## All Utterances

# All possible utterances (i.e. object features) that can be handled. Here, we assume a 3x3 matrix (three feature types with three expressions each)

allUtterances <- c('cloud', 'circle', 'square', 'solid', 'striped', 'dotted', 'blue', 'red', 'green')

allUtterancesNew1 <- c('cloud', 'circle', 'square', 'solid', 'striped', 'polka-dotted', 'blue', 'red', 'green')

allFeatureTypesNew1 <- c('shape','pattern','color')

allUttMatrix <- matrix(allUtterances, ncol=3, byrow=TRUE)

##

## All Objects

# all object matrix contains 3^3 types of objects.

# the matrix essentially specifies the 3 feature expressions for each object thus, the matrix maps objects to matching utterances

# all Objects implements the strings, allObjectsToUtterancesMappings encodes the index mappings

allObjects <- matrix('',27,3)

allObjectsToUtterancesMappings <- matrix(0,27,3)

for(index in c(1:27)) {

# print(c(1+((index-1)%%3), 1+floor(((index-1)%%9)/3), 1+floor((index-1)/9)))

allObjects[index,1] <- allUttMatrix[1,1+((index-1)%%3)]

allObjects[index,2] <- allUttMatrix[2,1+floor(((index-1)%%9)/3)]

allObjects[index,3] <- allUttMatrix[3,1+floor((index-1)/9)]

allObjectsToUtterancesMappings[index,1] <- 1+((index-1)%%3)

allObjectsToUtterancesMappings[index,2] <- 4+floor(((index-1)%%9)/3)

allObjectsToUtterancesMappings[index,3] <- 7+floor((index-1)/9)

}

## The relevant utterances are determined given currentObjects valid utterances correspond to all features present in the current objects!

determineValidUtterances <- function(currentObjects) {

validUtterances <- c()

for(i in c(1:length(currentObjects))) {

validUtterances <- c(validUtterances, allObjectsToUtterancesMappings[currentObjects[i],])

}

validUtterances <- sort(unique(validUtterances))

return(validUtterances)

}

###

## No preference is encoded with 4, whereas a specific feature expression preference is encoded by the respective index value

# get feature-respective priors returns general feature respective priors for all 3 features @deprecated (not used currently!)

getFeatureRespectivePriors <- function(softAddProb) {

featureRespectivePriors <- list()

for(i in c(1:3)) { ## for all three features generate a preference matrix

m <- matrix(0,4,3)

for(fPref in c(1:3)) {

m[fPref,fPref] <- 1

m[fPref,] <- m[fPref,] + softAddProb

m[fPref,] <- m[fPref,] / sum(m[fPref,])

}

m[4,] <- 1/3

featureRespectivePriors[[i]] <- m

}

return(featureRespectivePriors)

}

##

## Determining the specifc mapping of objects to utterances that applies given currentObjects mapping current objects to utterances

determineObjectToUtterancesMapping <- function(currentObjects) {

mapObjToUtt <- matrix(0, length(currentObjects), 3)

for(i in c(1:length(currentObjects))) {

mapObjToUtt[i,] <- allObjectsToUtterancesMappings[currentObjects[i],]

}

return(mapObjToUtt)

}

##

# Determining the corresponding mappings from all relevant utterances to objects

# parameter notObeyInst determines if the instruction does not need to be obeyed (0=full obedience: -> infty =full instruction ignorance)

determineUtteranceToObjectProbabilities <- function(consideredUtterances, currentObjects,

mapObjToUtt, notObeyInst) {

mapUttToObj <- list()

mapUttToObjProbs <- matrix(notObeyInst, length(consideredUtterances), length(currentObjects))

for(utt in rep(1:length(consideredUtterances)) ) {

# determine array of all objects that match the utterance

mapUttToObj[[utt]] = ((which(mapObjToUtt[,] == consideredUtterances[utt])-1)%%nrow(mapObjToUtt))+1

for(i in rep(1:length(mapUttToObj[[utt]]))) {

mapUttToObjProbs[utt,mapUttToObj[[utt]][i]] <- mapUttToObjProbs[utt,mapUttToObj[[utt]][i]] + 1;

}

mapUttToObjProbs[utt,] <- mapUttToObjProbs[utt,] / sum(mapUttToObjProbs[utt,])# length(mapUttToObj[[utt]])

}

return(mapUttToObjProbs)

}

## Priors on object preferences - automatically derived from considered utterances (i.e. derived from all relevant features)

# type == 0: hard priors; type > 0: soft prior with specified softness

# returns a list of preference priors for all considered features, i.e. utterances, as well as for "no preference" whatsoever, i.e., uniform prior over all three objects

getObjectPreferencePriors <- function(consideredUtterances, currentObjects, type, mapUttToObjProbs) {

objectPreferenceHardPriors <- list()

for(utt in rep(1:length(consideredUtterances)) ) {

objectPreferenceHardPriors[[utt]] <- mapUttToObjProbs[utt,]

}

objectPreferenceHardPriors[[length(consideredUtterances)+1]] =

rep(1/length(currentObjects), length(currentObjects) )

# soft preferences with uniform choice fusion.

softAddProb <- type

objectPreferenceSoftPriors <- list()

for(utt in rep(1:(length(consideredUtterances)+1)) ) {

objectPreferenceSoftPriors[[utt]] <- objectPreferenceHardPriors[[utt]] + softAddProb

objectPreferenceSoftPriors[[utt]] <- objectPreferenceSoftPriors[[utt]] / sum(objectPreferenceSoftPriors[[utt]])

}

return(objectPreferenceSoftPriors)

}

> print(ls())

[1] "allFeatureTypesNew1"

[2] "allObjects"

[3] "allObjectsToUtterancesMappings"

[4] "allUtterances"

[5] "allUtterancesNew1"

[6] "allUttMatrix"

[7] "currentObjects"

[8] "determineObjectToUtterancesMapping"

[9] "determineUtteranceToObjectProbabilities"

[10] "determineValidUtterances"

[11] "getFeatureRespectivePriors"

[12] "getObjectPreferencePriors"

[13] "getSpeakerUtteranceUniformPrior"

[14] "index"

[15] "KLdivergence"

[16] "mapObjToUtt"

[17] "mapUttToObjProbs"

[18] "notObeyInst"

[19] "objectPreferenceSoftPriors"

[20] "relevantUtterances"

[21] "simpleBestInfGainUtterance"

[22] "simpleListener"

[23] "simplePragmaticSpeaker"

[24] "softPrefValue"

> allFeatureTypesNew1

[1] "shape" "pattern" "color"

> allObjects

[,1] [,2] [,3]

[1,] "cloud" "solid" "blue"

[2,] "circle" "solid" "blue"

[3,] "square" "solid" "blue"

[4,] "cloud" "striped" "blue"

[5,] "circle" "striped" "blue"

[6,] "square" "striped" "blue"

[7,] "cloud" "dotted" "blue"

[8,] "circle" "dotted" "blue"

[9,] "square" "dotted" "blue"

[10,] "cloud" "solid" "red"

[11,] "circle" "solid" "red"

[12,] "square" "solid" "red"

[13,] "cloud" "striped" "red"

[14,] "circle" "striped" "red"

[15,] "square" "striped" "red"

[16,] "cloud" "dotted" "red"

[17,] "circle" "dotted" "red"

[18,] "square" "dotted" "red"

[19,] "cloud" "solid" "green"

[20,] "circle" "solid" "green"

[21,] "square" "solid" "green"

[22,] "cloud" "striped" "green"

[23,] "circle" "striped" "green"

[24,] "square" "striped" "green"

[25,] "cloud" "dotted" "green"

[26,] "circle" "dotted" "green"

[27,] "square" "dotted" "green"

> allObjectsToUtterancesMappings

[,1] [,2] [,3]

[1,] 1 4 7

[2,] 2 4 7

[3,] 3 4 7

[4,] 1 5 7

[5,] 2 5 7

[6,] 3 5 7

[7,] 1 6 7

[8,] 2 6 7

[9,] 3 6 7

[10,] 1 4 8

[11,] 2 4 8

[12,] 3 4 8

[13,] 1 5 8

[14,] 2 5 8

[15,] 3 5 8

[16,] 1 6 8

[17,] 2 6 8

[18,] 3 6 8

[19,] 1 4 9

[20,] 2 4 9

[21,] 3 4 9

[22,] 1 5 9

[23,] 2 5 9

[24,] 3 5 9

[25,] 1 6 9

[26,] 2 6 9

[27,] 3 6 9

> allUtterances

[1] "cloud" "circle" "square" "solid" "striped" "dotted" "blue"

[8] "red" "green"

> allUtterancesNew1

[1] "cloud" "circle" "square" "solid" "striped"

[6] "polka-dotted" "blue" "red" "green"

> allUttMatrix

[,1] [,2] [,3]

[1,] "cloud" "circle" "square"

[2,] "solid" "striped" "dotted"

[3,] "blue" "red" "green"

> currentObjects

[1] 1 2 6

> index

[1] 27

> mapObjToUtt

[,1] [,2] [,3]

[1,] 1 4 7

[2,] 2 4 7

[3,] 3 5 7

> mapUttToObjProbs

[,1] [,2] [,3]

[1,] 1.0000000 0.0000000 0.0000000

[2,] 0.0000000 1.0000000 0.0000000

[3,] 0.0000000 0.0000000 1.0000000

[4,] 0.5000000 0.5000000 0.0000000

[5,] 0.0000000 0.0000000 1.0000000

[6,] 0.3333333 0.3333333 0.3333333

> notObeyInst

[1] 0

> objectPreferenceSoftPriors

[[1]]

[1] 0.980582524 0.009708738 0.009708738

[[2]]

[1] 0.009708738 0.980582524 0.009708738

[[3]]

[1] 0.009708738 0.009708738 0.980582524

[[4]]

[1] 0.495145631 0.495145631 0.009708738

[[5]]

[1] 0.009708738 0.009708738 0.980582524

[[6]]

[1] 0.3333333 0.3333333 0.3333333

[[7]]

[1] 0.3333333 0.3333333 0.3333333

> relevantUtterances

[1] 1 2 3 4 5 7

> softPrefValue

[1] 0.01