COMP3220: Document Processing and Semantic Technologies Combining Rules with Probabilities

Rolf Schwitter

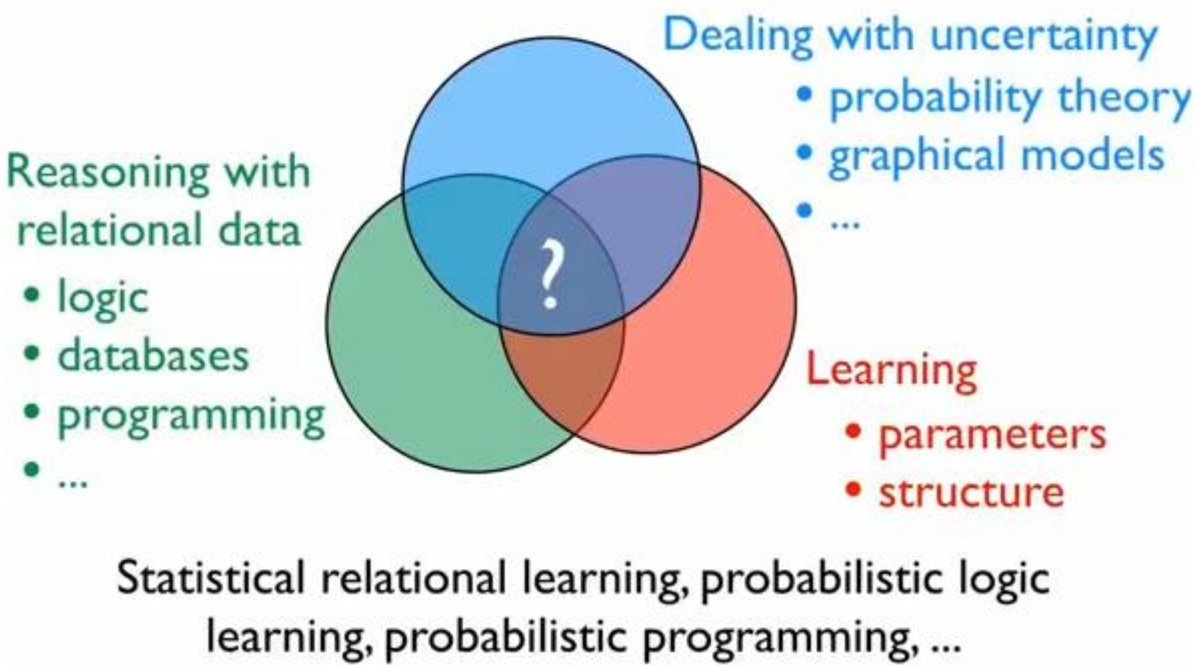
Rolf.Schwitter@mq.edu.au

Today’s Agenda

- Statistical Relational Learning
- A Motivating Example: Gambling
- Logic Programs with Annotated Disjunctions
  - MPE Inference
  - Marginal Probability
  - Conditional Probability
- Working with cplint on SWISH
- Exact and Approximate Inference
- Parameter and Structure Learning

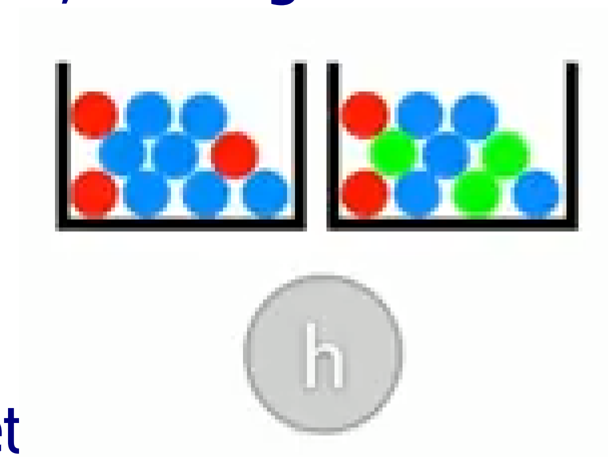
A Key Question in Artificial Intelligence

Statistical Relational Learning



A Motivating Example: Gambling

- Toss a biased coin and draw a ball from each of two urns.
- The first urn contains three red and seven blue balls; the second urn two red, three green and five blue balls.



- You win if you get
  – heads and a red ball or
  – two balls of the same colour.

How to Modell the Problem

- Random variables are represented as probabilistic facts.
- Probabilistic fact for the biased coin: **`heads`** is true with probability **`0.4`** and false with **`0.6`**.

```
heads:0.4.
```

How to Modell the Problem

- Annotated disjunctions:
  - first ball is **`red`** with probability of **`0.3`** and blue with **`0.7`**.
  - second ball is **`red`** with probability of **`0.2`**, **`green`** with **`0.3`** and **`blue`** with **`0.5`**.

```
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ;
col(2,blue):0.5.
```

How to Modell the Problem

- Logic rules encode the background knowledge about what it means to win:

```
% You win if you get heads and a red ball.
win :- heads, col(_,red).

% You win if you get two balls of the same
% colour.
win :- col(1,C), col(2,C).
```

Logic Program with Annotated Disjunctions

```
heads:0.4.
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ;
col(2,blue):0.5.
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

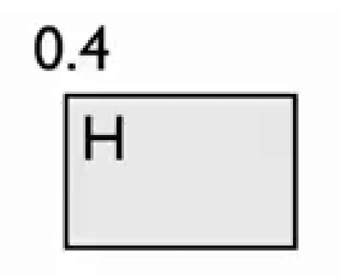
Possible Worlds

- This program generates possible worlds.
- We start with the empty possible world and go through all possible choices that we can make.

- The empty possible world:

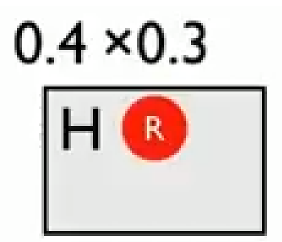
Logic Program with Annotated Disjunctions

```
heads:0.4.
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ;
col(2,blue):0.5.
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```



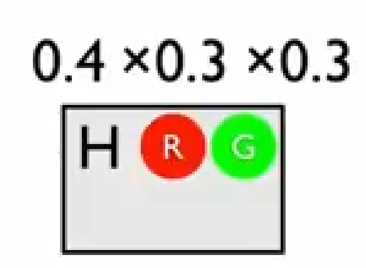
Logic Program with Annotated Disjunctions

```
heads:0.4.
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ;
col(2,blue):0.5.
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```



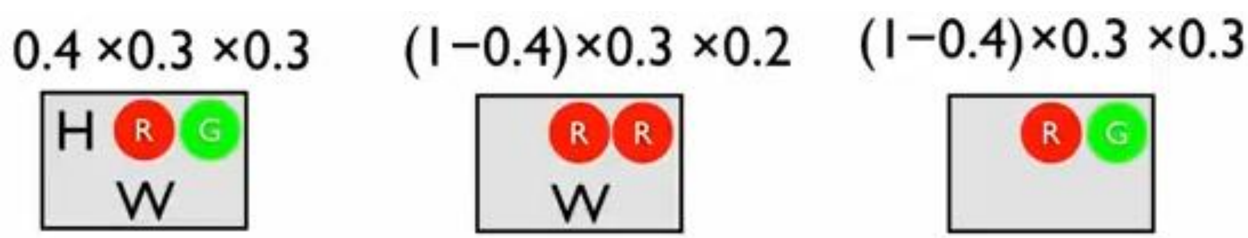
Logic Program with Annotated Disjunctions

```
heads:0.4.
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ;
col(2,blue):0.5.
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

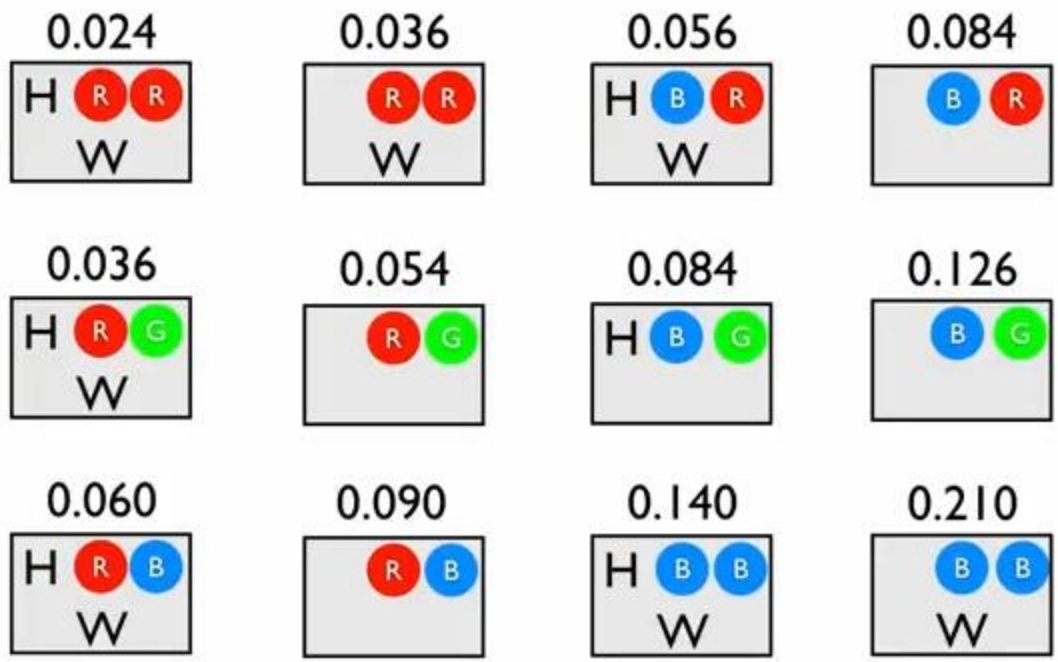


Logic Program with Annotated Disjunctions

```
heads:0.4.
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ;
col(2,blue):0.5.
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```



All Possible Worlds

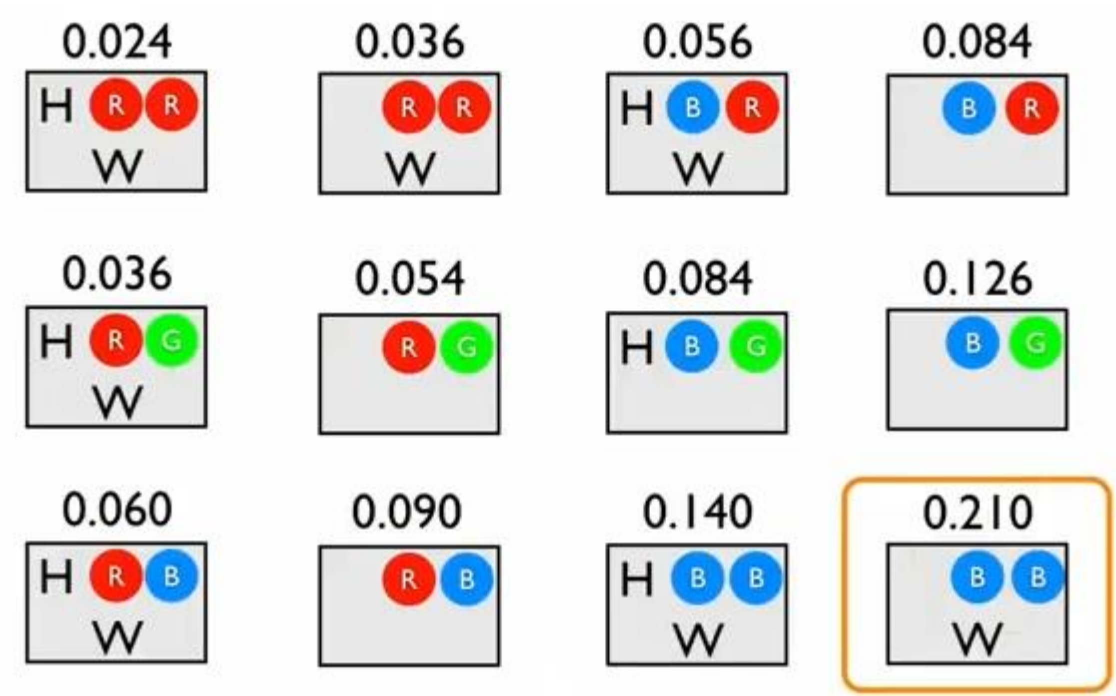


Questions

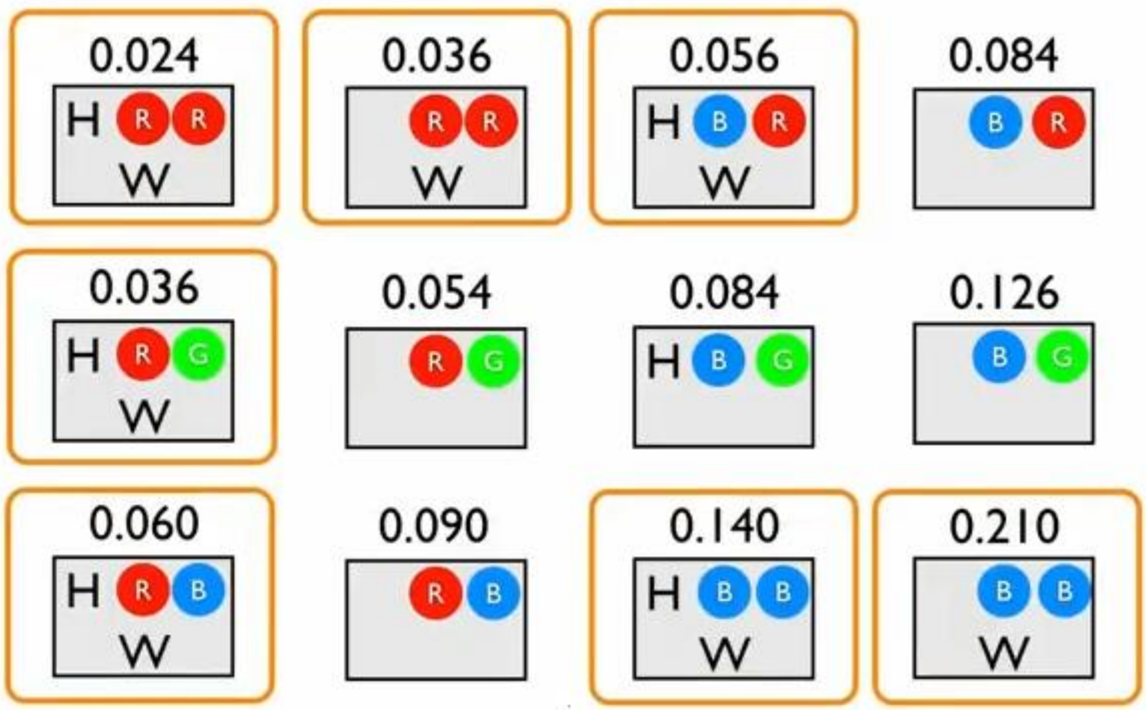
- What's the most probable world where **`win`** is true?
  -> MPE[1] Inference
- What's the probability of **`win`**?
  -> Marginal Probability
- What's the probability of **`win`** given **`col(2,green)`**?
  -> Conditional Probability

[1] MPE = Most Probable Explanation

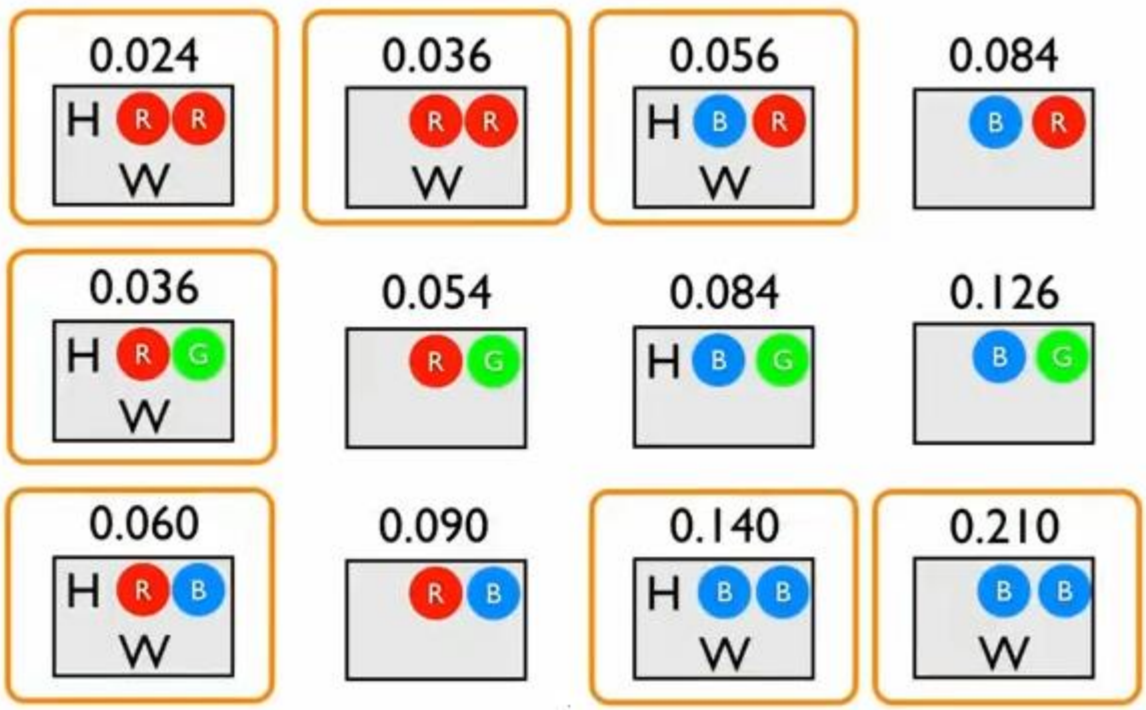
MPE Inference



Marginal Probability

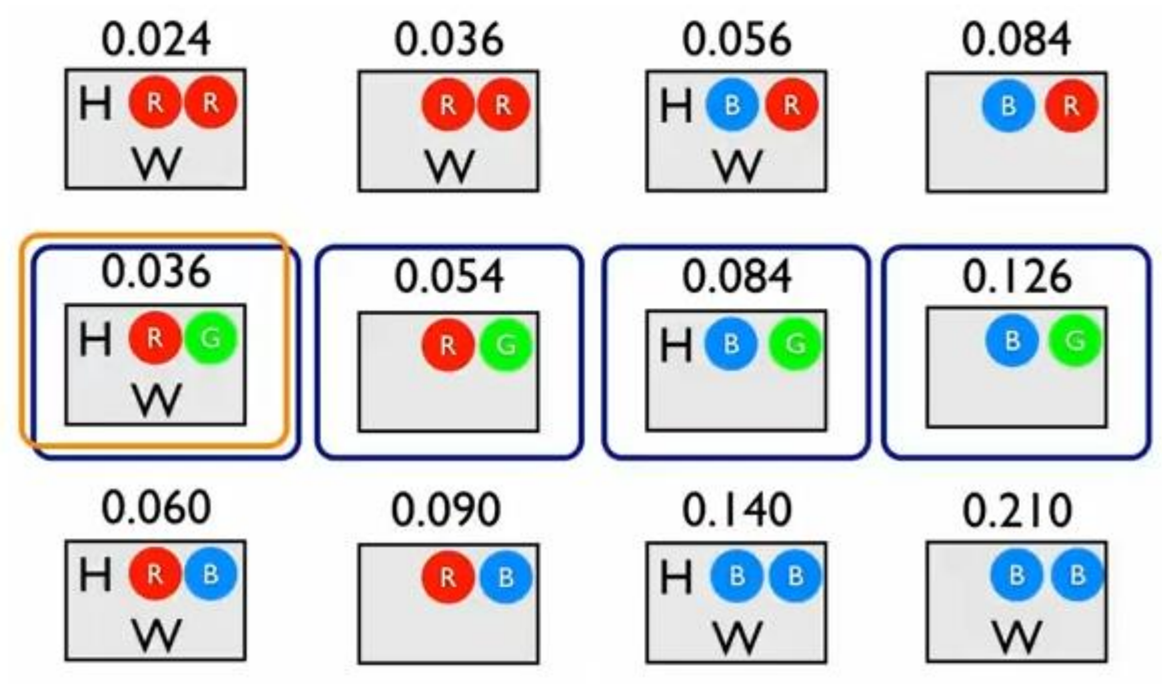


Marginal Probability

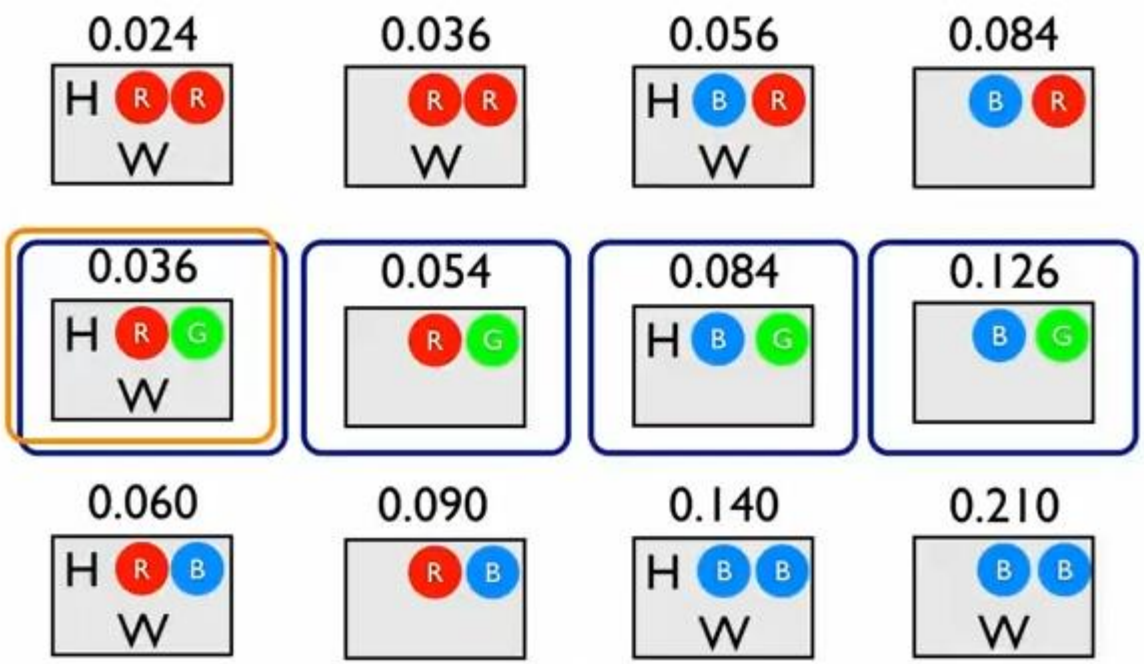


$P(win) = \sum = \mathbf{0.562}$

Conditional Probability



Conditional Probability

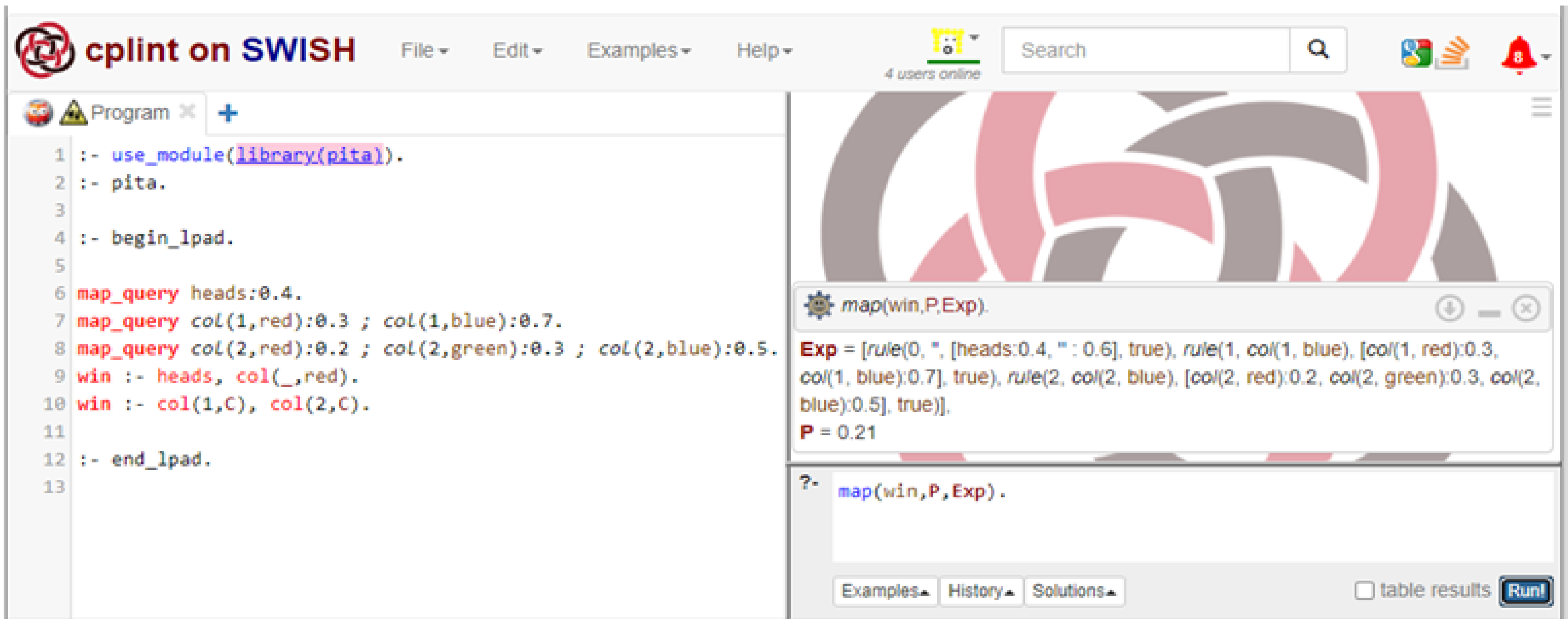


$$P(win|col(2,green)) = P(win \wedge col(2,green))/P(col(2,green)) = 0.036/0.3 = 0.12$$

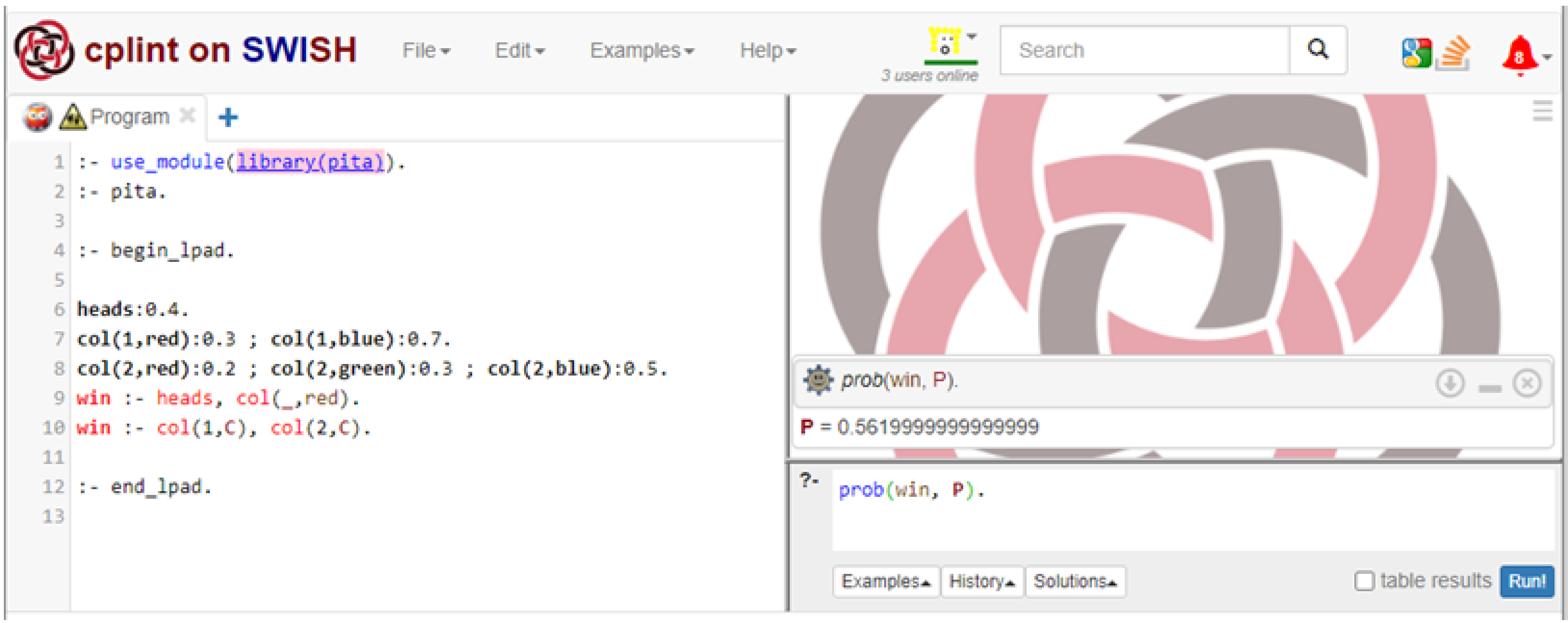
Working with cplint on SWISH

- cplint is a suite of programs for inference and learning of Logic Programs with Annotated Disjunctions.
- cplint is based on SWI Prolog, a logic programming language:
  `http://friguzzi.github.io/cplint/_build/html/index.html`
- cplint can be used for asking queries using exact inference and approximate inference.
- SWISH is a web framework for logic programming.
- You can find cplint on SWISH here:
  `http://cplint.lamping.unife.it/`

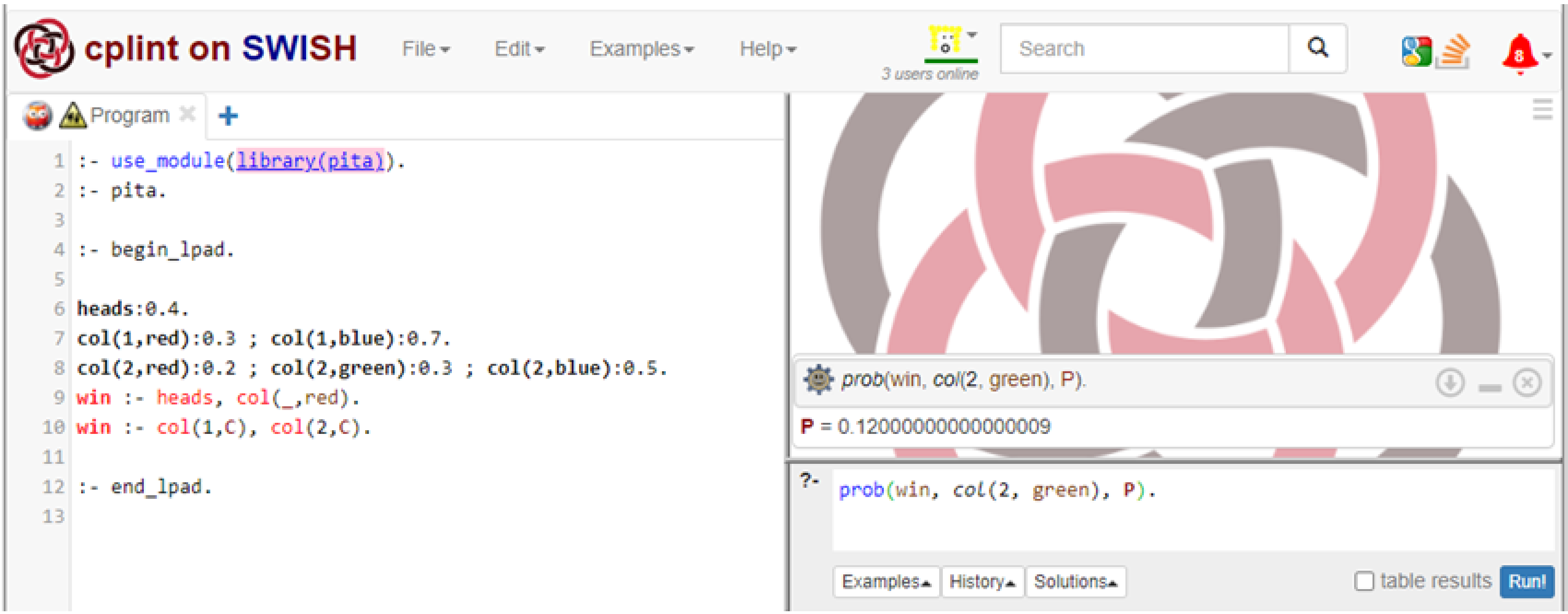
Example: Most Probable Explanation



Example: Marginal Probability



Example: Conditional Probability



Example: Code

```
:- use_module(library(pita)).
:- pita.

:- begin_lpad.

heads:0.4.
col(1,red):0.3 ; col(1,blue):0.7.
col(2,red):0.2 ; col(2,green):0.3 ; col(2,blue):0.5.
win :- heads, col(_,red).
win :- col(1,C), col(2,C).

:- end_lpad.
```

Explanation

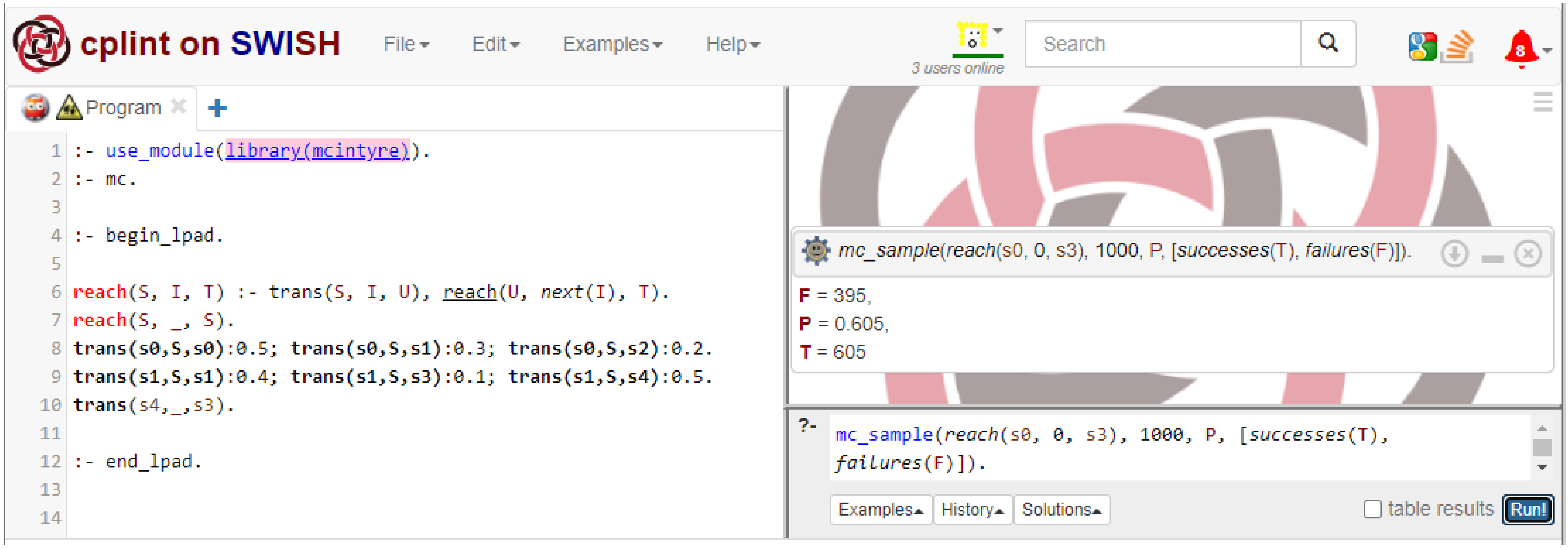
- The library **`pita`** is used to perform <u>exact</u> inference.
- This library is imported with the built-in predicate **`use_module/1`**.
- The actual code of the LPAD program is enclosed within two directives:

```
:- begin_lpad.
% Code goes here …
:- end_lpad.
```

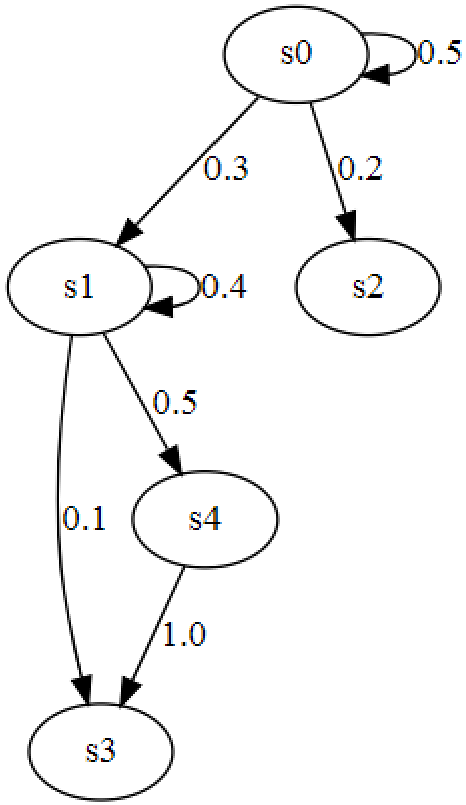
- We use a conditional query:

```
?- prob(win, col(2,green), P).
```

Example: Approximate Inference



Markov Chain



Example: Code

```
:- use_module(library(mcintyre)).
:- mc.

:- begin_lpad.

reach(S, I, T) :- trans(S, I, U), reach(U, next(I), T).
reach(S, _, S).
trans(s0,S,s0):0.5; trans(s0,S,s1):0.3; trans(s0,S,s2):0.2.
trans(s1,S,s1):0.4; trans(s1,S,s3):0.1; trans(s1,S,s4):0.5.
trans(s4,_,s3).

:- end_lpad.
```

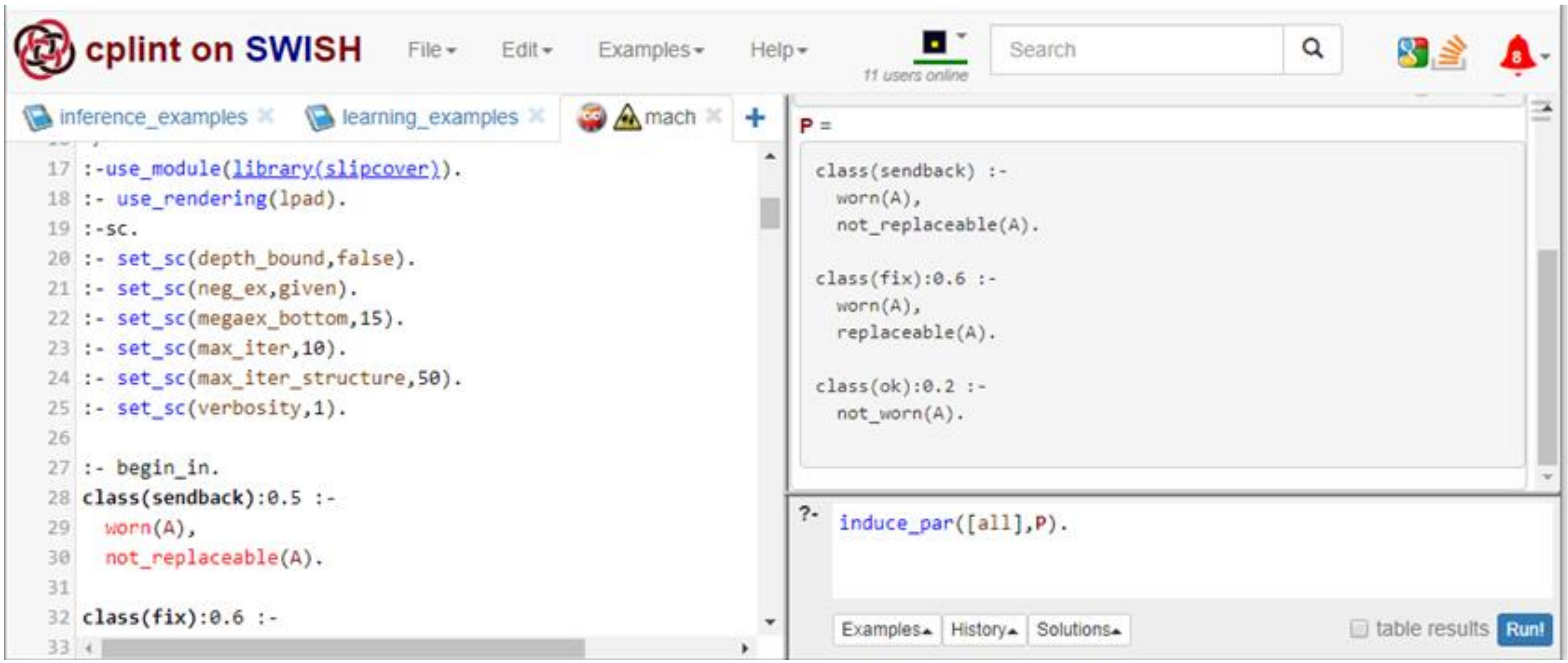
Explanation

- We want to know what is the likelihood that on an execution of a Markov chain from a start state 's', a final state 't' will be reached?
- The transitions between states can be modelled with the predicate **`reach/3`**.
- Starting at state **`S`** at instant **`I`**, state **`T`** is reachable if there is a transition from state **`S`** to state **`U`** at instant **`I`** and if **`T`** is reachable from **`U`** at the next instant.

Explanation

- The chains may be infinite so the query may have an infinite number of explanations.
- Therefore, we use MCINTYRE, a module that performs approximate inference with Monte Carlo sampling.
- Our query samples the predicate **`reach(s0,0,s3)`** **`1000`** times and returns in **`T`** the number of successes, in **`F`** the number of failures and in **`P`** the estimated probability.

Parameter Learning



Explanation

- Parameters for clauses are learned from observations.
- Initial program with default parameters in red:

```
:- begin_in.
class(sendback):0.5 :-
  worn(A),
  not_replaceable(A).
class(fix):0.5 :-
  worn(A),
  replaceable(A).
class(ok):0.5 :-
  not_worn(_A).
:- end_in.
```

Explanation

```
% Background theory
:- begin_bg.
  component(C):-  replaceable(C).
  component(C):- not_replaceable(C).
  replaceable(gear).
  replaceable(wheel).
  replaceable(chain).
  not_replaceable(engine).
  not_replaceable(control_unit).
  not_worn(C):- component(C), \+ worn(C).
  one_worn:- worn(_).
  none_worn:- \+ one_worn.
:- end_bg.
```

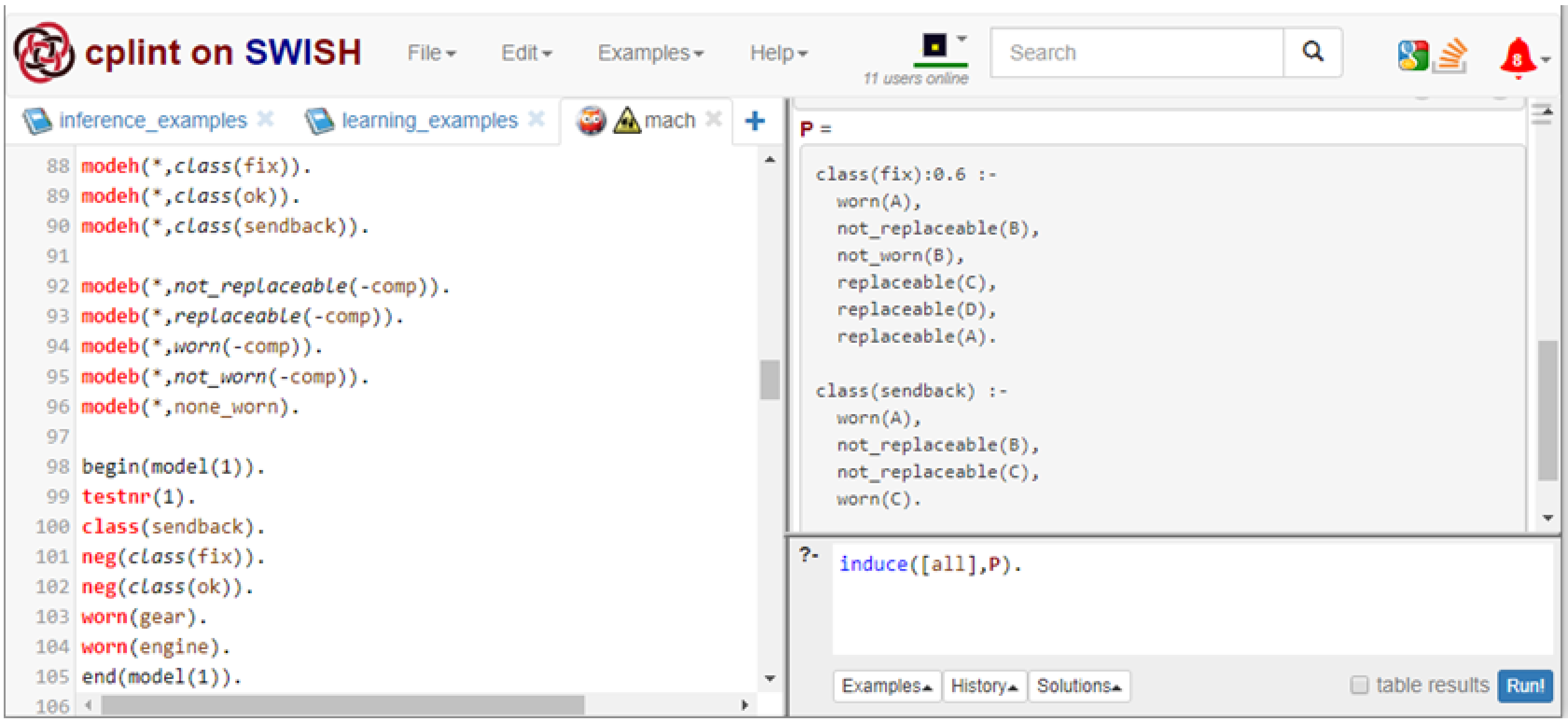
Explanation

- Here is an example of an observation with positive and negative information.

```
% We have to send back a car if the gear is worn and
% the engine is worn.

begin(model(1)).
    class(sendback).
    neg(class(fix)).
    neg(class(ok)).
    worn(gear).
    worn(engine).
 end(model(1)).
```

Structure Learning



Explanation

- Structure learning is more complex than parameter learning (it actually subsumes parameter learning).
- Structure learning requires mode declarations that specify what goes into the head and body of a rule:

```
modeh(*,class(fix)).
modeh(*,class(ok)).
modeh(*,class(sendback)).

modeb(*,not_replaceable(-comp)).
modeb(*,replaceable(-comp)).
modeb(*,worn(-comp)).
modeb(*,not_worn(-comp)).
modeb(*,none_worn).
```

Take-Home Messages

- Combining rules with probabilities is an emerging field of artificial intelligence, known as statistical relational learning.
- Probabilistic logic programming introduces probabilistic reasoning in logic programs in order to represent uncertain information.
- Reasoning can be exact or approximate.
- Parameters of clauses for probabilistic logic programs can be learned from observations.
- Entire structure of clauses for probabilistic logic programs can be learned from observations and mode declarations.