Infer.NET and CSOFT

A framework and language for Machine Learning

John Winn

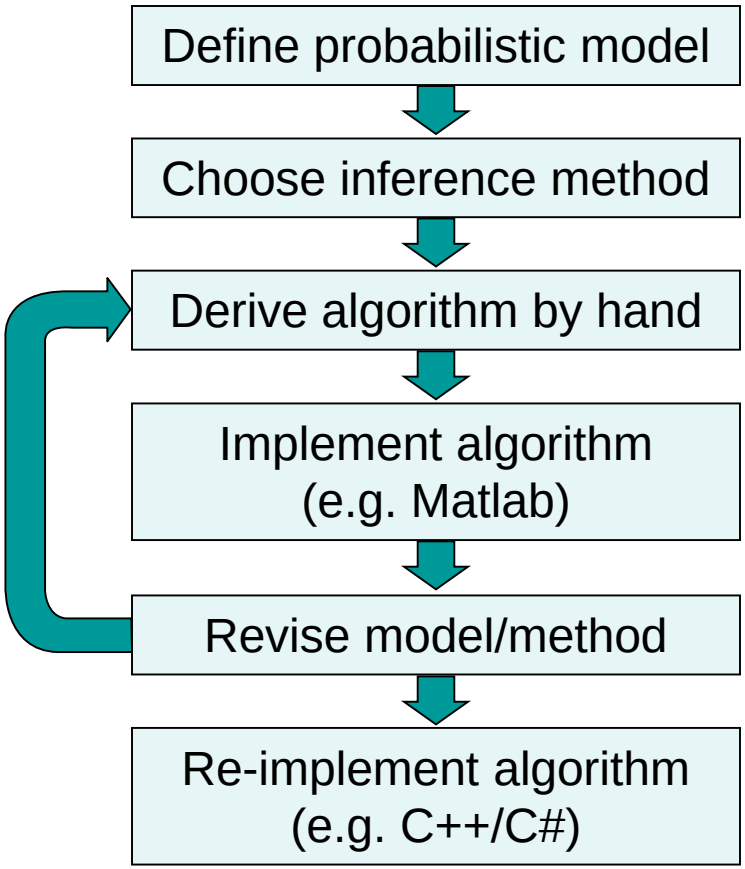
Machine Learning and Perception Group

NIPS, December 2008

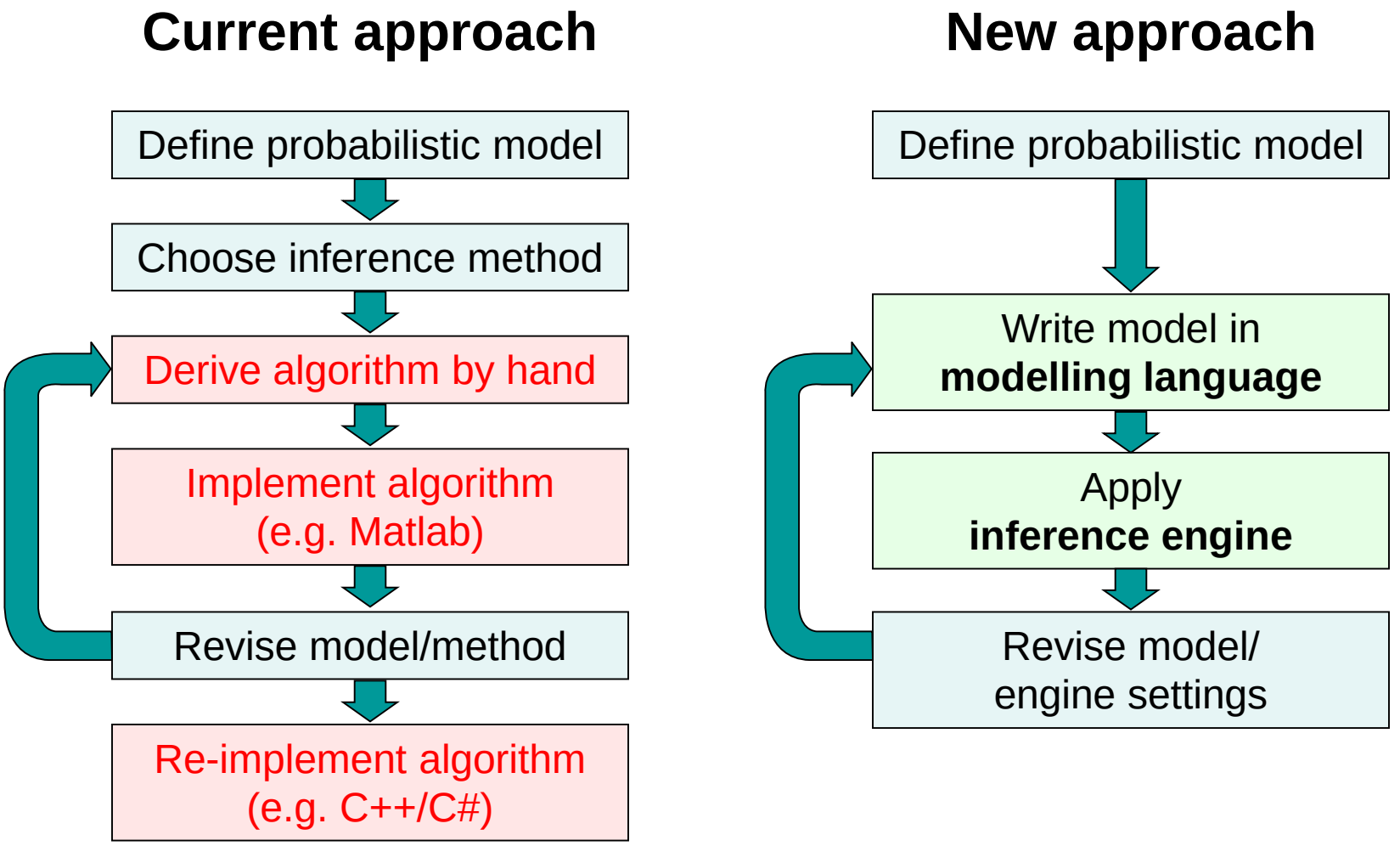
Microsoft Research Cambridge

How machine learning is applied

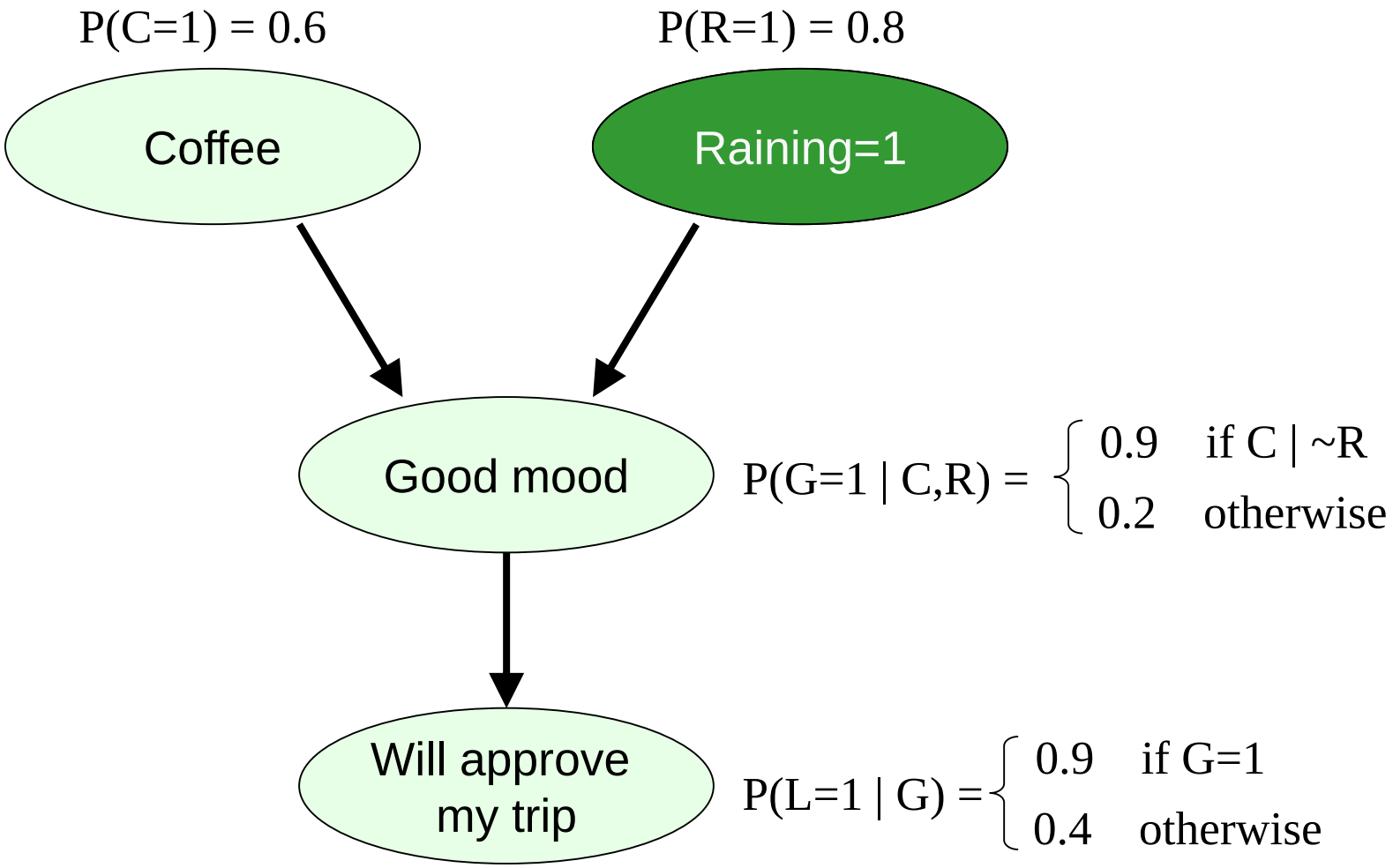
Current approach



How machine learning is applied



Example: BossPredictor



Hand-coded solution (Matlab)

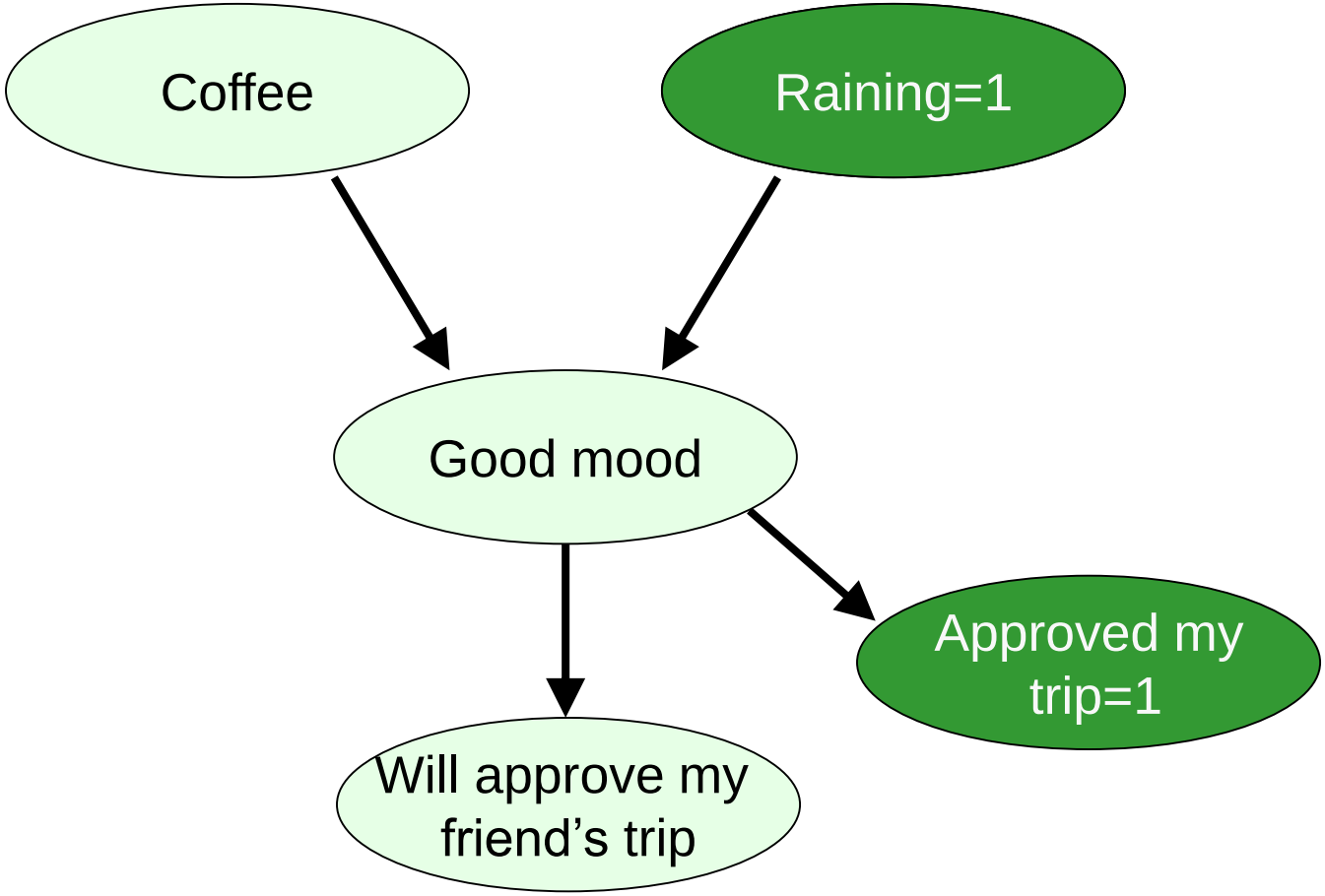
```
% Perform variable elimination on BossPredictor model
% Model specification
PCoffee  = [0.4 0.6];
PRaining = [0.2 0.8];

PGoodMood(2,:,:) = [0.9 0.2;0.9 0.9];
PGoodMood(1,:,:) = 1 - PGoodMood(2,:,:);

PLikesIdea(2,:) = [0.4 0.9];
PLikesIdea(1,:) = 1 - PLikesIdea(2,:);

% Add observation
PRaining = [0 1];

%%% Perform variable elimination
% Eliminate coffee
PGoodMood2 = zeros(2,2);
for coffee=1:2
    PGoodMood2 = PGoodMood2 + squeeze(PGoodMood(:,coffee,:)*PCoffee(coffee));
end

% Eliminate raining
PGoodMood3 = zeros(2,1);
for raining = 1:2
    PGoodMood3 = PGoodMood3 + PGoodMood2(:,raining)*PRaining(raining);
end

% Eliminate good mood
PLikesIdea2 = zeros(2,1);
for goodMood = 1:2
    PLikesIdea2 = PLikesIdea2 + PLikesIdea(:,goodMood)*PGoodMood3(goodMood);
end
PLikesIdea2(2)
```

P(likes idea)=

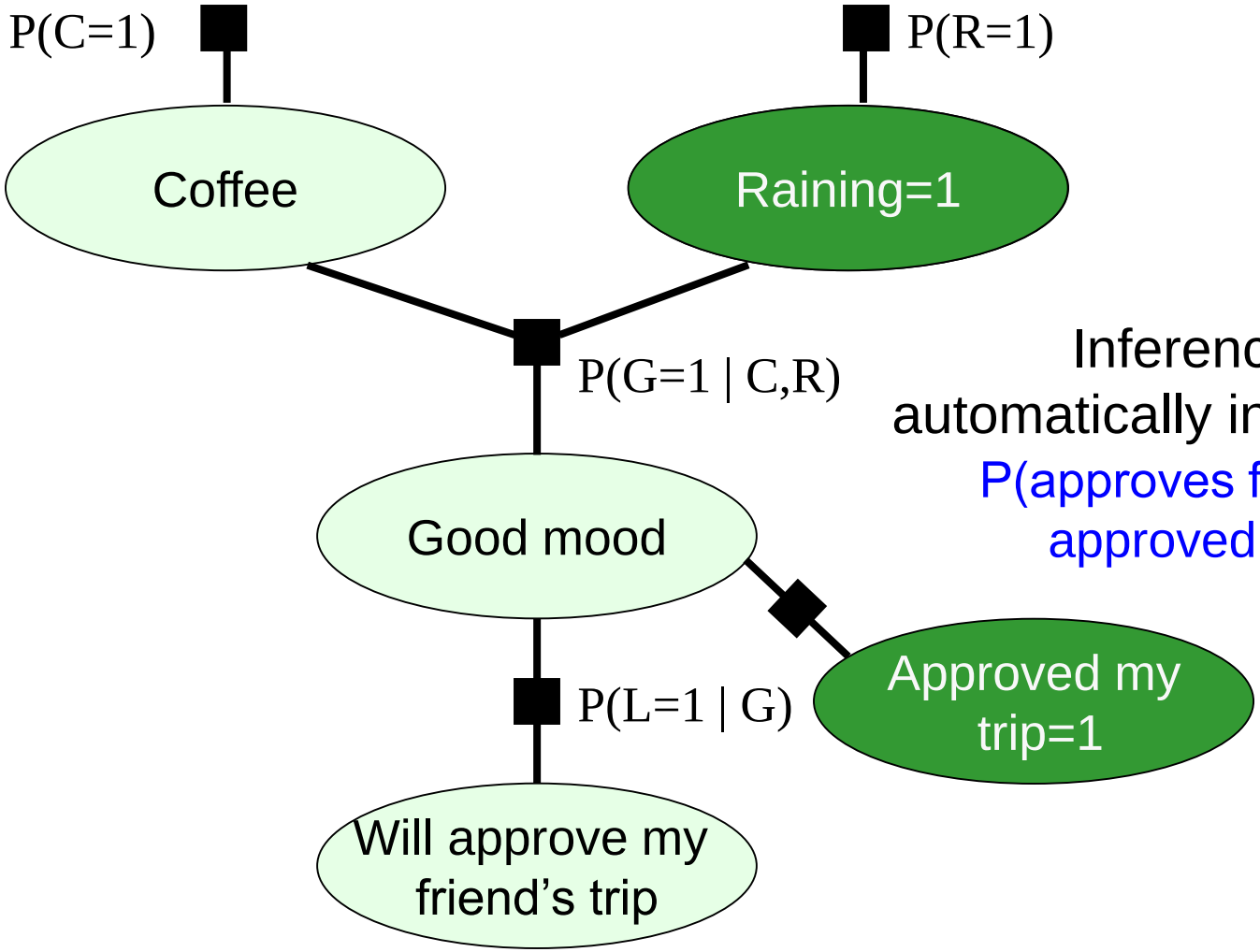
Is raining: 71%

Not raining: 85%

Example: BossPredictor



BossPredictor with Inference Engine



Inference runs automatically in new model:

P(approves friend's trip | approved my trip) = 79%

Existing modelling methods

- Graphical editors/factor graphs
  - Easy to use
  - Hard to develop and maintain large models
  - Hard to integrate with other code
  - Limited scope
- XML
  - Awkward syntax
  - Hard to integrate with other code
  - Limited toolset

CSOFT modelling language

- A representation language for probabilistic models.
- Takes C# and adds support for:
  - random variables
  - constraints on variables
  - inference
- Can be embedded in ordinary C# to allow integration of deterministic + stochastic code

CSOFT – random variables

- Normal variables have a fixed single value.
  e.g. `int length=6,`
  `bool visible=true.`

- Random variables have uncertain value specified by a probability distribution.
  e.g. `int length = random(Uniform(0,10))`
  `bool visible = random(Bernoulli(0.8)).`

- Introduce random operator which means ‘is distributed as’.

CSOFT –constraints

- We can define constraints on random variables, e.g.

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

- The constrain(b) operator means ‘we constrain b to be true’.

CSOFT – 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*.

BossPredictor in CSOFT

Model definition

```
bool coffee = random(Bernoulli(0.6));
bool raining = random(Bernoulli (0.8));
bool goodMood =
        random(Bernoulli((coffee||!raining)?0.9:0.2));
bool approvesTrip = random(Bernoulli(goodMood?0.9:0.4));
```

Constraints and inference

```
constrain(raining==true);
return infer(approvesTrip);
```

TrueSkill™ in CSOFT

TrueSkill model (without draws)

```
double[] skill=new double[nPlayers];
double[] performance=new double[nPlayers];
for (int j=0; j<nPlayers; j++) {
  skill[j]=random(Gaussian(mu[j],sigma[j]));
  double x=random(Gaussian(0, beta));
  performance[j] = skill[j] + x;
  if (j>0) constrain(performance[j-1] >
                     performance[j]);
}
return infer(skill);
```

Microsoft Research Cambridge

CSOFT for analysing existing code

```
int i=random(Uniform(-100,100));
bool b = false;
try {
  Read(i);
} catch (Exception ex) {
  b = true;
```

Existing code called with *random* parameter

```
public byte[] Read(int numBytes) {
    if (numBytes<0) throw new
        ArgumentOutOfRangeException();
                …
}
```

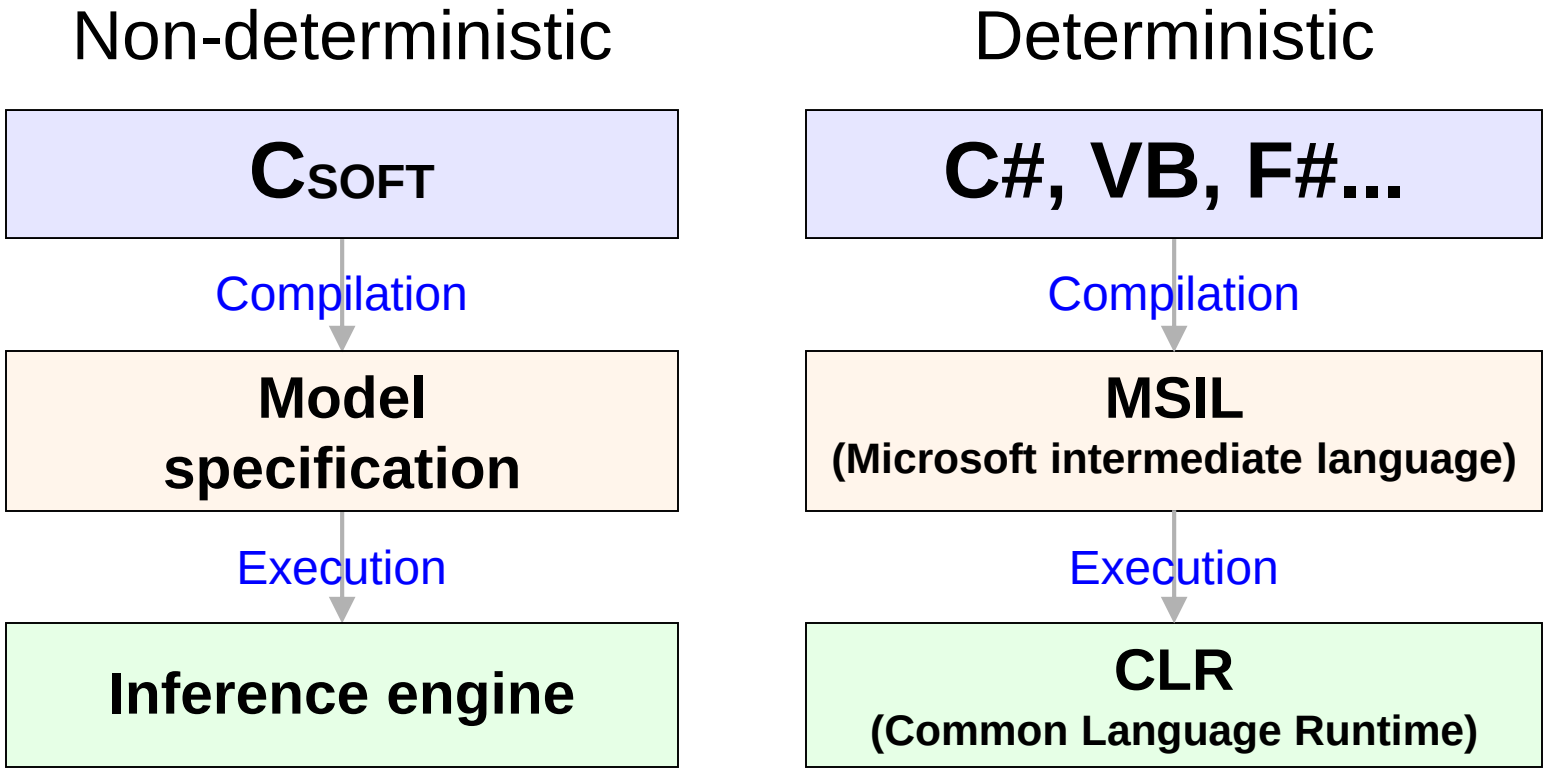
Microsoft Research Cambridge

Random objects

- CSOFT is object-oriented. Random objects are objects whose members are random variables.
- Useful for domain-specific inference: can provide a library of random objects relevant to a domain e.g. `Image`, `ImageOperation`, `Texture`.
- For example: machine vision models can be specified as a series of graphics operations which generate an image.

Implementing CSOFT

CSOFT is implemented as a .NET library. The operators appear as static methods e.g. `Csoft.random()`



Inference engine requirements

To support CSOFT, engine must be:

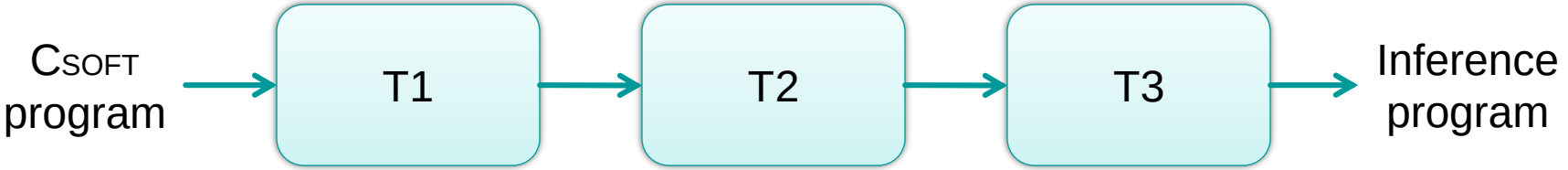
- Flexible: capable of handling very broad range of model specifications
- Accurate: must give appropriately accurate inference results so must use an appropriate inference algorithm
- Efficient: must scale to run on large models with large data sets

Infer.NET version 1

- Flexible: Yes - general purpose architecture for discrete/continuous variables and a large variety of factors
- Accurate: Yes – supported multiple inference algorithms: VMP/EP/Gibbs
- Efficient: No – constructed in-memory factor graphs and traversed them during inference, introducing considerable overhead. Also, made the code very difficult to maintain as more features were added.

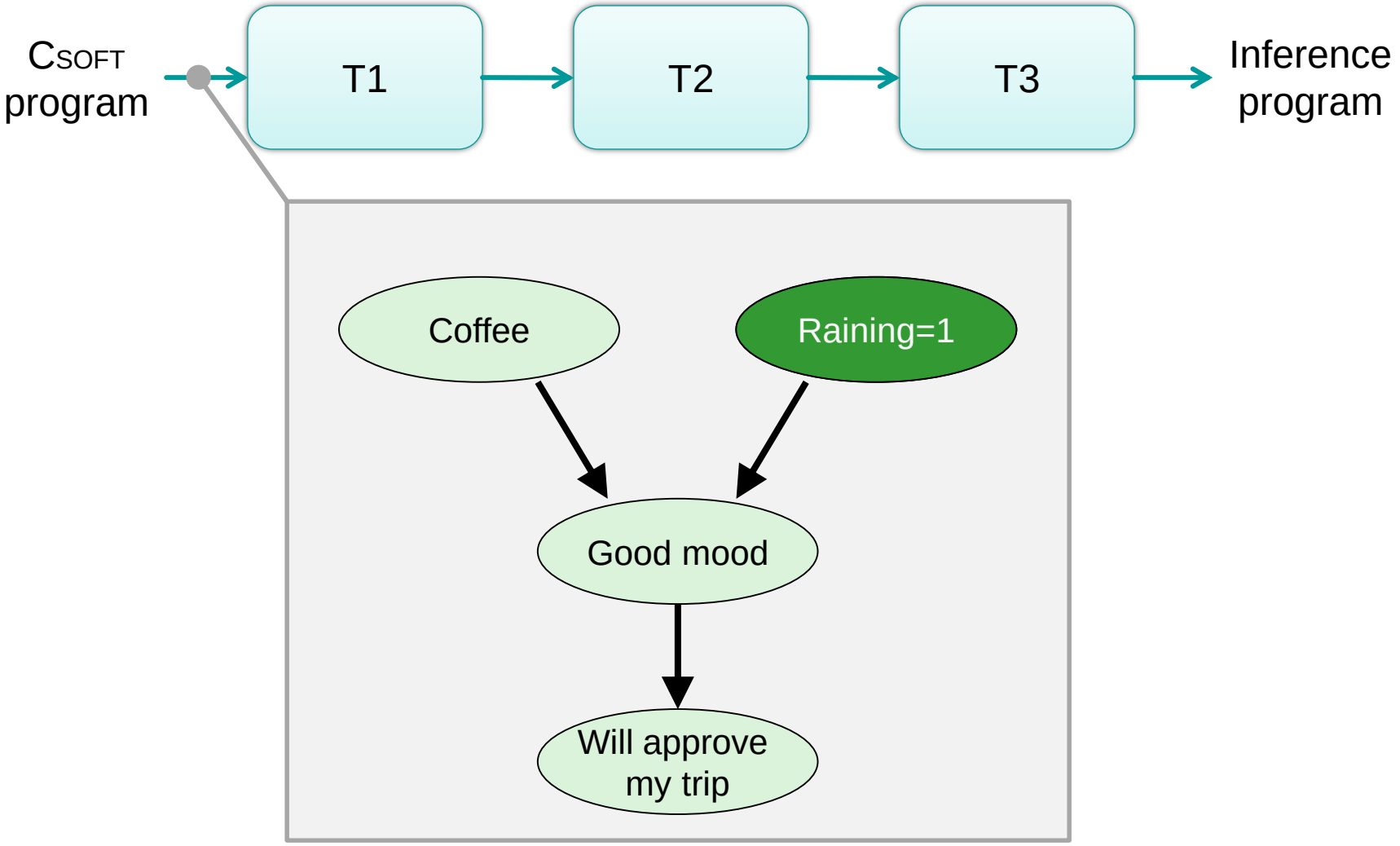
Infer.NET version 2

- Version 2 *compiles* modelling code into inference code.
- No in-memory factor graphs = no overhead
- Consists of a chain of code transformations:

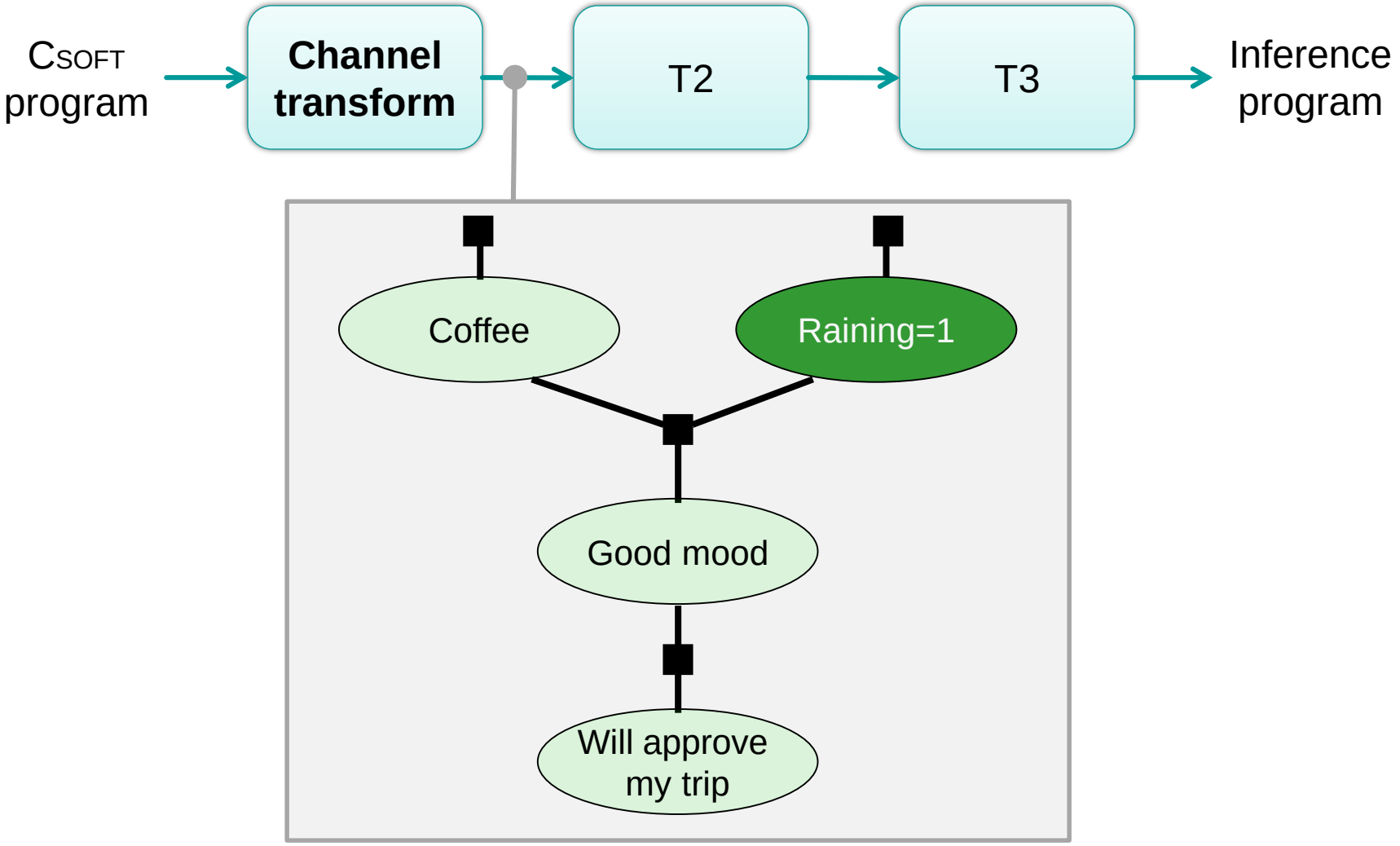


- Each intermediate program is a valid C# program.

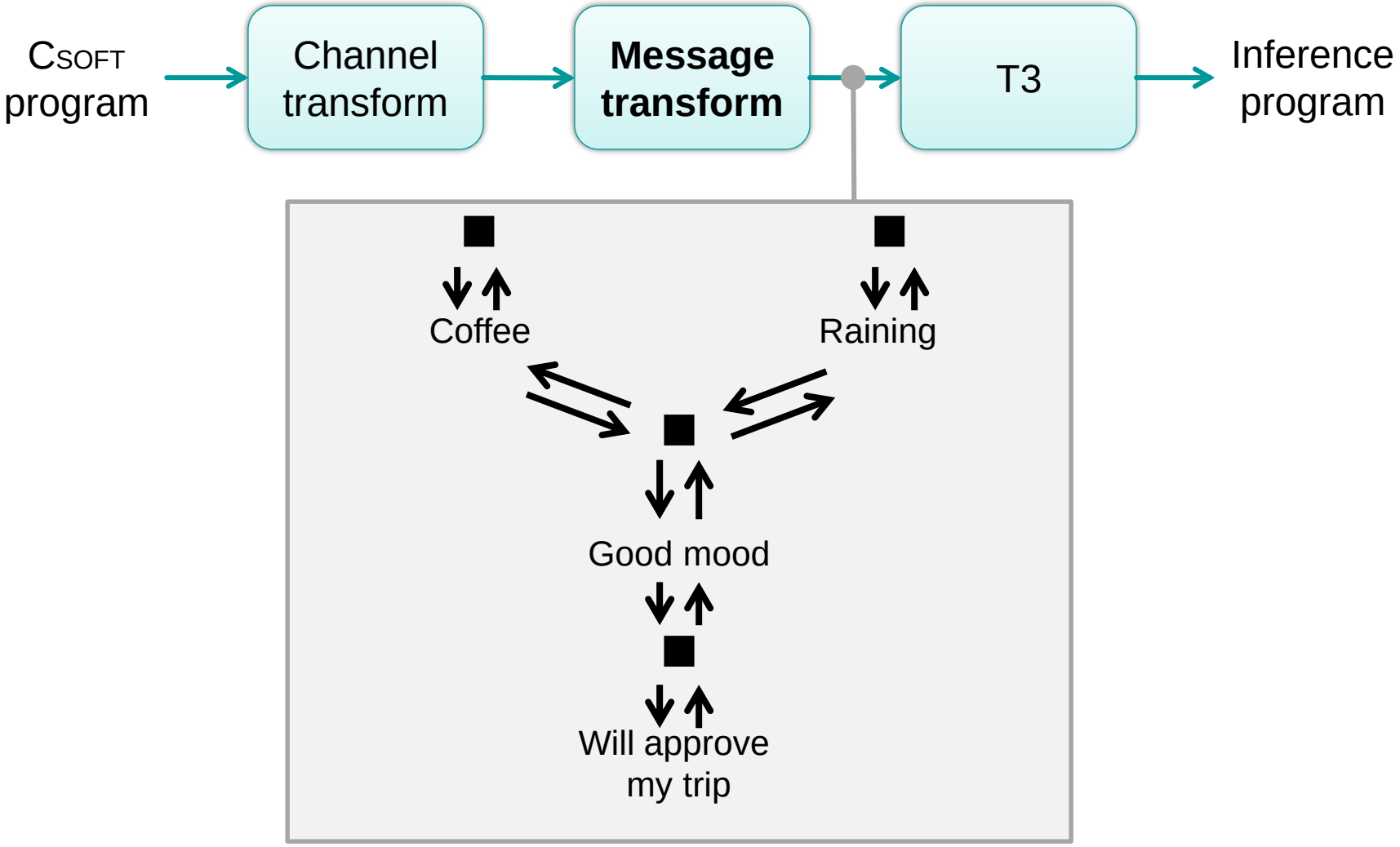
Infer.NET inference engine



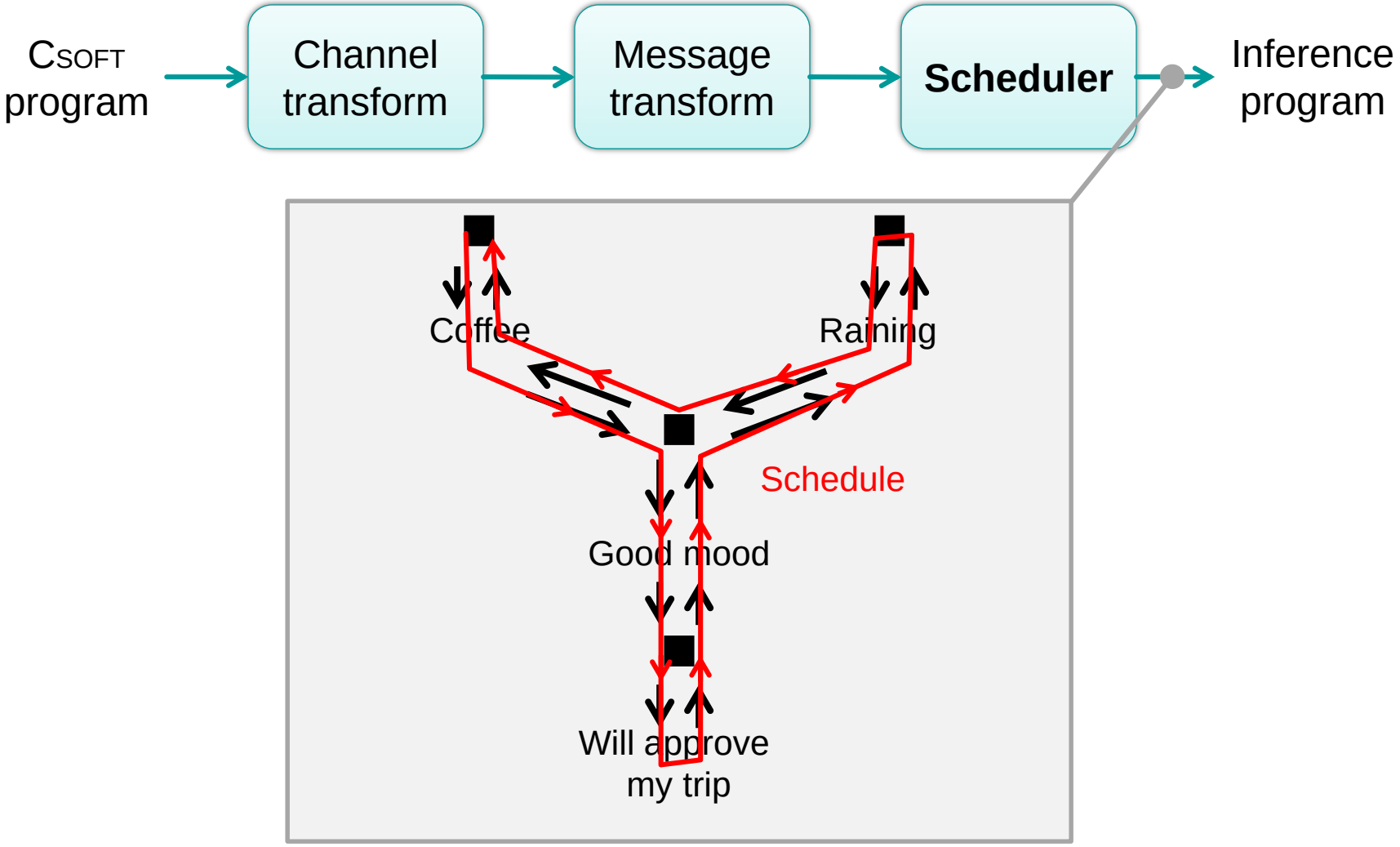
Infer.NET compiler



Infer.NET compiler



Infer.NET compiler



Infer.NET Demo

Advantages of CSOFT + Infer.NET

- **Rich and compact modelling language**
  Wide range of complex models can be represented succinctly.
- **Powerful inference framework**
  Supports multiple inference algorithms, highly customisable.
- **Efficient inference**
  Compilation means almost no overhead.
- **Easy integration**
  Inference can be invoked from .NET with minimal effort.
- **Easy to learn**
  Just the three new operators added to the language

Thanks!

http://research.microsoft.com/infernet

Microsoft Research Cambridge