Probabilistic / evidential reasoning about algorithms

ProbLog

- A probabilistic extension of Prolog
- Created by the Declarative Languages and Artificial Intelligence research group at KU Leuven.
- According to the authors: "…ProbLog is a tool that allows you to intuitively build programs that do not only encode **complex interactions** between a large sets of **heterogenous components** but also the inherent **uncertainties** that are present in real-life situations."
- In this workshop we will use version 2 of ProbLog.

Useful Links on ProbLog 2

- Interactive Tutorial
  - https://dtai.cs.kuleuven.be/problog/tutorial.html
- Online ProbLog Programming Environment (for your own code)
  - https://dtai.cs.kuleuven.be/problog/editor.html

Recap: main syntactic constructs of Prolog

- Facts

it_is_late. (asserts that proposition it_is_late=True)

- Rules

is_alien :- is_clingon. (is_alien=true *if* is_clingon=True)

- Conjunction (and) is expressed using ”,” comma

is_superman :- looks_human, has_superpowers.

- Disjunction (or) is expressed using ”;” semicolon

is_twilight :- before_sunrise ; after_sunset.

- Negation (not) is expressed using ”\+”

is_unhappy :- \+ is_happy.

Probabilistic extensions of ProbLog

- "Probabilistic" facts:

```
0.5::coin_heads.   % coin_heads=True with probability 0.5
```

- Probabilistic rules (a.k.a. "intuitionistic" facts):

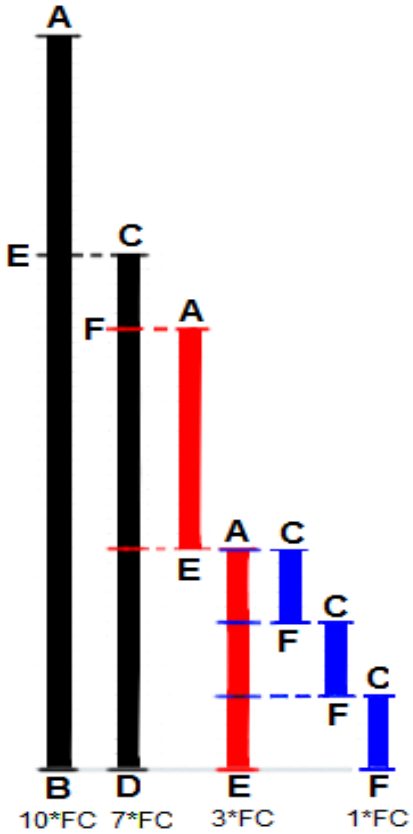
```
0.7::breathalyzer_alert :- drunk.    % if drunk=True, then
breathalyzer_alert=True with probability 0.7
```

- Probabilistic disjunctions (specify probability distribution for a set of mutually exclusive events rather than a signle true/false fact):

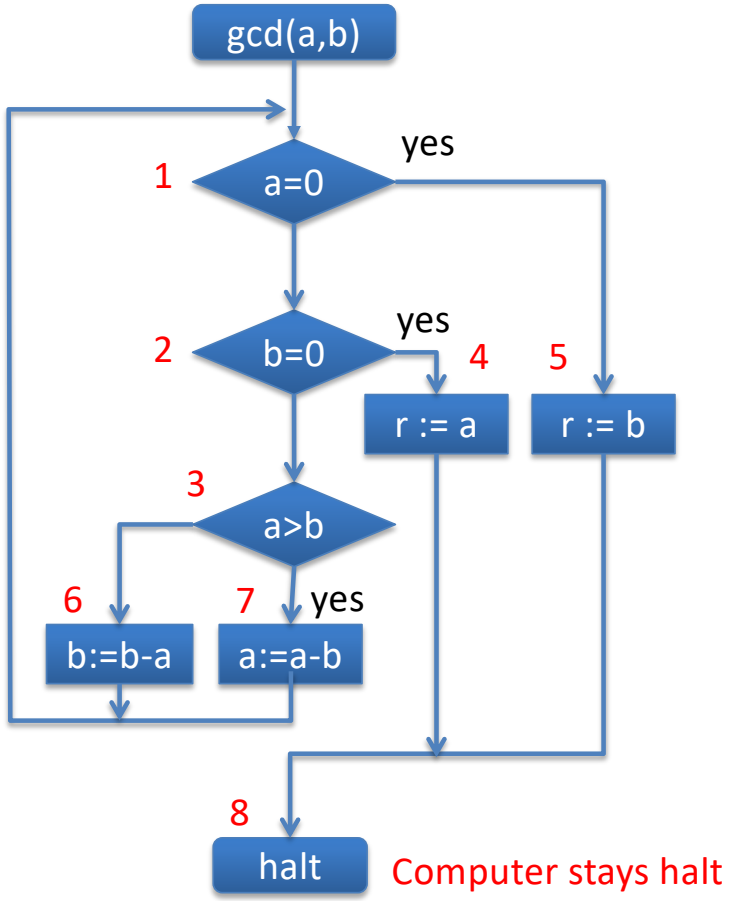
```
1/6::dice(1); 1/6::dice(2); 1/6::dice(3); 1/6::dice(4); 1/6::dice(5);
1/6::dice(6).  % only one of dice(x) events will occur!
```

Reasoning about algorithms

- gcd(a,b) – largest number that divides both a and b



Euclid's example

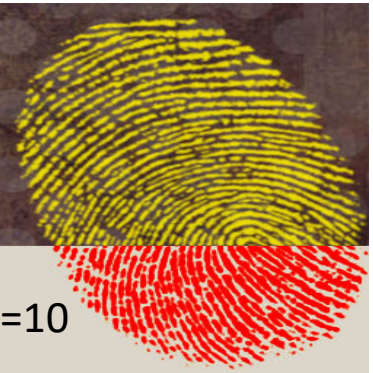

gcd(a,b)
1
a=0
yes
2
b=0
yes
4
5
r := a
r := b
3
a>b
6
7
yes
b:=b-a
a:=a-b
8
halt
Computer stays halt

gcd in ProbLog

```
gcd(A,B,Result) :- A =< 0, Result = B.
gcd(A,B,Result) :- B =< 0, Result = A.
gcd(A,B,Result) :- A > 0, B > 0, A>=B, NewA is A-B, gcd(NewA,B,Result).
gcd(A,B,Result) :- A > 0, B > 0, A<B, NewB is B-A, gcd(A,NewB,Result).

query(gcd(10,5,Result)).
```

Specifying probability distribution of initial states



```
% Probability distribution of A: 1<=A<=10
0.1::a(1);
0.1::a(2);
0.1::a(3);
0.1::a(4);
0.1::a(5);
0.1::a(6);
0.1::a(7);
0.1::a(8);
0.1::a(9);
0.1::a(10).
```

```
% Probability distribution of B: 1<=B<=10
0.1::b(1);
0.1::b(2);
0.1::b(3);
0.1::b(4);
0.1::b(5);
0.1::b(6);
0.1::b(7);
0.1::b(8);
0.1::b(9);
0.1::b(10).

model(A,B,Result) :- a(A),b(B),gcd(A,B,Result).
query(model(A,B,Result)).
```

Resulting distribution

Query▼	Location	Probability
model(1,1,1)	32:7	0.01
model(1,2,1)	32:7	0.01
model(1,3,1)	32:7	0.01
model(1,4,1)	32:7	0.01
model(1,5,1)	32:7	0.01
model(1,6,1)	32:7	0.01
model(1,7,1)	32:7	0.01
model(1,8,1)	32:7	0.01

Querying probability of getting specific outputs

To get a distribution of specific algorithm outputs, we need to define a helper predicate (a ProbLog limitation):

result(Result) :- model(_,_,Result).

query(result(Result)).

Resulting distribution

Query▼	Location	Probability
result(1)	33:7	0.63
result(2)	33:7	0.19
result(3)	33:7	0.07
result(4)	33:7	0.03
result(5)	33:7	0.03
result(6)	33:7	0.01
result(7)	33:7	0.01
result(8)	33:7	0.01
result(9)	33:7	0.01
result(10)	33:7	0.01

Querying probability of input given evidence of output

To calculate a posterior distribution of algorithm inputs, we need to define an additional helper predicate inputs and specify evidence using built-in **evidence()** predicate:

```
result(Result) :- model(_,_,Result).
Inputs(A,B) :- model(A,B,_).
evidence(result(5)).
query(inputs(A,B)).
```

Resulting distribution

inputs(5,1)	36:7	0
inputs(5,2)	36:7	0
inputs(5,3)	36:7	0
inputs(5,4)	36:7	0
inputs(5,5)	36:7	0.33333333
inputs(5,6)	36:7	0
inputs(5,7)	36:7	0
inputs(5,8)	36:7	0
inputs(5,9)	36:7	0
inputs(5,10)	36:7	0.33333333
inputs(6,1)	36:7	0
inputs(6,2)	36:7	0

Exercise: try querying posterior probability of parameter A probability distribution

query(a(A)).

Resulting distribution

Query▼	Location	Probability
a(1)	36:7	0
a(2)	36:7	0
a(3)	36:7	0
a(4)	36:7	0
a(5)	36:7	0.66666667
a(6)	36:7	0
a(7)	36:7	0
a(8)	36:7	0
a(9)	36:7	0
a(10)	36:7	0.33333333

query(a(A))

Initial probabilities:

100 possible initial states determined by **model(A,B,Result) :- a(A),b(B),gcd(A,B,Result).**

What is the probability that parameter A was 5 at T=0? (0.01+0.01)/0.03 = 0.666666(6)

Further reading

- "Learn Prolog Now!"
  - http://www.learnprolognow.org
  - A very good and comprehensive Intro to programming in Prolog.

- ProbLog tutorials
  - https://dtai.cs.kuleuven.be/problog/tutorial.html

Optional self-practice exercise

- Try to modify our model so that instead of GCD it calculates a the sum of natural numbers from 1 to n for the given n.

- Contact me if you would like to get an answer! My email is on http://dfire.ucd.ie/