A Course Project for “Computational Pragmatics” in SS 2020 at Osnabrück University

Mastermind

A Pragmatic Model in WebPPL

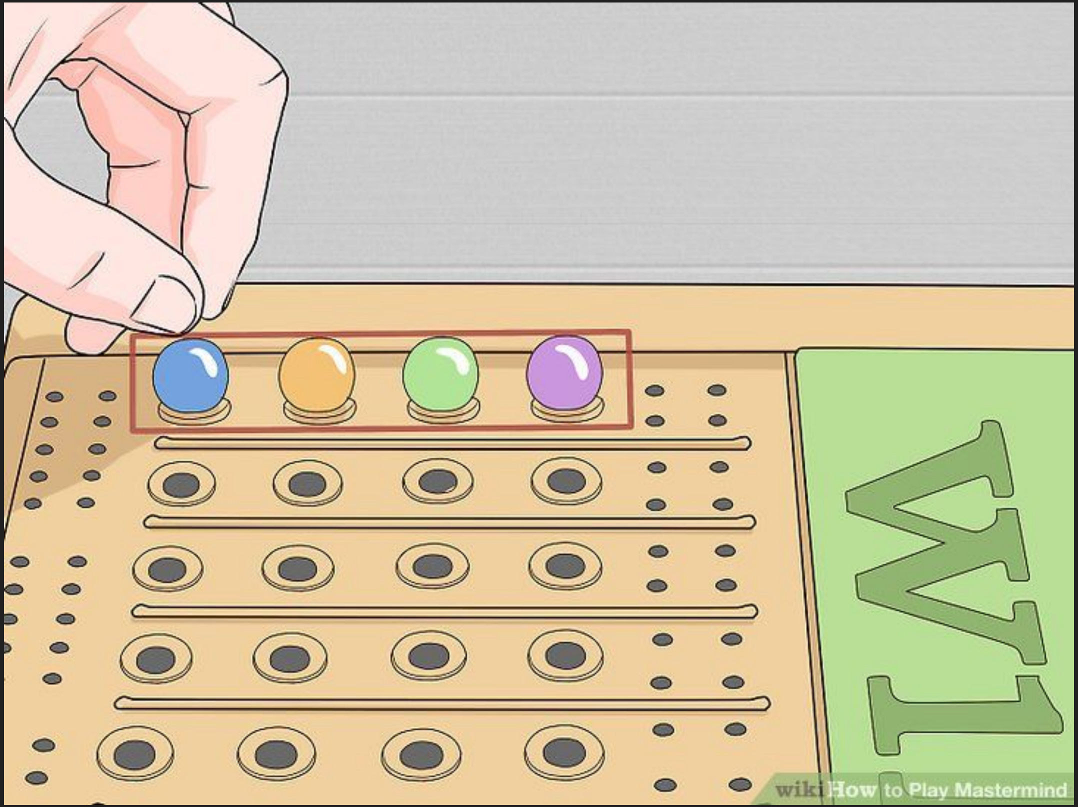
by mpoemsl & rakrueger

Episode I: Game Explanation

How Mastermind Works

Listener must make predictions about the true state

Example Predictions:
[0, 2, 1, 3],
[1, 0, 2, 1],
[0, 0, 0, 0]



Speaker must reply to predictions with true utterances

Example Utterances:
many, some, none, ... (are correct)

An Example Mastermind Match

States
are restricted
to 2 colors and
4 pins in this
example

Round 1:
Listener predicts 1,0,0,0
Speaker says “some”

Round 2:
Listener predicts 0,1,1,0
Speaker says “some”

Round 3:
Listener predicts 0,0,1,1
Speaker says “some”

Round 4:
Listener predicts 1,1,1,1
Speaker says “some”

Round 5:
Listener predicts 0,1,0,1
Speaker says “many”

Round 6:
Listener predicts 1,1,0,1
Speaker says “some”

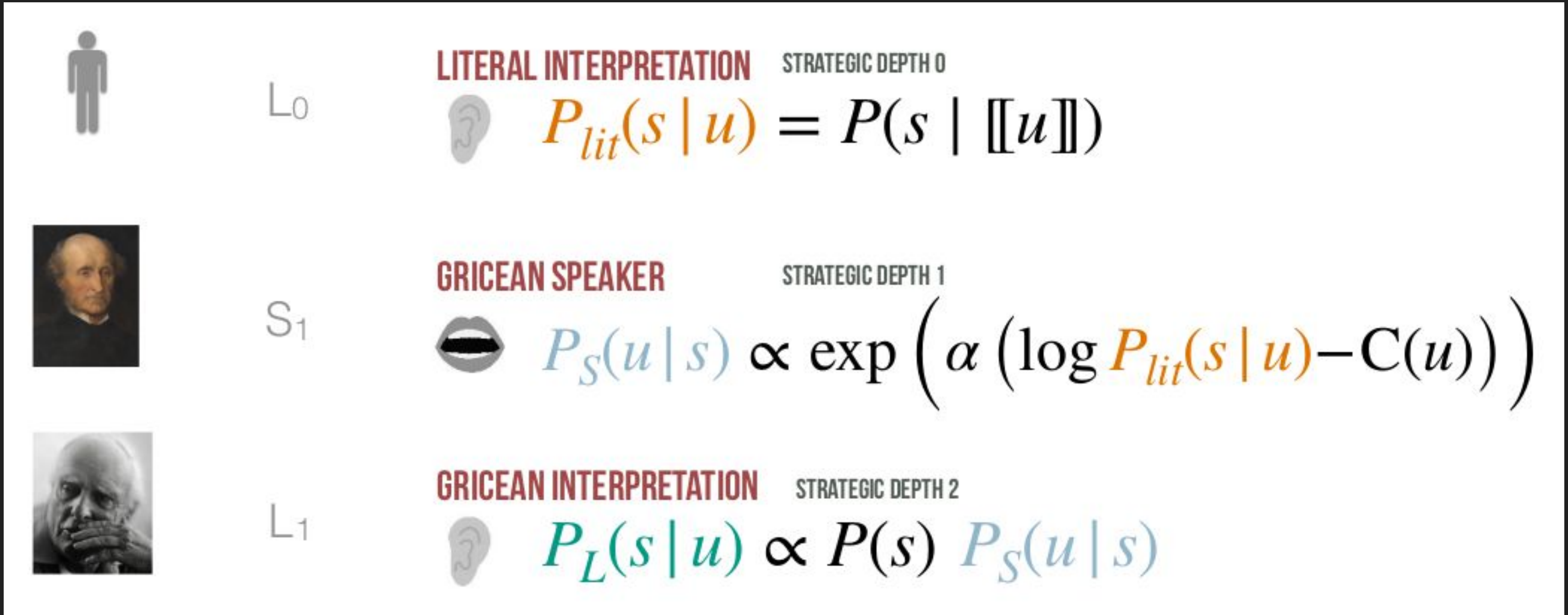
Round 7:
Listener predicts 0,0,0,1
Speaker says some

Round 8:
Listener predicts 0,1,0,0
Correct! Game finished

True State
is 0,1,0,0
in this
example

Episode II: Model Overview

Rational Speech Act (RSA) Framework



LITERAL INTERPRETATION STRATEGIC DEPTH 0

$$P_{lit}(s \mid u) = P(s \mid [\![u]\!])$$

GRICEAN SPEAKER STRATEGIC DEPTH 1

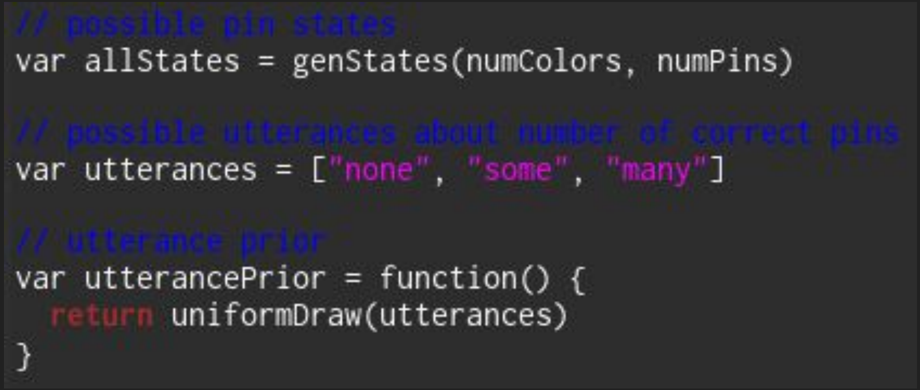
$$P_S(u \mid s) \propto \exp\left(\alpha\left(\log P_{lit}(s \mid u) - \mathrm{C}(u)\right)\right)$$

GRICEAN INTERPRETATION STRATEGIC DEPTH 2

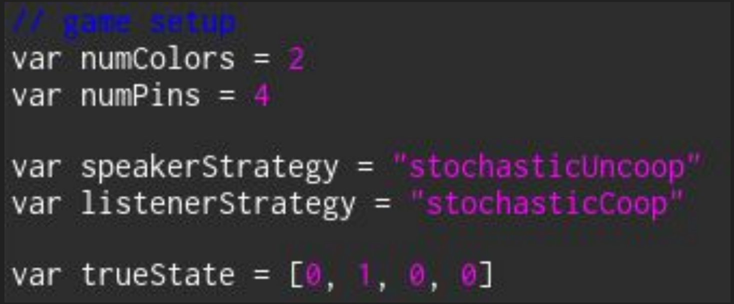
$$P_L(s \mid u) \propto P(s)\, P_S(u \mid s)$$

Pragmatic Mastermind Simulation in WebPPL: github.com/mpoemsl/mastermind

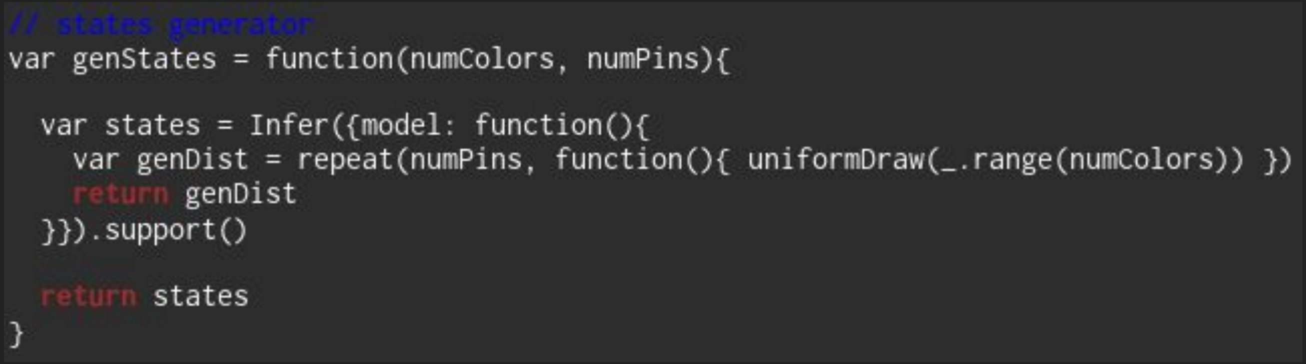
Code Structure 1: Preliminaries



```
// possible pin states
var allStates = genStates(numColors, numPins)

// possible utterances about number of correct pins
var utterances = ["none", "some", "many"]

// utterance prior
var utterancePrior = function() {
  return uniformDraw(utterances)
}
```



```
// game setup
var numColors = 2
var numPins = 4

var speakerStrategy = "stochasticUncoop"
var listenerStrategy = "stochasticCoop"

var trueState = [0, 1, 0, 0]
```

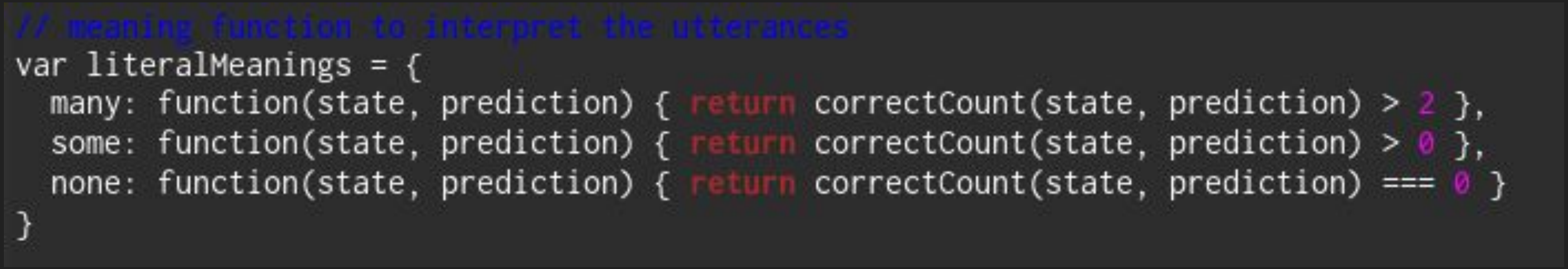


```
// states generator
var genStates = function(numColors, numPins){

  var states = Infer({model: function(){
    var genDist = repeat(numPins, function(){ uniformDraw(_.range(numColors)) })
    return genDist
  }}).support()

  return states
}
```

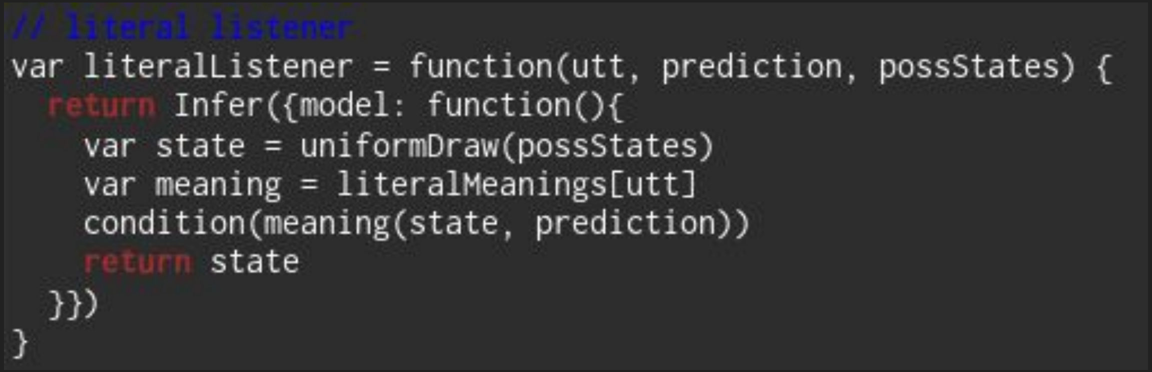
Strategies:
"greedyUncoop"
"stochasticUnoop"
"stochasticCoop"
"greedyCoop"

where "coop" means "as few rounds as possible"

Code Structure 2: Meaning & Literal Listener



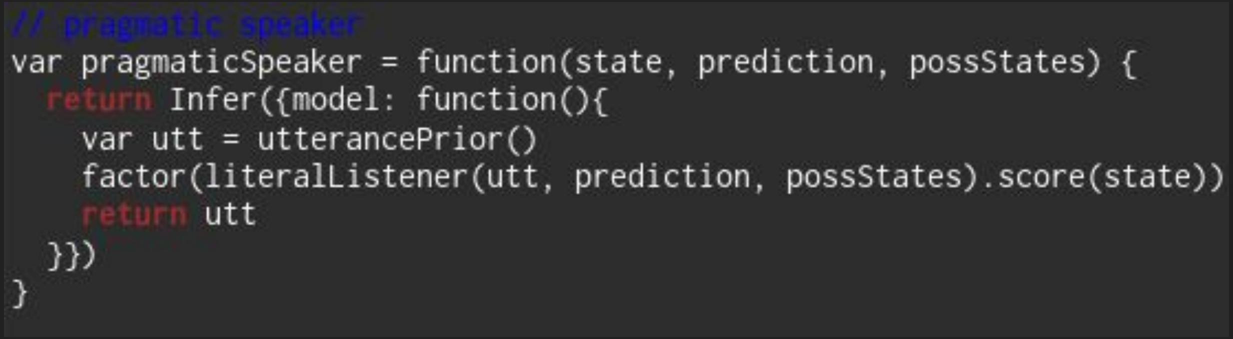
```
// meaning function to interpret the utterances
var literalMeanings = {
  many: function(state, prediction) { return correctCount(state, prediction) > 2 },
  some: function(state, prediction) { return correctCount(state, prediction) > 0 },
  none: function(state, prediction) { return correctCount(state, prediction) === 0 }
}
```



```
// literal listener
var literalListener = function(utt, prediction, possStates) {
  return Infer({model: function(){
    var state = uniformDraw(possStates)
    var meaning = literalMeanings[utt]
    condition(meaning(state, prediction))
    return state
  }})
}
```

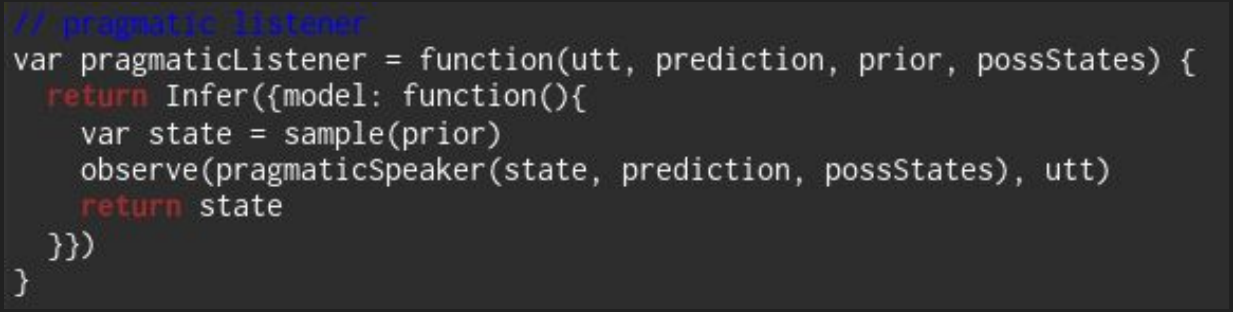
Literal Listener:
What would a non-pragmatic listener believe to be the true state after hearing **utt in** response to **prediction** when the only options left are **possStates**?

Code Structure 3: Pragmatic Speaker & Listener



```
// pragmatic speaker
var pragmaticSpeaker = function(state, prediction, possStates) {
  return Infer({model: function(){
    var utt = utterancePrior()
    factor(literalListener(utt, prediction, possStates).score(state))
    return utt
  }})
}
```

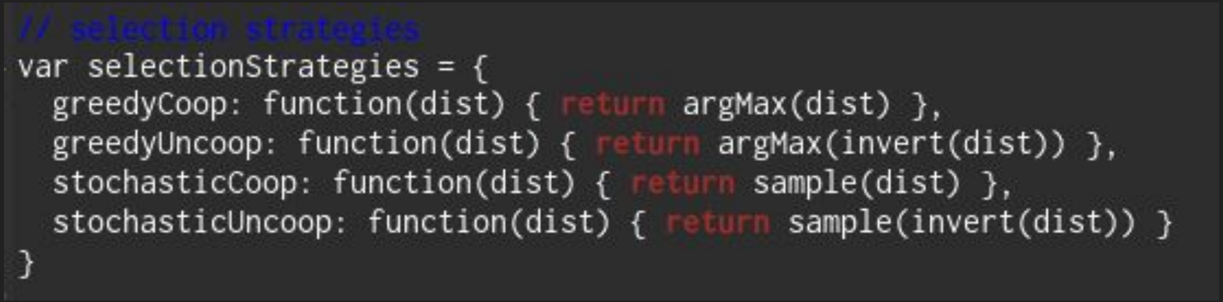
Pragmatic Speaker
Which utterance would make a literal listener believe the most in the given state?



```
// pragmatic listener
var pragmaticListener = function(utt, prediction, prior, possStates) {
  return Infer({model: function(){
    var state = sample(prior)
    observe(pragmaticSpeaker(state, prediction, possStates), utt)
    return state
  }})
}
```

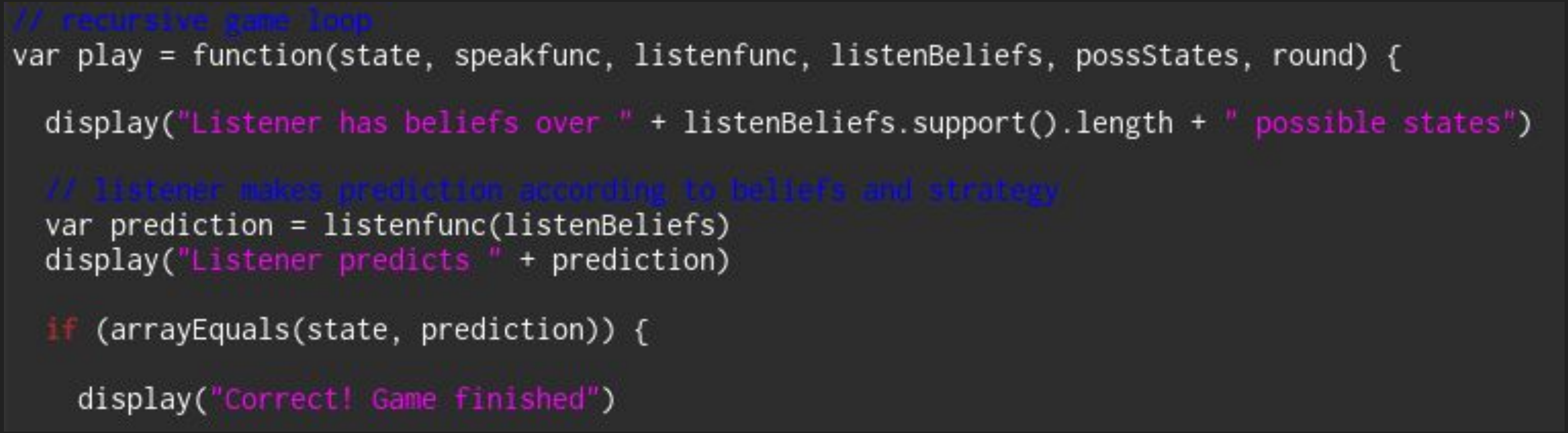
Pragmatic Listener
In which true state would a pragmatic speaker choose the given utterance?

Code Structure 4: Selection Strategies and Beliefs



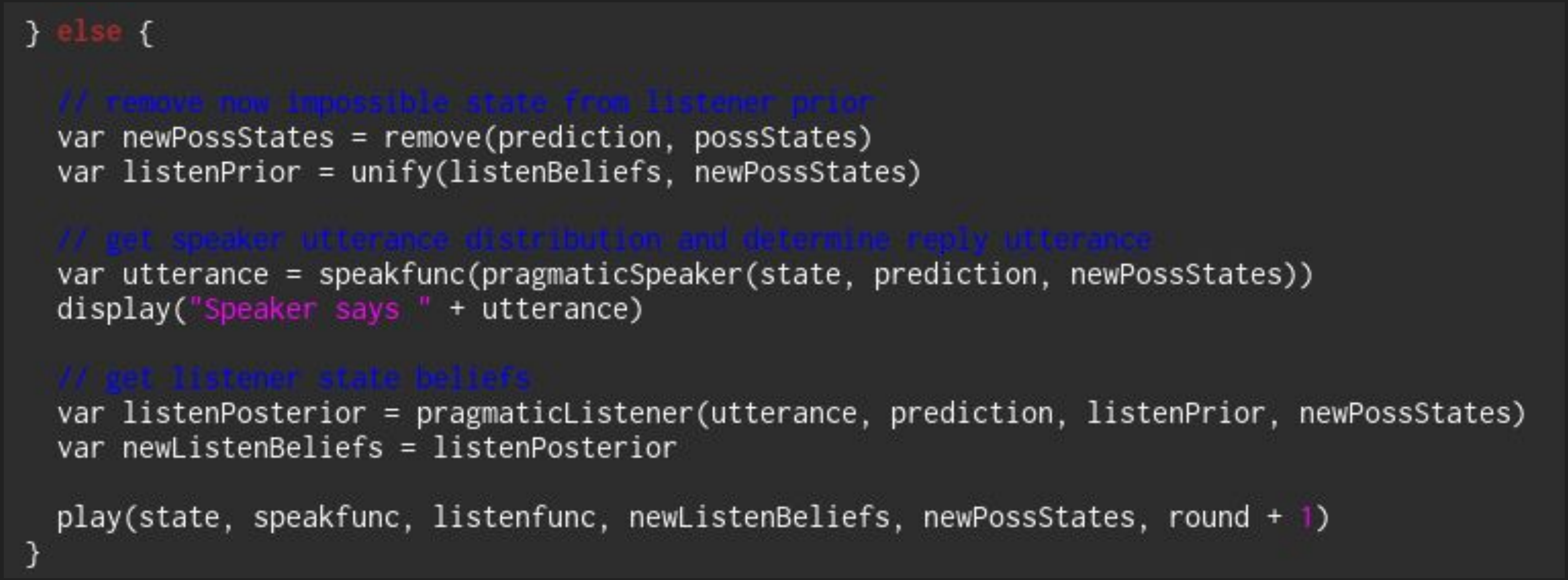
```
// selection strategies
var selectionStrategies = {
  greedyCoop: function(dist) { return argMax(dist) },
  greedyUncoop: function(dist) { return argMax(invert(dist)) },
  stochasticCoop: function(dist) { return sample(dist) },
  stochasticUncoop: function(dist) { return sample(invert(dist)) }
}
```

Selection Strategies
Strategies are used to determine the actions of speaker and listener given their beliefs.
speakfunc and listenfunc are both selection strategies



```
// recursive game loop
var play = function(state, speakfunc, listenfunc, listenBeliefs, possStates, round) {

 display("Listener has beliefs over " + listenBeliefs.support().length + " possible states")

 // listener makes prediction according to beliefs and strategy
 var prediction = listenfunc(listenBeliefs)
 display("Listener predicts " + prediction)

 if (arrayEquals(state, prediction)) {

   display("Correct! Game finished")
```

Code Structure 5: Recursive Main Loop

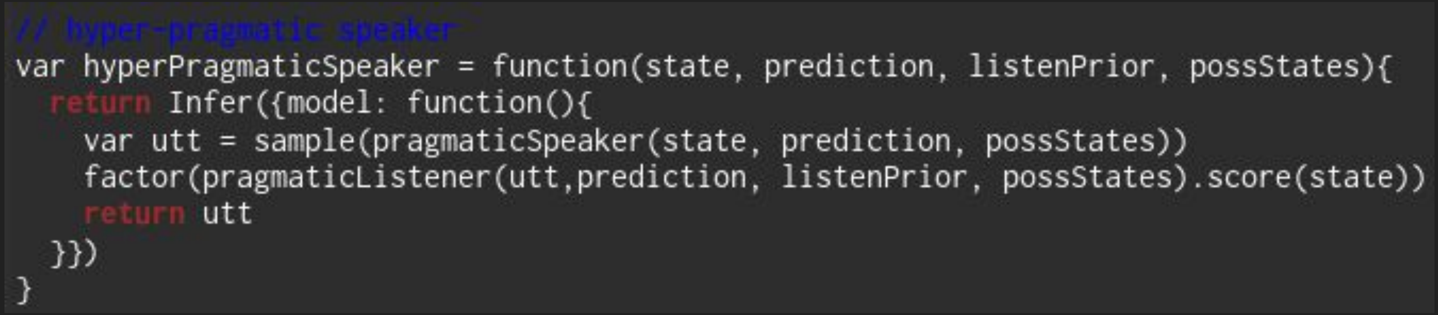


```
} else {

  // remove now impossible state from listener prior
  var newPossStates = remove(prediction, possStates)
  var listenPrior = unify(listenBeliefs, newPossStates)

  // get speaker utterance distribution and determine reply utterance
  var utterance = speakfunc(pragmaticSpeaker(state, prediction, newPossStates))
  display("Speaker says " + utterance)

  // get listener state beliefs
  var listenPosterior = pragmaticListener(utterance, prediction, listenPrior, newPossStates)
  var newListenBeliefs = listenPosterior

  play(state, speakfunc, listenfunc, newListenBeliefs, newPossStates, round + 1)
}
```

Main Loop

If prediction is not correct, the predicted state is not possible and listener beliefs are updated.
After speaker responds to prediction, listener beliefs are updated again based on utterance.

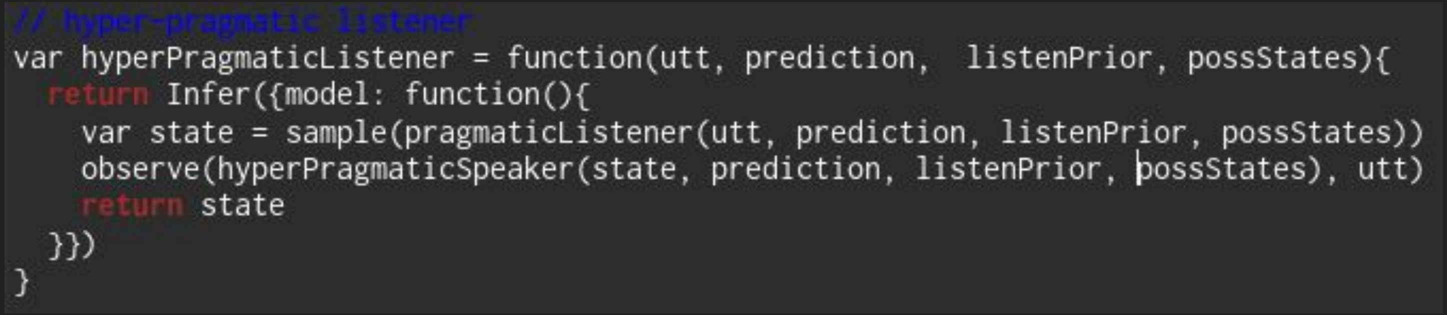
Code Structure 6: Hyperpragmatic Extensions



```
// hyper-pragmatic speaker
var hyperPragmaticSpeaker = function(state, prediction, listenPrior, possStates){
  return Infer({model: function(){
    var utt = sample(pragmaticSpeaker(state, prediction, possStates))
    factor(pragmaticListener(utt,prediction, listenPrior, possStates).score(state))
    return utt
  }})
}
```

Hyperpragmatic Speaker

Takes into account what a stochastic-cooperative pragmatic listener would believe to be true think given an utterance



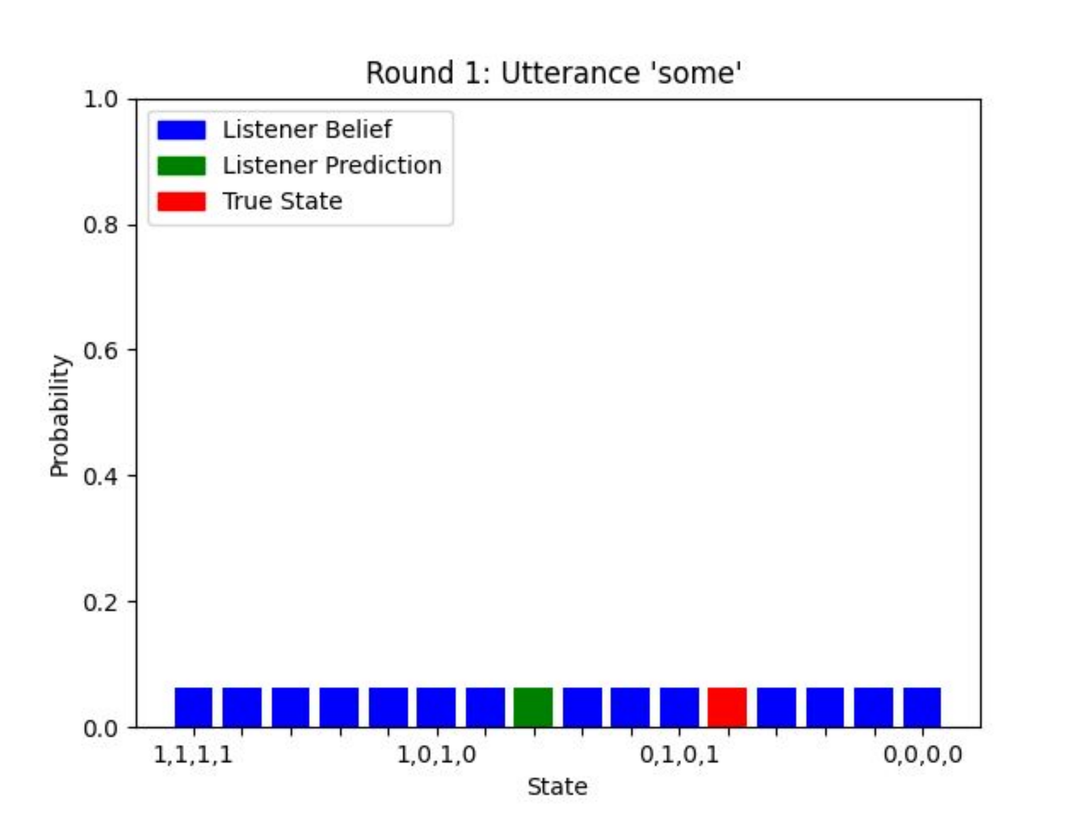
```
// hyper-pragmatic listener
var hyperPragmaticListener = function(utt, prediction,  listenPrior, possStates){
  return Infer({model: function(){
    var state = sample(pragmaticListener(utt, prediction, listenPrior, possStates))
    observe(hyperPragmaticSpeaker(state, prediction, listenPrior, possStates), utt)
    return state
  }})
}
```

Hyperpragmatic Listener

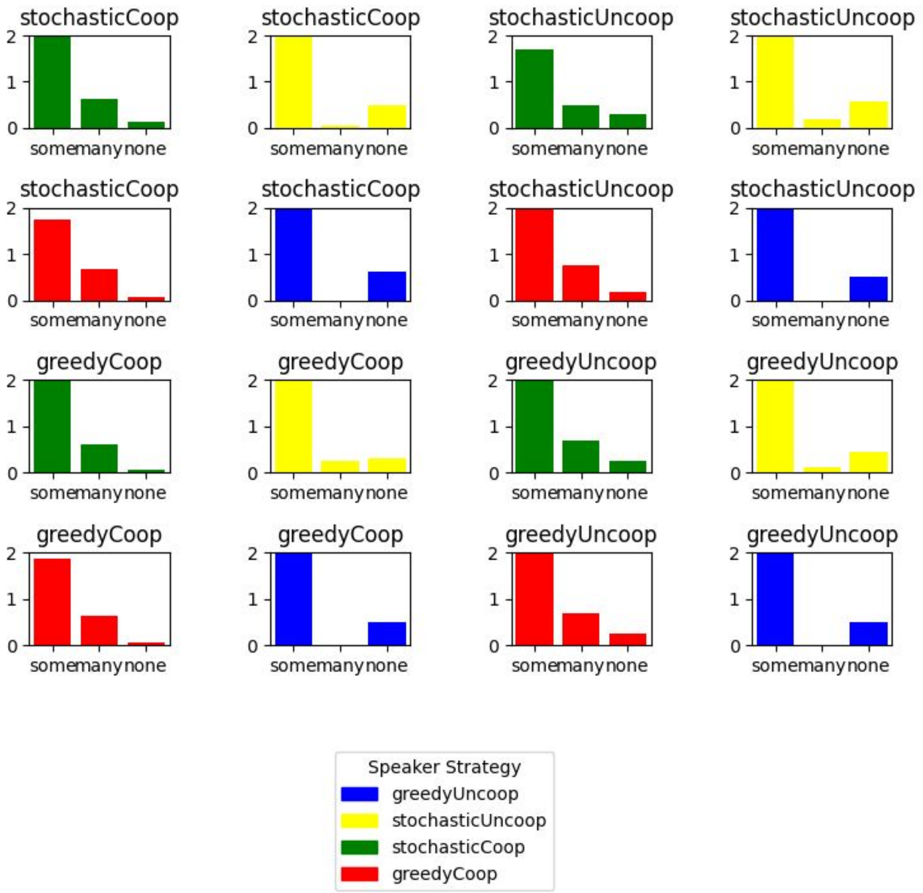
Takes into account what a stochastic-cooperative hyperpragmatic speaker would say given a prediction

Episode III: Meta-Analysis

Exemplary Listener Beliefs Over Time



Pragmatic Mastermind
Mean Utterance Frequencies by Listener Strategy


stochasticCoop

stochasticCoop

stochasticUncoop

stochasticUncoop

stochasticCoop

stochasticCoop

stochasticUncoop

stochasticUncoop

greedyCoop

greedyCoop

greedyUncoop

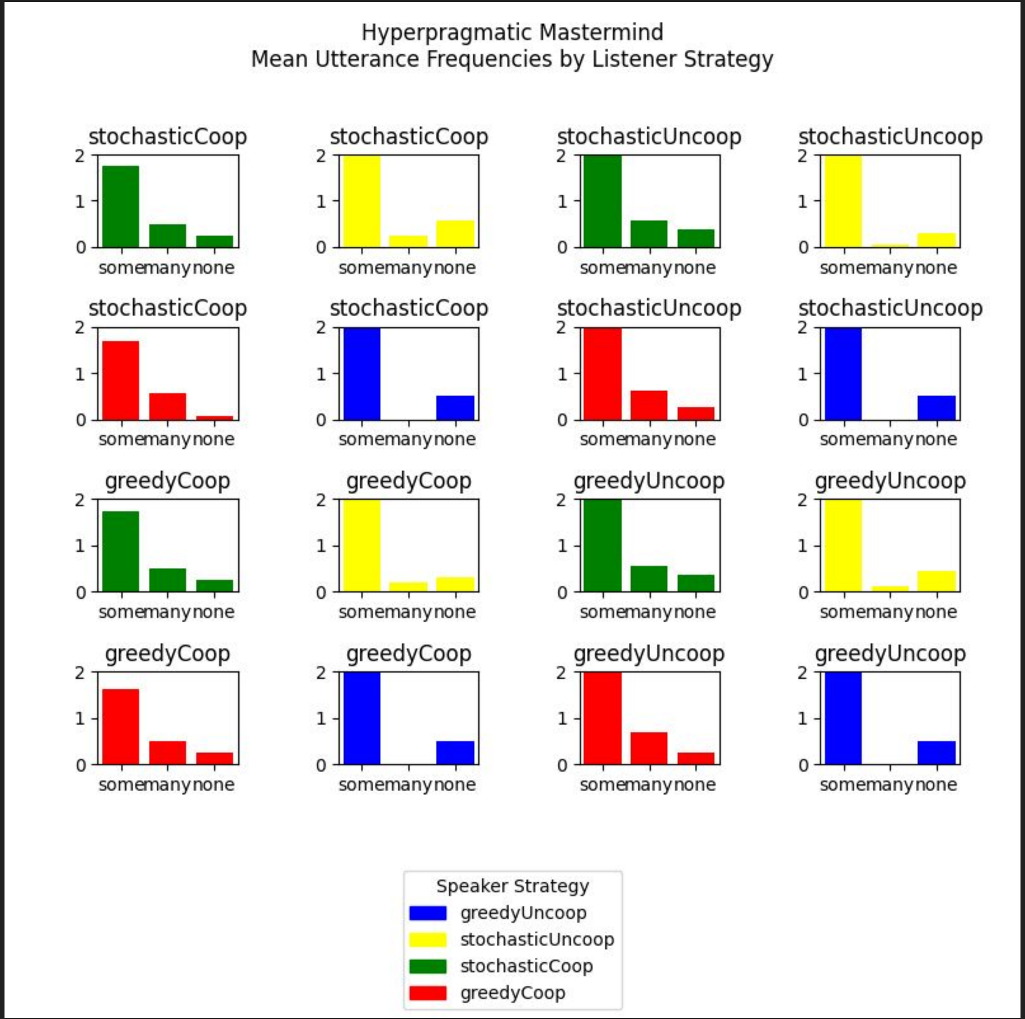
greedyUncoop

greedyCoop

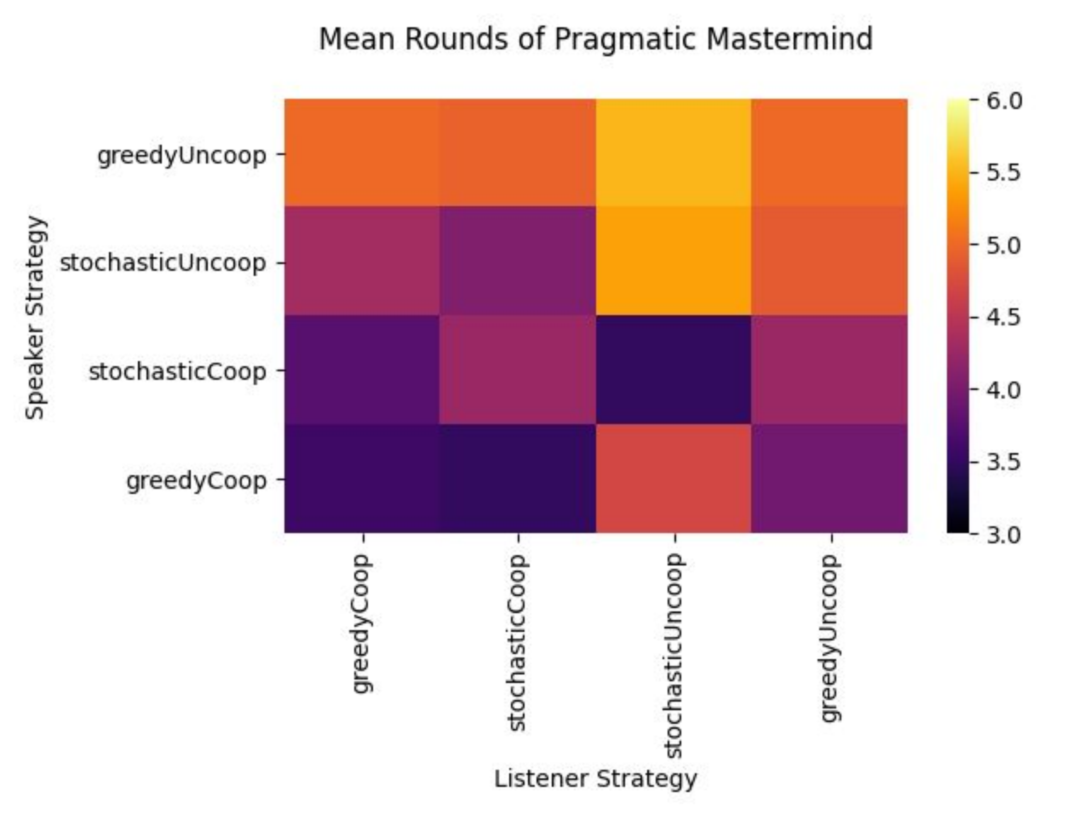
greedyCoop

greedyUncoop

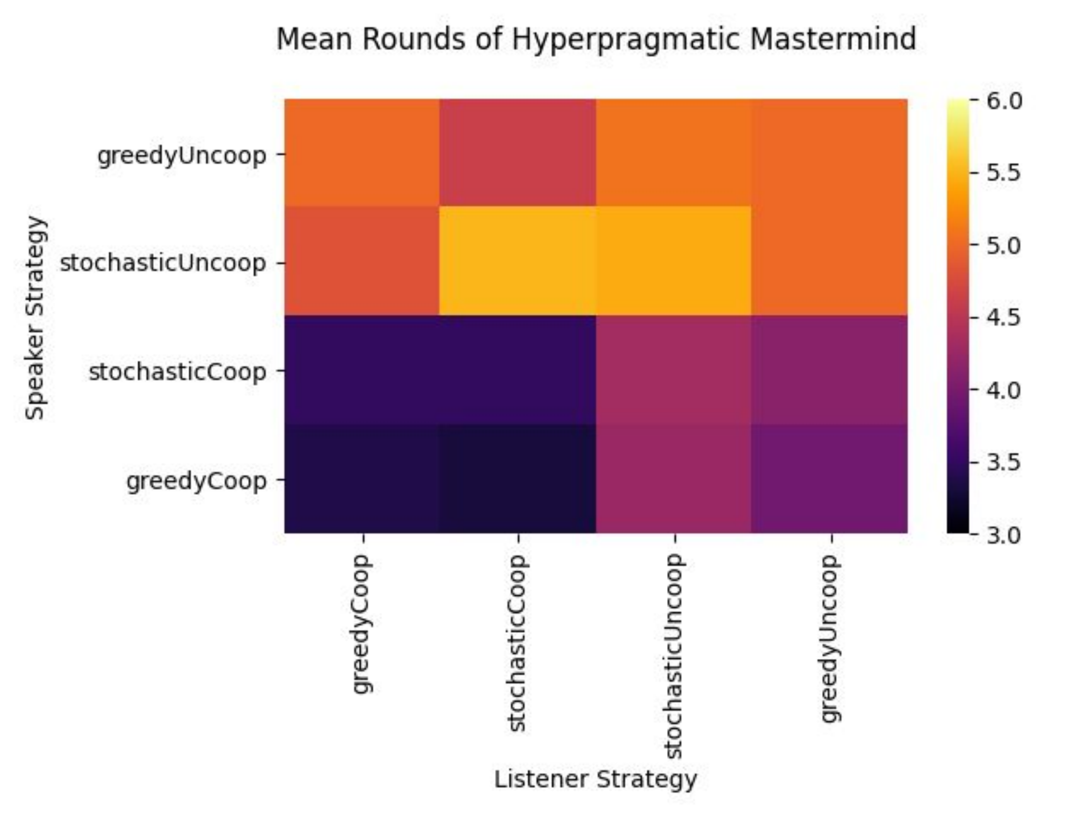
greedyUncoop


Hyperpragmatic Mastermind
Mean Utterance Frequencies by Listener Strategy
stochasticCoop
stochasticCoop
stochasticUncoop
stochasticUncoop
stochasticCoop
stochasticCoop
stochasticUncoop
stochasticUncoop
greedyCoop
greedyCoop
greedyUncoop
greedyUncoop
greedyCoop
greedyCoop
greedyUncoop
greedyUncoop
0
1
2
some
many
none
Speaker Strategy
greedyUncoop
stochasticUncoop
stochasticCoop
greedyCoop

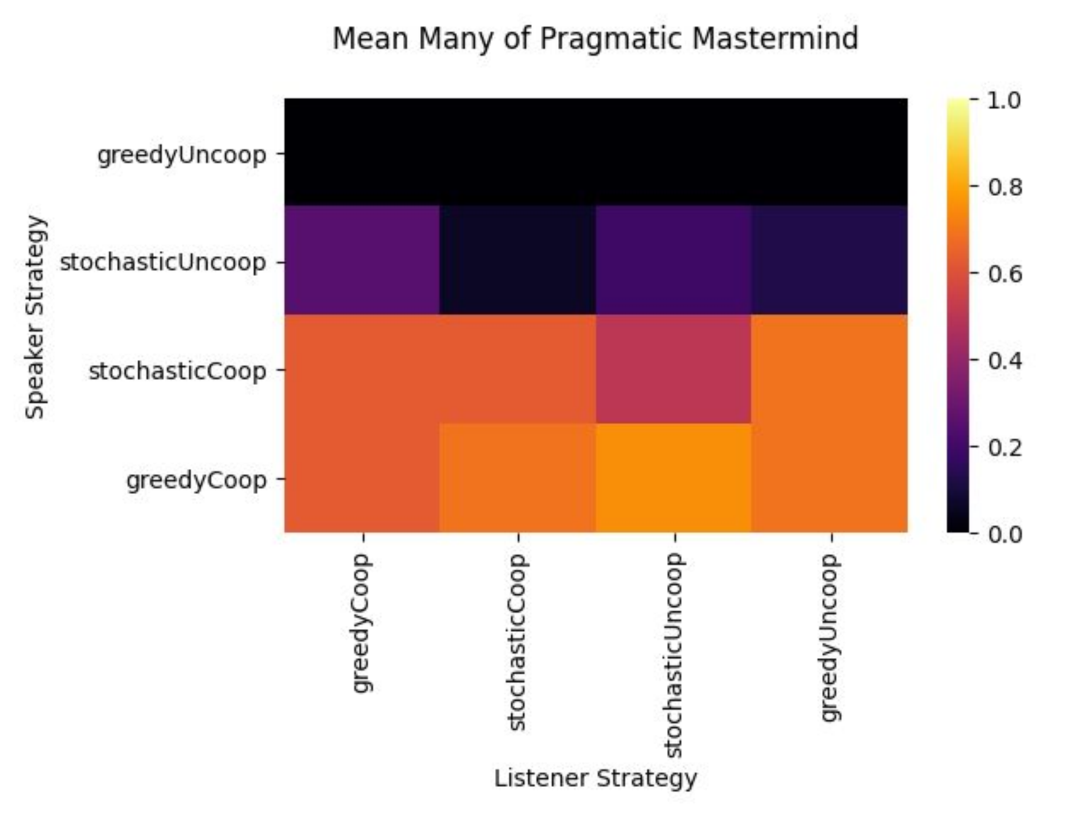
Pragmatic Strategy Comparison



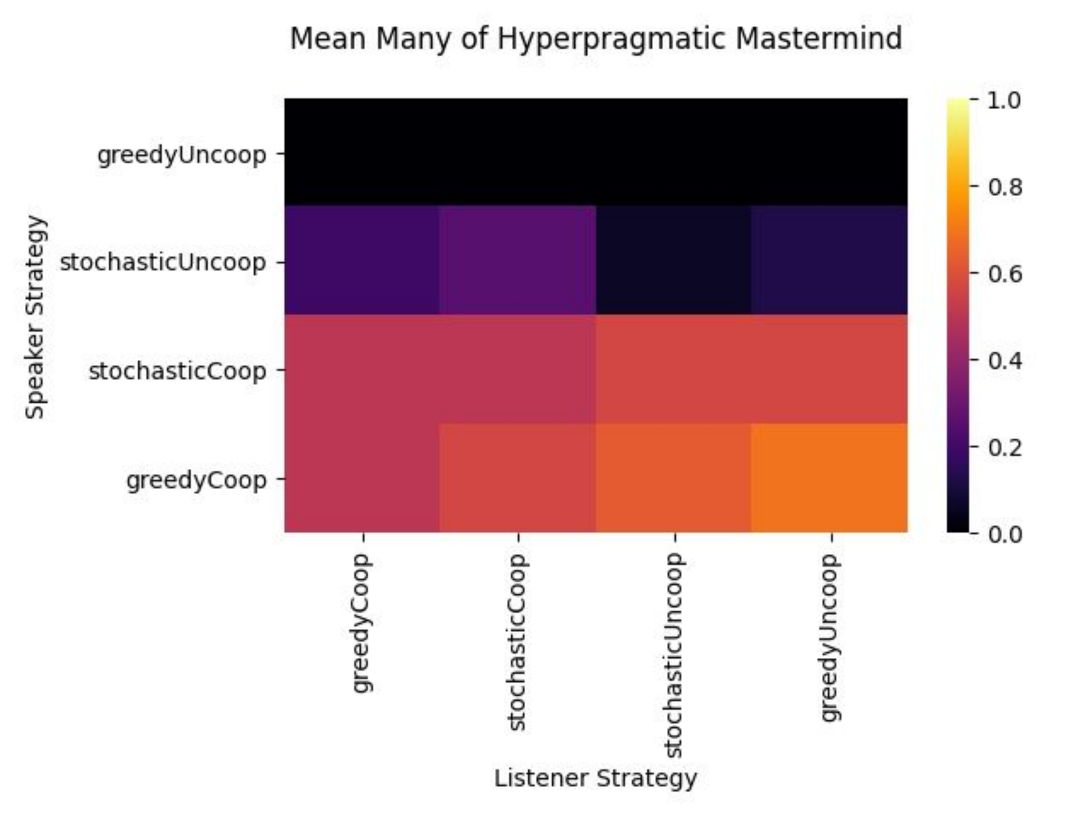
Hyperpragmatic Strategy Comparison



Pragmatic Speaker Frequency “Many”



Hyperpragmatic Speaker Frequency “Many”



Thanks for Watching!

R2D2 and C3PO speak utterance "GOODBYE"