities are re-used, but instantiated with different variables. It is the variables used in the top level calculation that represent the 'random variables', not the parameters to intermediate functions.

Inference Rules

Variable Elimination

This identity allows for variable elimination:

$$\sum_{y \in s} t_1 \times \cdots \times t_n \;=\; t_i \times \sum_{y \in s} t_1 \times \cdots \times t_{i-1} \times t_{i+1} \times \cdots \times t_n$$

-- if y is not free in t_i.

e.g. Computation of
$(sum\ s_1\ \lambda x.(sum\ s_2\ \lambda y.(sum\ s_3\ \lambda z.((g_1\ x) \times (g_2\ (x,y)) \times (g_3\ (y,z))))))$

$$\sum_{x \in s_1} \sum_{y \in s_2} \sum_{z \in s_3} (g_1\ x) \times (g_2\ (x,y)) \times (g_3\ (y,z))$$

$$\sum_{x \in s_1} \sum_{y \in s_2} (g_1\ x) \times \sum_{z \in s_3} (g_2\ (x,y)) \times (g_3\ (y,z))$$

$$\sum_{x \in s_1} (g_1\ x) \times \sum_{y \in s_2} \sum_{z \in s_3} (g_2\ (x,y)) \times (g_3\ (y,z))$$

$$\sum_{x \in s_1} (g_1\ x) \times \sum_{y \in s_2} (g_2\ (x,y)) \times \sum_{z \in s_3} (g_3\ (y,z))$$

Inversion Elimination

In simplified form:

$$\sum_{f \in Y^X} \prod_{x \in X} (\phi\ (f\ x)) = \prod_{x \in X} \sum_{f \in Y^{\{x\}}} (\phi\ (f\ x)),$$

Counting Elimination

In simplified form:

$$\sum_{p \in \{\top,\bot\}^X} \prod_{x \in X} (\phi\ (p\ x)) \;=\; \sum_{i=0}^{|X|} \binom{|X|}{i} (\phi\ \top)^i (\phi\ \bot)^{|X|-i},$$

Shattering

$$\sum_{f \in Z^X} t_1 \times t_2 = \left(\sum_{f \in Z^Y} t_1 \right) \times \left(\sum_{f \in Z^{X \setminus Y}} t_2 \right)$$

if there exists $Y \subset X$ such that

1. for all free subterm $(f\ t)$ in t_1, either $t \in Y$ or t is a variable that can only be substituted with an element in Y;
2. for all free subterm $(f\ t)$ in t_2, either $t \notin Y$ or t is a variable that can only be substituted with an element in $X \setminus Y$.

Comparison with IBAL / Church

NICTA

Random variables

IBAL/Church Each execution of a *flip* function introduces a random variable

Bach Random variables are listed explicitly in the definition of the joint distribution, but may be structured (e.g. lists, trees)

Structural Uncertainty

IBAL/Church Random variables can be used to control structure generation

Bach Structured random variables are used

Computational Mechanism

IBAL Specially developed exact reasoning procedure

Church Monte-Carlo sampling over program space

Bach Symbolic/equational reasoning

Toy code comparison with IBAL

IBAL	Bach

```
flip 0.7
```

```
flip : Float -> Bool -> Float;
(flip m x) = if x then m else (sub 1 m);
```

```
: (flip 0.7 True);
: (flip 0.7 False);
```

```
x = flip 0.7
y = flip 0.7
if x == y then 'equal else 'notequal
```

```
Equal, NotEqual : Label;

cpd : Bool -> Bool -> Label -> Float;
(cpd x1 x2 m) = if (== (== x1 x2) (== m Equal)) then
                   1.0
                else
                   0.0 ;
```

```
joint1 : Bool -> Bool -> Label -> Float;
(joint1 x1 x2 m) = (mul (flip 0.7 x1) (mul (flip 0.7 x2) (cpd x1 x2 m)));

answer1 : Label -> Float;
(answer1 m) = (sum {True, False} \x1. (sum {True, False} \x2. (joint1 x1 x2 m)));

:(answer1 Equal);
:(answer1 NotEqual);
```

```
z = x
if x == z then 'equal else 'notequal
```

```
cpd2 : Bool -> Bool -> Float;
(cpd2 x1 x2) = if (== x1 x2) then 1.0 else 0.0;

joint2 : Bool -> Bool -> Label -> Float;
(joint2 x1 x2 m) = (mul (flip 0.7 x1)
                        (mul (cpd2 x1 x2) (cpd x1 x2 m)));
```

```
answer2 : Label -> Float;
(answer2 m) = (sum {True, False} \x1.
                   (sum {True, False} \x2.
                        (joint2 x1 x2 m)));

:(answer2 Equal);
:(answer2 NotEqual);
```

```
w = if flip 0.5 then x else flip 0.7
if w == x then 'equal else 'notequal
```

```
cpd3: Bool -> Bool -> Bool -> Float;
(cpd3 x1 x2 x3) = if x1 then
                     (cpd2 x2 x3)
                  else
                     (flip 0.7 x3);

joint3 : Bool -> Bool -> Bool -> Label -> Float;
(joint3 x1 x2 x3 m) = (mul (flip 0.5 x1)
                           (mul (flip 0.7 x2)
                                (mul (cpd3 x1 x2 x3)
                                     (cpd x2 x3 m))));
```

```
answer3 : Label -> Float;
(answer3 m) = (sum {True, False} \x1.
                   (sum {True, False} \x2.
                        (sum {True, False} \x3.
                             (joint3 x1 x2 x3 m))));

:(answer3 Equal);
:(answer3 NotEqual);
```

```
:l list.ibl
gthelp(x,y,z) = if x == y then
                   false
                else
                   if x == z then
                      true
                   else
                      gthelp(x,y-1,z+1)
gt(x,y) = gthelp(x,y,y)
grow(x) = ~(gt(list.length(x),2) | flip 0.1)
f(x) = if grow(x) then f('elt :: x) else x
f([])
```

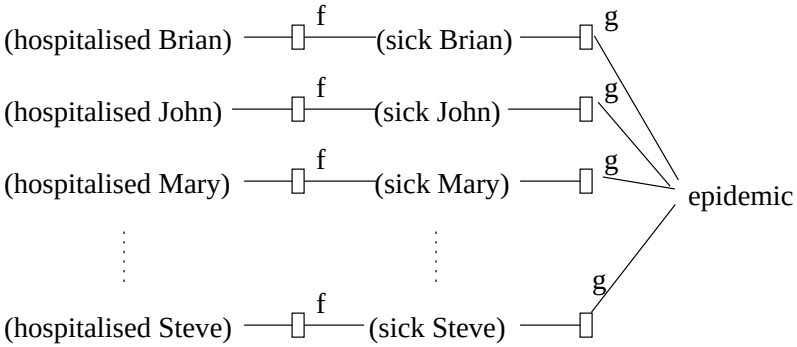
```
import lists.es ;

Elt : Thing;

f : (List Thing) -> (List Thing) -> Float;
(f x y) = if (> (length x) 2) then
             if (== x y) then 1.0 else 0.0
          else
             if (== x y) then 0.1 else
                (mul 0.9 (f (# Elt x) y));
```

```
: (f [] []);
: (f [] [Elt]);
: (f [] [Elt, Elt]);
: (f [] [Elt, Elt, Elt]);
: (f [] [Elt, Elt, Elt, Elt]);
```

Lifted Inference - example problem



Variables in Bayes net (note parameterisation):

$$epidemic : \Omega$$
$$hospitalised : Person \rightarrow \Omega$$
$$sick : Person \rightarrow \Omega$$

$$all, vst, othr : Person \rightarrow \Omega$$
$$all = \{Brian, John, Mary, Mike, Liz, Steve\}$$
$$vst = \{Brian, John, Mary, Mike\}$$
$$othr = all \setminus vst$$

$$joint : (Person \rightarrow \Omega) \times (Person \rightarrow \Omega) \times \Omega \rightarrow Real$$
$$(joint\ (s, h, e)) = \prod_{x \in all} (f\ (s\ x)\ (h\ x)) \times (g\ (s\ x)\ e)$$

Factor definitions:

$$f : \Omega \rightarrow \Omega \rightarrow Real$$
$$(f\ \top\ \top) = 7.0 \qquad (f\ \bot\ \top) = 0.5$$
$$(f\ \top\ \bot) = 3.0 \qquad (f\ \bot\ \bot) = 9.5$$

$$g : \Omega \rightarrow \Omega \rightarrow Real$$
$$(g\ \top\ \top) = 1.0 \qquad (g\ \bot\ \top) = 0.5$$
$$(g\ \top\ \bot) = 9.0 \qquad (g\ \bot\ \bot) = 9.5$$

Query:

$$Pr(e = \top \mid (h\ Brian) = \top, (h\ John) = \top, (h\ Mary) = \top, (h\ Mike) = \bot) =$$
$$\frac{1}{K} \sum_{s \in 2^{all}} \sum_{h \in 2^{othr}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top) \left(\prod_{x \in vst \setminus \{Mike\}} (f\ (s\ x)\ \top)(g\ (s\ x)\ \top) \right) \left(\prod_{x \in all \setminus vst} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top) \right)$$

Lifted Inference - example calculation

$$\sum_{s\in 2^{all}} \sum_{h\in 2^{othr}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big(\prod_{x\in vst\setminus\{Mike\}} (f\ (s\ x)\ \top)(g\ (s\ x)\ \top)\Big)\Big(\prod_{x\in othr} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top)\Big) \quad [24]$$

$$\sum_{s\in 2^{all}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big(\prod_{x\in vst\setminus\{Mike\}} (f\ (s\ x)\ \top)(g\ (s\ x)\ \top)\Big) \sum_{h\in 2^{othr}} \prod_{x\in othr} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top) \quad [33]$$

$$\Big(\sum_{s\in 2^{\{Mike\}}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big) \sum_{s\in 2^{all\setminus\{Mike\}}} \Big(\prod_{x\in vst\setminus\{Mike\}} (f\ (s\ x)\ \top)(g\ (s\ x)\ \top)\Big) \sum_{h\in 2^{othr}} \prod_{x\in othr} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top) \quad [33]$$

$$\Big(\sum_{s\in 2^{\{Mike\}}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big)\Big(\sum_{s\in 2^{vst\setminus\{Mike\}}} \prod_{x\in vst\setminus\{Mike\}} (f\ (s\ x)\ \top)(g\ (s\ x)\ \top)\Big) \sum_{s\in 2^{othr}} \sum_{h\in 2^{othr}} \prod_{x\in othr} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top) \quad [32]$$

$$\Big(\sum_{s\in 2^{\{Mike\}}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big)\Big(\sum_{\pi\in M_2^{|vst\setminus\{Mike\}|}} C_\pi \prod_{x'\in\{\top,\bot\}} ((f\ x'\ \top)(g\ x'\ \top))^{(\mu_\pi\ x')}\Big) \sum_{s\in 2^{othr}} \sum_{h\in 2^{othr}} \prod_{x\in othr} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top) \quad [31]$$

$$\Big(\sum_{s\in 2^{\{Mike\}}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big)\Big(\sum_{\pi\in M_2^{|vst\setminus\{Mike\}|}} C_\pi \prod_{x'\in\{\top,\bot\}} ((f\ x'\ \top)(g\ x'\ \top))^{(\mu_\pi\ x')}\Big) \sum_{s\in 2^{othr}} \prod_{x\in othr} \sum_{h\in 2^{\{x\}}} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top) \quad [31]$$

$$\Big(\sum_{s\in 2^{\{Mike\}}} (f\ (s\ Mike)\ \bot)(g\ (s\ Mike)\ \top)\Big)\Big(\sum_{\pi\in M_2^{|vst\setminus\{Mike\}|}} C_\pi \prod_{x'\in\{\top,\bot\}} ((f\ x'\ \top)(g\ x'\ \top))^{(\mu_\pi\ x')}\Big) \prod_{x\in othr} \sum_{s\in 2^{\{x\}}} \sum_{h\in 2^{\{x\}}} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top)$$

$$\vdots$$

$$7.75 \times \Big(\sum_{\pi\in M_2^{|vst\setminus\{Mike\}|}} C_\pi \prod_{x'\in\{\top,\bot\}} ((f\ x'\ \top)(g\ x'\ \top))^{(\mu_\pi\ x')}\Big) \prod_{x\in othr} \sum_{s\in 2^{\{x\}}} \sum_{h\in 2^{\{x\}}} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top)$$

$$\vdots$$

$$7.75 \times 381.08 \times \prod_{x\in othr} \sum_{s\in 2^{\{x\}}} \sum_{h\in 2^{\{x\}}} (f\ (s\ x)\ (h\ x))(g\ (s\ x)\ \top)$$

$$\vdots$$

$$7.75 \times 381.08 \times 225$$

Continuous Distributions

$$gaussian : Vector \rightarrow Matrix \rightarrow Density\ Vector$$
$$(gaussian\ \mu\ \Sigma) = \lambda x.\frac{1}{\sqrt{(2\pi)^k|\Sigma|}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

Definition of the Gaussian distribution

$$(normalise\ \lambda s.((gaussian\ \mu_1\ \Sigma_1)\ s) \times ((gaussian\ \mu_2\ \Sigma_2)\ s)) =$$
$$(\lambda n.(gaussian\ \ (\Sigma_1 n\mu_2 + \Sigma_2 n\mu_1)\ \ (\Sigma_1 n\Sigma_2))\ \ (\Sigma_1 + \Sigma_2)^{-1})$$

Definition of conjugate prior update for Gaussian distribution

Actual Location		Desired Action		Actual Action		Observed Location		Estimated State		
x	y	x	y	x	y	x	y	mean		Variance
0.0	0.0	5.0	0.0	4.881	0.035			5.0	0.0	1.1
4.881	0.035					5.227	-0.228	5.208	-0.209	0.092
4.881	0.035	5.0	5.0	5.053	4.721			10.208	4.791	0.192
9.934	4.756					10.136	4.717	10.161	4.743	0.066
9.934	4.756	0.0	5.0	0.009	5.110			10.161	9.743	0.166
9.943	9.866					9.806	9.929	9.940	9.859	0.062
9.943	9.866	-5.0	0.0	-4.909	0.744			4.940	9.860	0.162
5.034	10.610					5.477	10.827	5.272	10.458	0.062
5.034	10.610	-5.0	-5.0	-4.788	-5.628			0.272	5.458	0.162
0.246	4.982					0.256	4.744	0.262	5.017	0.062
0.246	4.982	0.0	-5.0	0.050	-5.006			0.262	0.017	0.162
0.296	-0.024					0.143	-0.543	0.189	-0.329	0.062

The actual and estimated state of a 2D robot given a series of noisy actions and observations. In this example, the covariance matrix is always diagonal with equal entries, and so for compactness we have only given one number for the variance.

Monte-Carlo evaluation of probabilities

A Sampling Scheme for (Basic) Terms

Goal Need a scheme to define and sample from $P(t|s)$.

Idea Locally perturb s to obtain t.

Mechanism An edit $s \rightsquigarrow t \quad [x_1\ d_1, \ldots, x_k\ d_k]$.

Example edit grammar

$$
\begin{aligned}
x &\rightsquigarrow x + 1.0 \\
x &\rightsquigarrow x + y \quad [y \sim (N\ x\ 0.1)] \\
(\#\ x\ y) &\rightsquigarrow (\#\ x\ (\#\ w\ (\mathit{reverse}\ y))) \quad [w \sim (N\ x\ 0.1)]
\end{aligned}
$$

Example edits of $(\#\ 3.0\ (\#\ 4.0\ (\#\ 5.0\ [])))$:

$$
\begin{aligned}
&(\#\ 3.0\ (\#\ 4.0 + 1.0\ (\#\ 5.0\ []))) \\
&(\#\ 3.0\ (\#\ 4.0\ (\#\ 5.0 + 5.3\ []))) \\
&(\#\ 3.0\ (\#\ 3.1\ (\mathit{reverse}\ (\#\ 4.0\ (\#\ 5.0\ [])))))
\end{aligned}
$$

Issues Need to handle *equivalent* terms so that e.g. $P(1.0 + 2.0|s)$ and $P(3.0|s)$ don't get counted separately.

Contact

NICTA

ARIA Research Group

Will Uther	William.Uther@nicta.com.au	National ICT Australia & University of New South Wales
Kee-Siong Ng	KeeSiong.Ng@nicta.com.au	National ICT Australia & Australian National University
John Lloyd	john.lloyd@anu.edu.au	College of Engineering and Computer Science, Australian National University

References

[Far08] William M. Farmer. The seven virtues of simple type theory. *Journal of Applied Logic*, 6(3):267–286, 2008.

[FGKP99] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1300–1307, 1999.

[KD07] Kristian Kersting and Luc De Raedt. Bayesian logic programming: Theory and tool. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 10. MIT Press, 2007.

[LN07] John W. Lloyd and Kee Siong Ng. Learning modal theories. In S. Muggleton, R. Otero, and A. Tamaddoni-Nezhad, editors, *Proceedings of the 16th International Conference on Inductive Logic Programming*, LNAI 4455, pages 320–334, 2007.

[NL08] K. S. Ng and J.W. Lloyd. Probabilistic reasoning in a classical logic. *Journal of Applied Logic*, 2008. In press.

[NLU09] K. S. Ng, J.W. Lloyd, and W.T.B. Uther. Probabilistic modelling, inference and learning using logical theories. *Annals of Mathematics and Artificial Intelligence*, 2009. To Appear.