Probabilistic Programming
or
Revd. Bayes meets Countess Lovelace

John Winn, Microsoft Research Cambridge

Bayes 250 Workshop, Edinburgh, September 2011

“Reverend Bayes, meet Countess Lovelace”

Statistician
1702 – 1761

Programmer
1815 – 1852

Roadmap

- Bayesian inference is hard
- Two key problems
- Probabilistic programming
- Examples
- Infer.NET
- An application
- Future of Bayesian inference

Bayesian inference is *hard*

! **Complex mathematics**

! **Approximate algorithms**

! **Error toleration**

! **Hard to schedule**

! **Hard to detect convergence**

! **Numerical stability**

! **Computational cost**

The average developer…

The expert statistician

The expert statistician

Probabilistic programming

- Bayesian inference at the language level
- BUGS & WinBUGS showed the way
- Three keywords added to (any) language
  - `random` – makes a random variable
  - `constrain` – constrains a variable e.g. to data
  - `infer` – returns the distribution of a variable

Random variables

- Normal variables have a fixed single value:

```
int length=6,
bool visible=true.
```

- Random variables have uncertain value specified by a probability distribution:

```
int length = random Uniform(0,10)
bool visible = random Discrete(0.8)
```

- **`random`** operator means ‘is distributed as’.

Constraints

- We can define constraints on random variables:

```
constrain(visible==true)
constrain(length==4)
constrain(length>0)
constrain(i==j)
```

- **`constrain`(b)** means ‘we constrain b to be true’.

Inference

- The **infer** operator gives the posterior distribution of one or more random variables.
- Example:

```
int i = random Uniform(1,10);
bool b = (i*i>50);
Dist bdist = infer(b);//Bernoulli(0.3)
```

- Output of **infer** is always *deterministic* even when input is *random*.

Hello Uncertain World

```
string A = random new Uniform<string>();
string B = random new Uniform<string>();
string C = A+" "+B;
constrain(C == "Hello Uncertain World");

infer(A)
// 50%: "Hello", 50%: "Hello Uncertain"
infer(B)
// 50%: “Uncertain World", 50%: “World"
```

Semantics: sampling interpretation

Imagine running the program many times:

- **`random`**`(d)` *samples* from the distribution `d`
- **`constrain`**`(b)` *discards* the run if `b` is false
- **`infer`**`(x)` *collects* the value of x into a persistent memory
  - If enough x's have been stored, returns their distribution
  - Otherwise starts a new run

Bayesian Model Comparison (if, else)

```
bool drugWorks = random new Bernoulli(0.5);
if (drugWorks) {
  pControl = random new Beta(1,1);
  control[:]  = random new Bernoulli(pControl);
  pTreated = random new Beta(1,1);
  treated[:]  = random new Bernoulli(pTreated);
} else {
  pAll     = random new Beta(1,1);
  control[:]  = random new Bernoulli(pAll);
  treated[:]  = random new Bernoulli(pAll);
}
// constrain to data
constrain(control == controlData);
constrain(treated == treatedData);
// does the drug work?
infer(drugWorks)
```

Probabilistic programs and graphical models

Probabilistic Program	Graphical Model
Variables	Variable nodes
Functions/operators	Factor nodes/edges
Fixed size loops/arrays	Plates
If statements	Gates (Minka & Winn)
Variable sized loops, Complex indexing, jagged arrays, mutation, recursion, objects/ properties…	No common equivalent

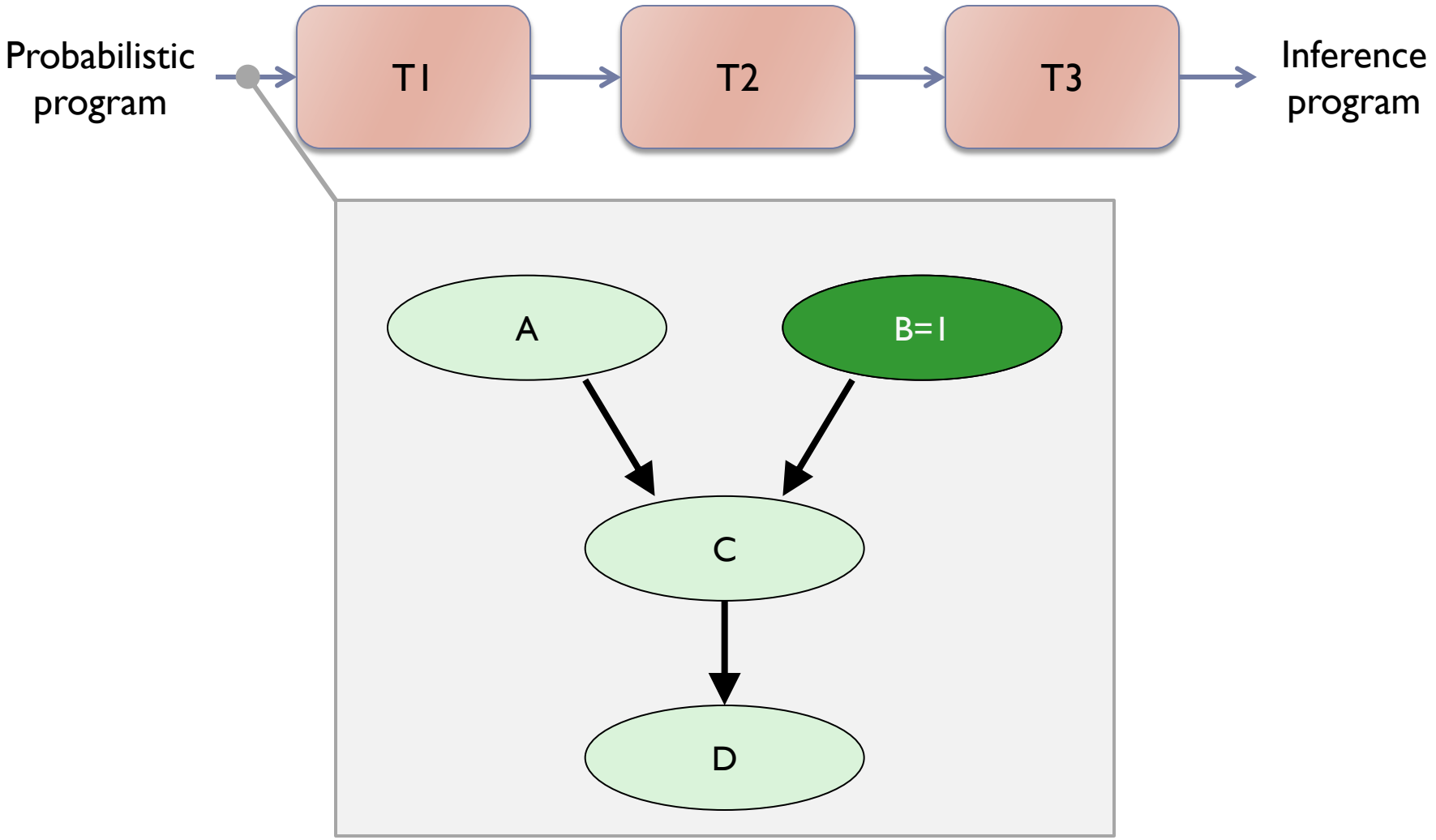
Causality

```
bool AcausesB = random new Bernoulli(0.5);
if (AcausesB) {
  A = random Aprior;
  B = NoisyFunctionOf(A);
} else {
  B = random Bprior;
  A = NoisyFunctionOf(B);
}
// intervention replaces above definition of B
if (interventionOnB) B = interventionValue;
// constrain to data
constrain(A == AData);
constrain(B == BData);
constrain(interventionOnB==interventionData);
// does A causes B, or vice versa?
infer(AcausesB)
```

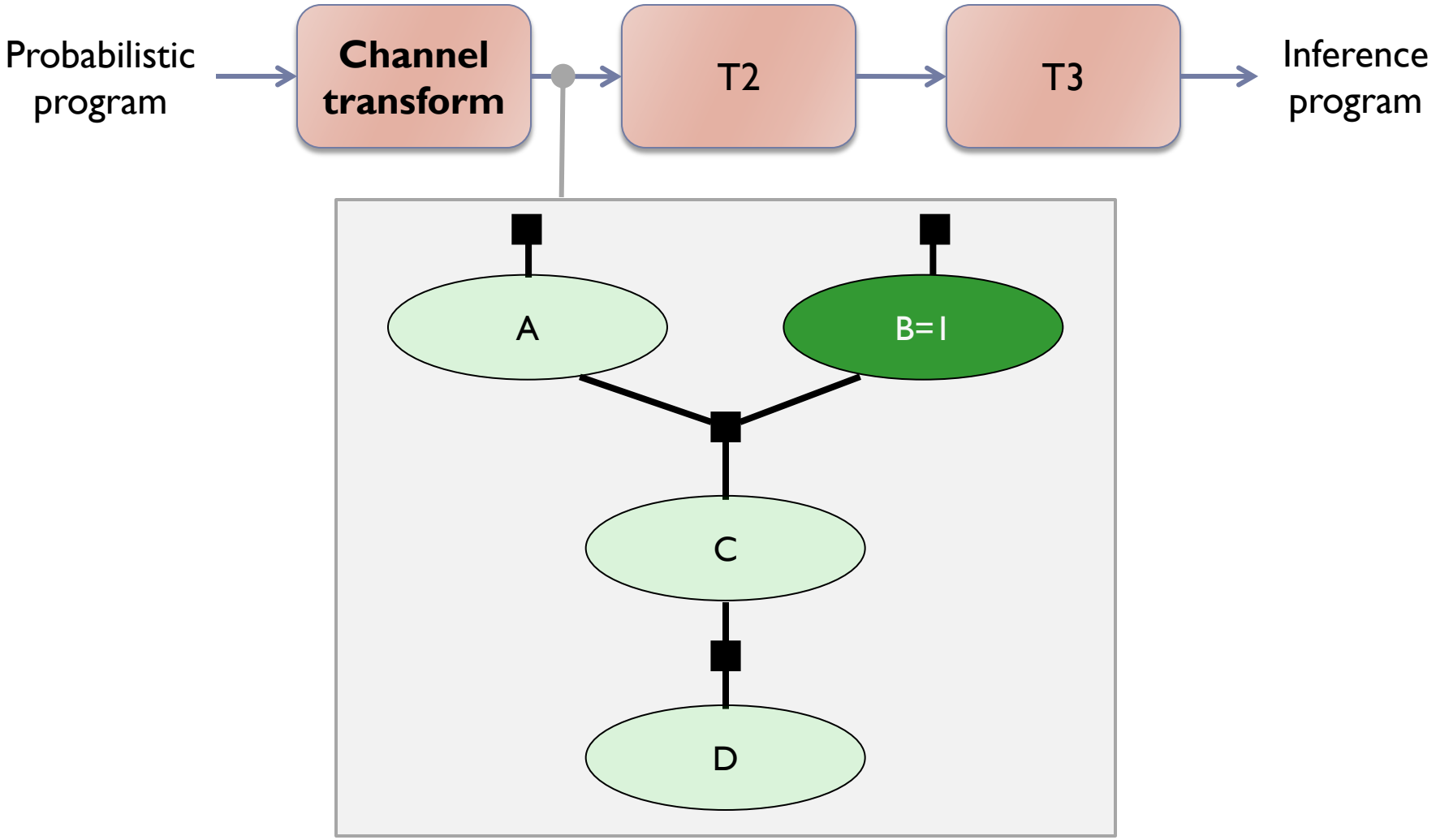
Infer.NET

- *Compiles* probabilistic programs into inference code (EP/VMP/Gibbs).
- Supports many (but not all) probabilistic program elements
- Extensible – distribution channel for new machine learning research
- Consists of a chain of code transformations:

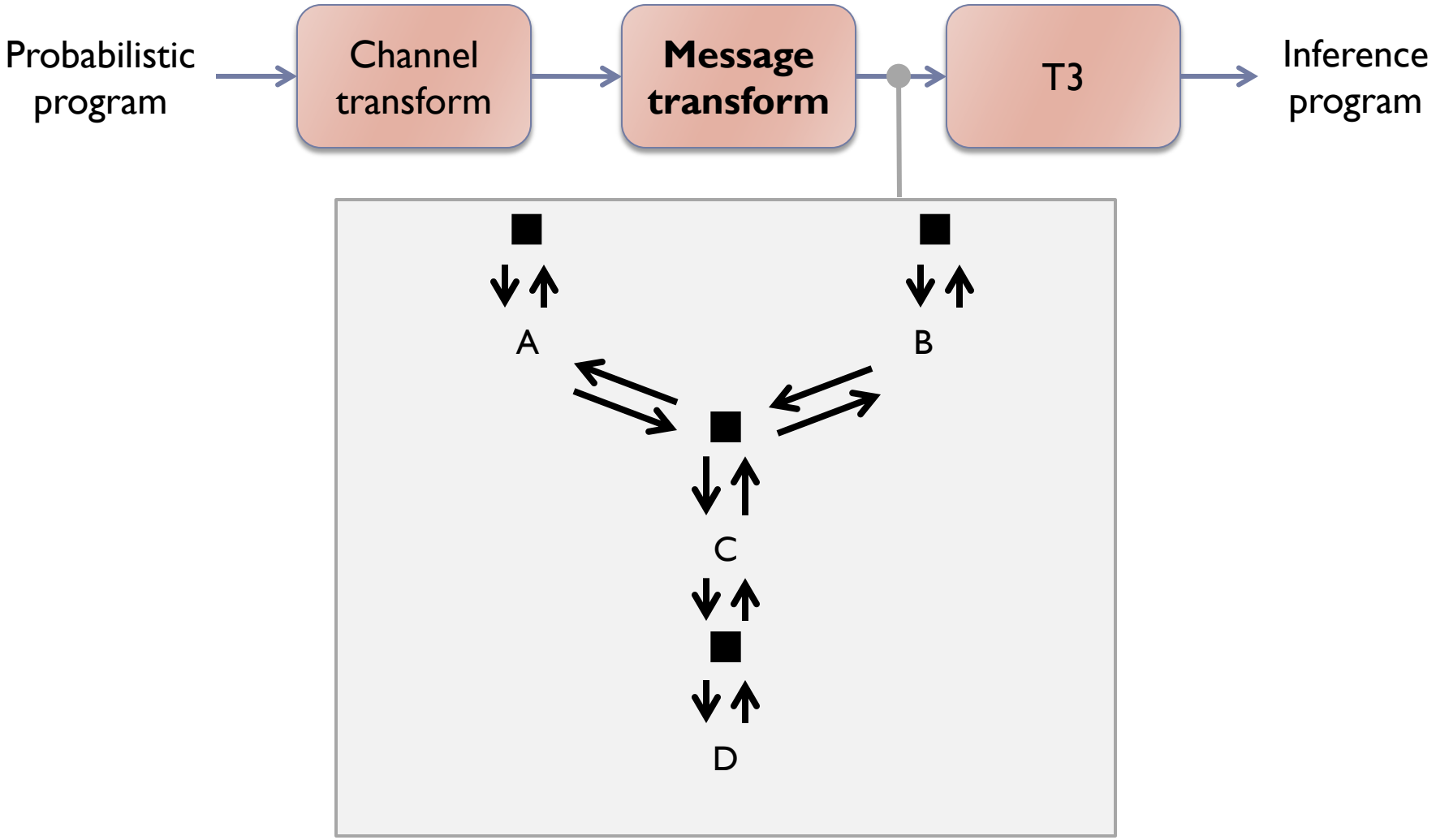
Infer.NET inference engine



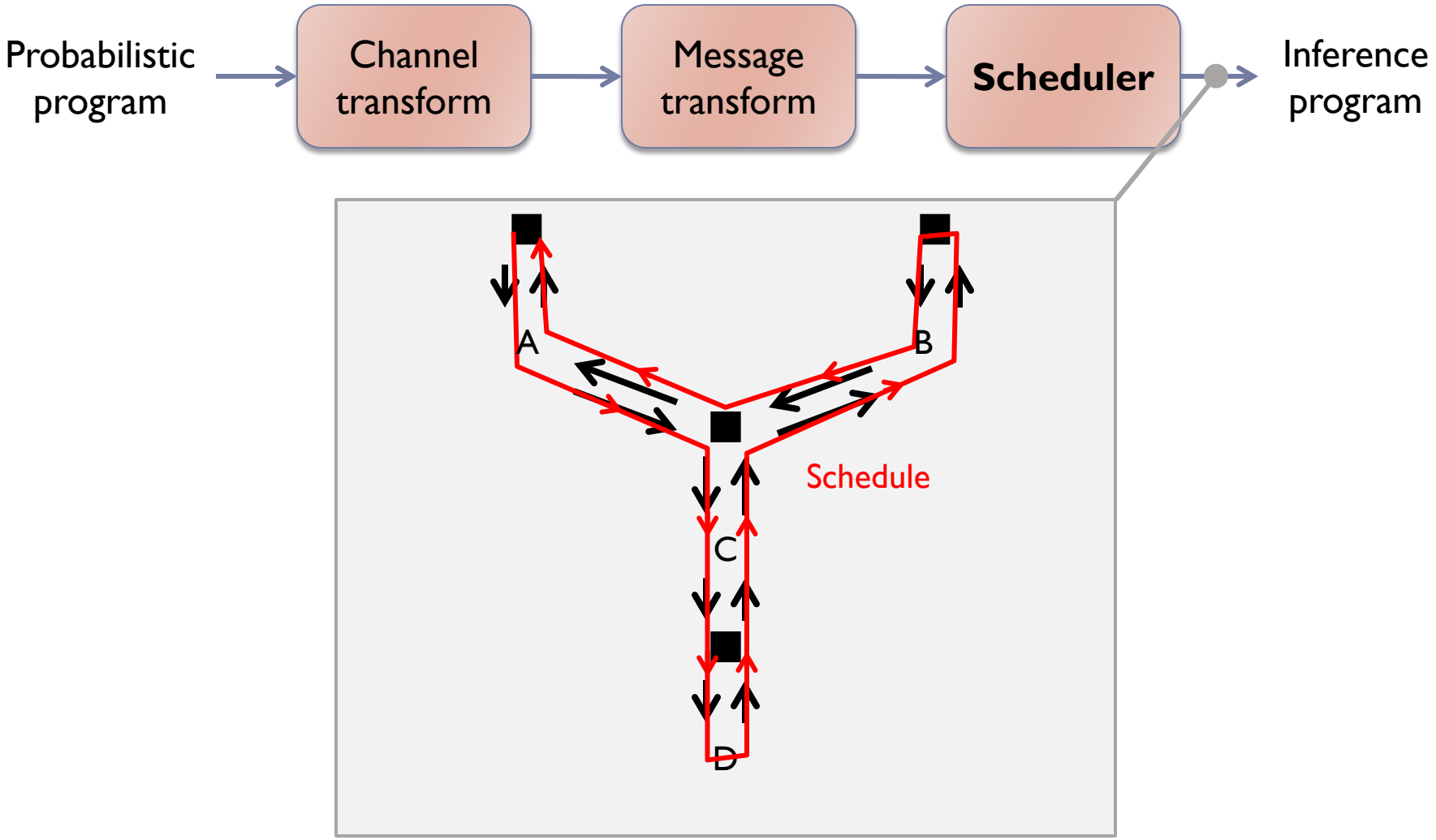
Infer.NET compiler



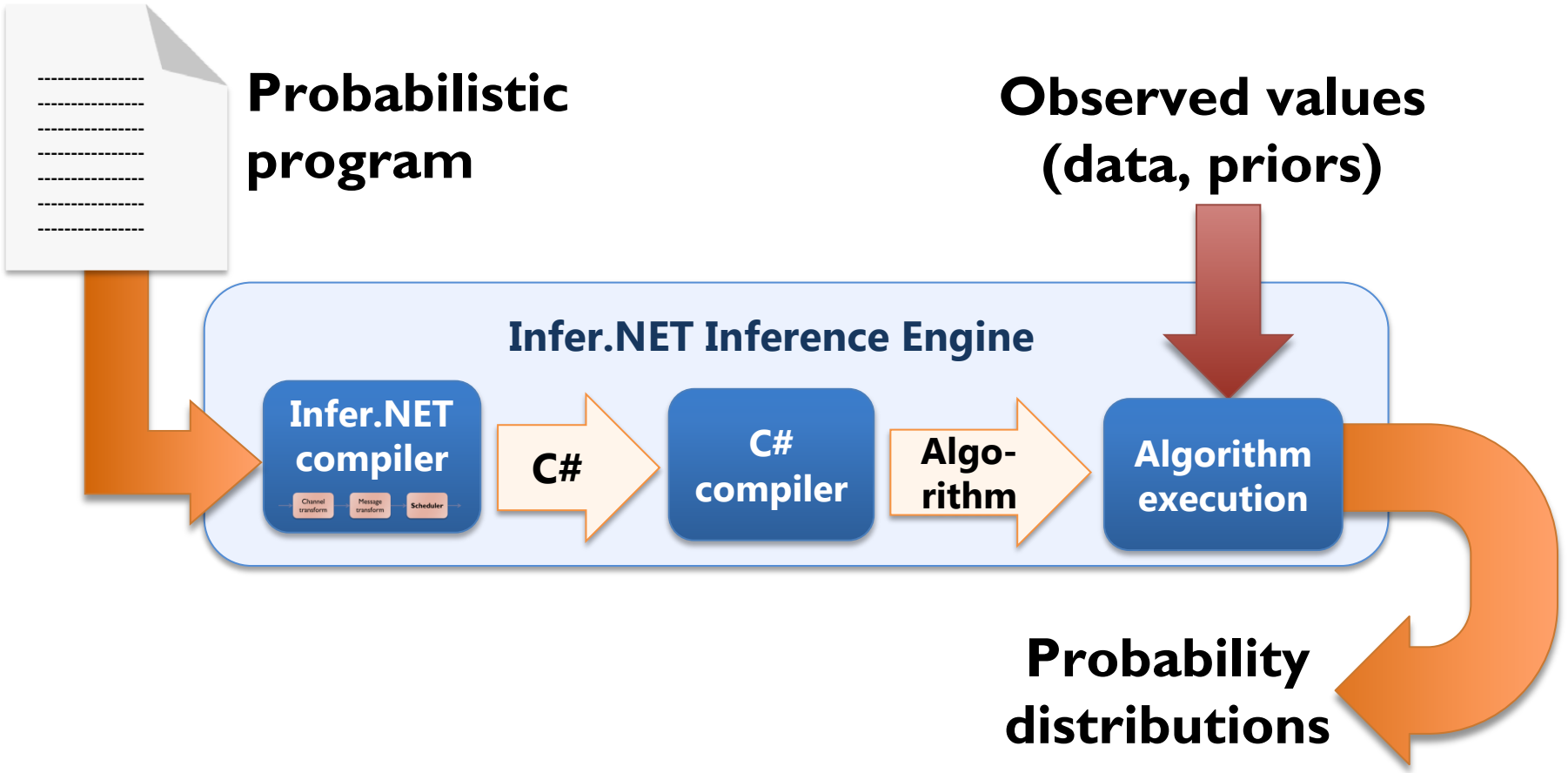
Infer.NET compiler



Infer.NET compiler



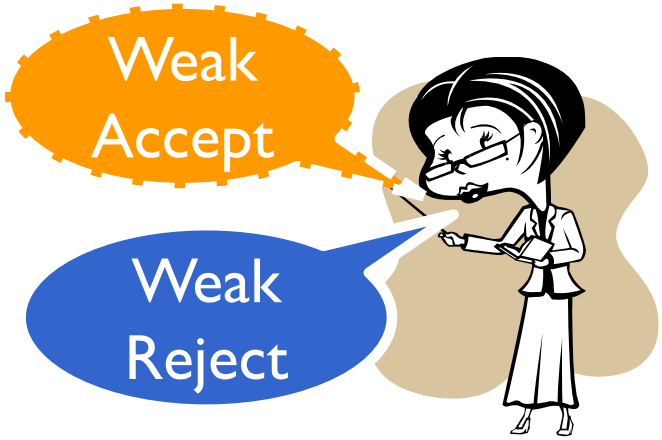
Infer.NET architecture

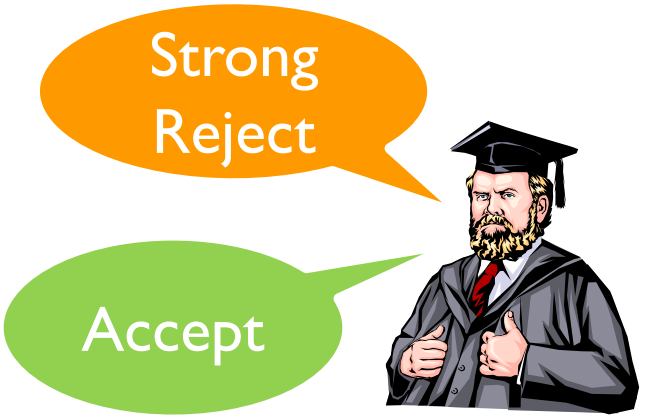


Application: Reviewer Calibration

[SIGKDD Explorations '09]

Submissions





Reviewers

Aim: infer calibrated score

Reviewer calibration code

```
// Calibrated score – one per submission
Quality[s] = random Gaussian(qualMean,qualPrec).ForEach(s);

// Precision associated with each expertise level
Expertise[e] = random Gamma(expMean,expVar).ForEach(e);

// Review score – one per review
Score[r]= random Gaussian(Quality[sOf[r]],Expertise[eOf[r]]);

// Accuracy of judge
Accuracy[j] = random Gamma(judgeMean,judgeVar).ForEach(j);

// Score thresholds per judge
Threshold[t][j] = random Gaussian(NomThresh[t], Accuracy[j]);

// Constrain to match observed rating
constrain(Score[r] > Threshold[rating][jOf[r]]);
constrain(Score[r] < Threshold[rating+1][jOf[r]]);
```

Results for KDD 2009

- Paper scores
  - Highest score: 1 'strong accept' and 2 'accept'
  - Beat paper with 3 'strong accept' from more generous reviewers
- Score certainties
  - Most certain: 5 'weak accept' reviews
  - Least certain: 'weak reject', 'weak accept', and 'strong accept'.
- Reviewer generosity
  - Most generous reviewer: 5 strong accepts
- More expert reviews are higher precision:
  - Informed Outsider: 1.22, Knowledgeable: 1.35 Expert: 1.59
  - Experts are more likely to agree with each other (!)

Future of Bayesian inference

How to make Bayesian inference accessible to the average developer + break the complexity barrier?

- Probabilistic programming in familiar languages
- Probabilistic debugging tools
- Scalable execution
- Online community with shared programs and shared data + continual evaluation of each program against all relevant data and vice versa.

We hope Infer.NET will be part of this future!

research.microsoft.com/infernet

Microsoft Research

infer.net

- Home
- Download

Documentation

- User Guide
- Tutorials & Examples
- API Documentation
- Resources & References

Support

- FAQ
- Forum
- Blog
- Contact Us

Team

- Meet the Team
- MLP Group
- MSR Cambridge

Infer.NET

Infer.NET is a framework for running Bayesian inference in graphical models. It can also be used for probabilistic programming as shown in this video.

You can use Infer.NET to solve many different kinds of machine learning problems, from standard problems like classification or clustering through to customised solutions to domain-specific problems. Infer.NET has been used in a wide variety of domains including information retrieval, bioinformatics, epidemiology, vision, and many others.

Infer.NET 2.4 beta 2 is now available [17th December, 2010].
This release is a *minor update* to Infer.NET 2.4 beta 1. See the release change history for details.

Download Infer.NET

Questions? Suggestions? Please use the forum to provide feedback and to share the ways in which you are using Infer.NET (or send e-mail to infersup@microsoft.com).

Citing Infer.NET If you use Infer.NET as part of your research, please cite us as detailed in the FAQ.

Questions?

Infer.NET now and next