Visualising the entropy state space of signalling games

Kevin Stadler

March 12, 2014

The point of this document is to visualise the distribution of visitable points in $H(M|S) \times H(S|M)$ space. Entropy in signalling games can stem from one of two sources: the first source is misalignment between agents, and the second is the conditional entropy of the average production system (which, assuming all agents are perfectly aligned, is identical to the individual agents' production system).

These two sources of entropy as well as the fact that production matrices can vary arbitrarily means that there is an enormous number of visitable states to be plotted. By adding a number of (strong) constraints to the individual as well as system-average production matrices we consider, we reduce the number of plotted states (which might all end up in an arbitrarily dense and boring part of the entropy space anyway) while still trying to grasp the overall topology of the entropy state space.

Our strongest assumption is that all agents are purely categorical, deterministic 'winner-takes-all' (WTA) producers, i.e. their distribution of productions for any given meaning is a Dirac distribution. This helps us to massively reduce the number of production matrices we have to consider, for an $m \times n$ signalling game (with m meanings and n signals) to an *absolute* (but normally much lower) maximum of n^m production matrices.

In order to explore where non-aligned populations of WTA production agents (or, equivalently, aligned populations of agents with probabilistic production functions) behave, we also consider increasingly non-Dirac system-average production matrices. We do so by enumerating all signal production distributions composed of increasingly smaller fractions. A fraction of 2 for example allows for a 50:50 split between two signals produced for a particular meaning, while a fraction of 3 allows for 66:33 splits and, if there are three or more possible signals, also for 33:33:33 splits. In general any smaller fraction (in terms of the denominator, i.e. 3 is a smaller fraction than 2) can generate more different distributions than the higher ones, although not necessarily a superset of those. While any fraction will still be able to generate the 100:0 Dirac distribution, a fraction of 3 cannot generate a 50:50 split of the 2 fraction. When it comes to the plots below, the possible points of lower fractions are plotted first, so that the smaller number of points produced by fractions closer to 1 are printed on top of the more densely-spaced lower fraction ones. This will become more clear when you get to the plots below.

As mentioned above, the states in the entropy space obtained by plotting the conditional entropy of such fractional production matrices can have two interpretations: they are either nonaligned WTA agents (a 50:50 split corresponds to an even number of agents who are split in their deterministic production between two different signals), or otherwise simply a step-wise iteration towards enumerating the (uncountably infinite) unconstrained production matrices. For the former case it is informative to think of the 'fractions' as the number of agents in the population, i.e. assuming that agents use WTA production (or any other kind of production that constrains the possible per-meaning signal production distributions), the plots below show how new, formerly unvisitable states in the entropy space crop up as the population size of interacting agents is increased.

The functions below help enumerate all such matrices in a relatively optimal way.

```
# returns a matrix of all rows of length n where the elements sum to s
# enumeraterows(n, 1) basically enumerates all Dirac distributions over support [1,n]
enumeraterows <- function(n, s) {
  if (s == 0) {
    # return n zeros
    return(matrix(0, nrow=1, ncol=n))
  } else if (n == 1) {
    return(matrix(s))
```

```
  } else {
    # all possible first values (s:0)
    do.call(rbind, lapply(s:0,
      function(first) cbind(first, enumeraterows(n-1, s-first))))
  }
}
# as a side-effect this function can also be used to generate all optimal
# production matrices (i.e. permutation matrices)
# enumeraterows(n, 1)

# having enumerated all possible production distributions (rows), we now need a
# way to generate from that all possible production matrices, i.e. combinations
# of the rows we just generated, in a relatively efficient and non-redundant
# manner (i.e. avoiding permutations). go combinatorics!

# returns a matrix with k columns which has as its rows all unique,
# order-insensitive k-combinations with repetitions of k integers from [1,n],
# starting with [1, 1, 1,..., 1] (i.e. rep(1, k)) and ending on rep(n, k).
# there are (n+k-1 over k) of those, so the result is an (n+k-1 over k)xk matrix.
combnr <- function(k, n, minelement=1) {
  if (k == 1) {
    return(matrix(minelement:n, ncol=1))
  } else {
    # recurse
    do.call(rbind, lapply(max(1, minelement):n, function(first)cbind(first, combnr(k-1, n, first))))
  }
}

# calculate the multiset coefficient which gives the number of possible
# k-combinations with repetition)
ncombnr <- function(k, n)
  choose(n+k-1, k)

# if the enumeration algorithm works these should be identical
c(dim(combnr(5,7))[1], ncombnr(5,7))

## [1] 462 462
```

Now that we can enumerate all possible production matrices, here come the functions to compute their conditional entropies:

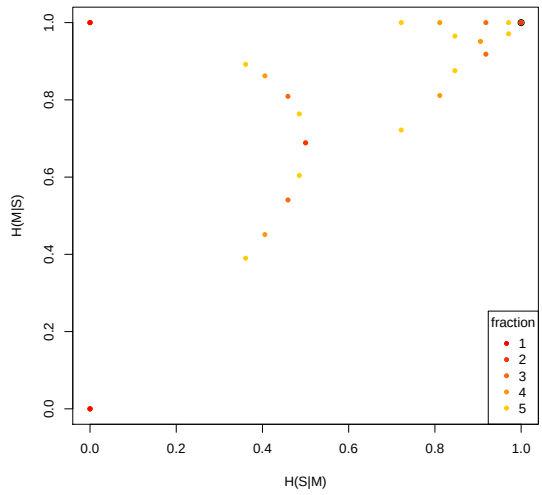
```
# in R, log2(0)=-Inf which carries over to 0*log2(0)=NaN which we don't like, no no no!
safelog <- function(x) {
  res <- log2(x)
  # it's gonna be multiplied with 0 anyway, so return anything but Inf and NaN
  res[res==-Inf] <- 0
  return(-res)
}

# optimality = lack of homonymy = conditional entropy H(M|S)
# H(M|S) = Es p(s) * H(M|S=s) = Es p(s) * Em [ p(m|s) * 1/log(p(m|s)) ]
homonymy <- function(prodmatrix) {
  # drop columns(signals) which never occur since they're a) not relevant
    # and b) lead to NaNs when trying to normalise a bit further down
  strippedmatrix <- prodmatrix[, colSums(prodmatrix)!=0, drop=FALSE]
  # to calculate all the H(m|S=s) we need the per-signal counts to normalise
```

```
  signalprobs <- colSums(strippedmatrix)
  # from this we can calculate the p(m|s) for every cell - sweep(prodmatrix, 2, x, `/`)
  signalnormalised <- strippedmatrix / signalprobs[col(strippedmatrix)]
  # then we can calculate the weighted sum p(m|s)*log(1/p(m|s)) for every meaning
  permeaningconditionalentropies <- apply(signalnormalised, 2,
    function(row) weighted.mean(safelog(row), row))
  # and finally the outer weighted sum over all signals
  weighted.mean(permeaningconditionalentropies, signalprobs)
  # per-signal weight normalisation (divide by number of meanings dim(prodmatrix)[2])
  # is taken care of internally by weighted.mean()
}
#homonymy(matrix(c(0,0,1,1),nrow=2))

# H(S|M) = Em p(m) * H(S|M=m) = Em p(m) * Es [ p(s|m) * 1/log(p(s|m)) ]
# FOR THIS FUNCTION THE PRODMATRIX ACTUALLY HAS TO BE NORMALISED!!
# (i.e. the values in every row have to sum to 1) - otherwise the log will mess up.
synonymy <- function(prodmatrix) {
  # for uniform distribution of meanings all p(m) are identical, no need to compute
  mean(apply(prodmatrix, 1, function(row)weighted.mean(safelog(row), row)))
}

# I(x;y) = ExEy p(x,y) * log(p(x,y) / p(x)*p(y))
# other ways to compute:
# I(x;y) = H(X) - H(X|Y) = H(M) - H(M|S) = log2(|M|) - homonymy(prodmatrix)
mutualinformation <- function(associationmatrix) {
  px <- colSums(associationmatrix)
  # we know that all these will be identical for production matrices..:
  py <- rowSums(associationmatrix)
  # create the matrix of all p(x)*p(y)
  # then use it to point-divide p(x,y) = p(x|y)*p(y) = p(y|x)*p(x)
  # all p(m) are equal and we can simply take the p(s|m) straight from the matrix!
  logs <- safelog(px%*%t(py)/associationmatrix)
  # and return the weighted sum of its log!
  # (the weighted mean is applied across both axes here, whoaah!)
  weighted.mean(logs, associationmatrix)
}
```

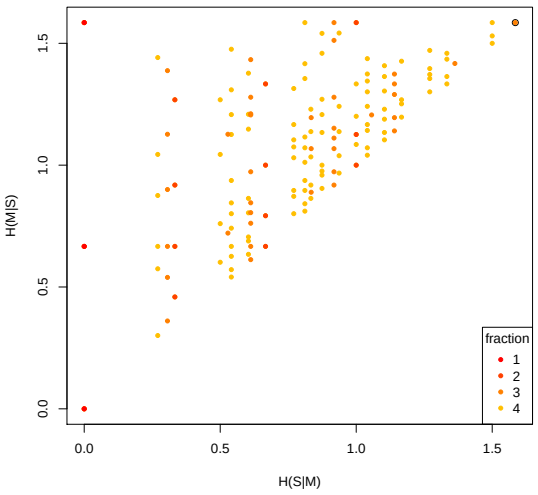
```
# this is the working beast for the beautiful plots!
spaceplot <- function(m, n, fractions=5:1, pch=20, cols=heat.colors(2+length(fractions))) {
  plot(log2(n), log2(m), xlim=c(0,log2(n)), ylim=c(0,log2(m)), xlab='H(S|M)', ylab='H(M|S)')
  for (s in fractions) {
    rows <- enumeraterows(n, s)/s
    # try using system.time() here, just for a laugh.
    # unique() on matrices finds unique rows by default, need to adjust MARGIN
    x <- unique(MARGIN=2, x=apply(combnr(m, dim(rows)[1]), 1,
      function(matrixcomposition)
        c(synonymy(rows[matrixcomposition,]), homonymy(rows[matrixcomposition,]))))
    cat(m, 'x', n, ', fraction=', s, ', ', dim(rows)[1], ' possible rows: ', ncombnr(m, dim(rows)[1]
    points(t(x), pch=pch, col=cols[s])
  }
  legend('bottomright', title='fraction', legend=rev(fractions), pch=pch, col=cols)
}
```

Let's have some plots!

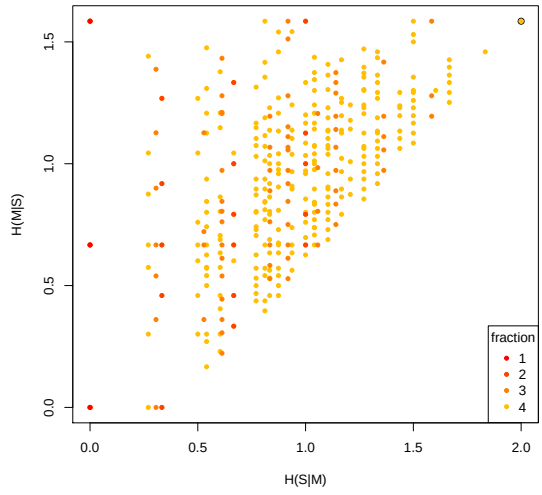
```
spaceplot(2,2,5:1)

## 2x2, fraction=5, 6 possible rows: 21 possible matrices map onto 12 distinct points
## 2x2, fraction=4, 5 possible rows: 15 possible matrices map onto 9 distinct points
## 2x2, fraction=3, 4 possible rows: 10 possible matrices map onto 6 distinct points
## 2x2, fraction=2, 3 possible rows: 6 possible matrices map onto 4 distinct points
## 2x2, fraction=1, 2 possible rows: 3 possible matrices map onto 2 distinct points

spaceplot(3,3,4:1)

## 3x3, fraction=4, 15 possible rows: 680 possible matrices map onto 116 distinct points
## 3x3, fraction=3, 10 possible rows: 220 possible matrices map onto 40 distinct points
## 3x3, fraction=2, 6 possible rows: 56 possible matrices map onto 14 distinct points
## 3x3, fraction=1, 3 possible rows: 10 possible matrices map onto 3 distinct points

spaceplot(3,4,4:1)

## 3x4, fraction=4, 35 possible rows: 7770 possible matrices map onto 281 distinct points
## 3x4, fraction=3, 20 possible rows: 1540 possible matrices map onto 73 distinct points
## 3x4, fraction=2, 10 possible rows: 220 possible matrices map onto 19 distinct points
## 3x4, fraction=1, 4 possible rows: 20 possible matrices map onto 3 distinct points

spaceplot(4,3,4:1)

## 4x3, fraction=4, 15 possible rows: 3060 possible matrices map onto 373 distinct points
## 4x3, fraction=3, 10 possible rows: 715 possible matrices map onto 99 distinct points
## 4x3, fraction=2, 6 possible rows: 126 possible matrices map onto 26 distinct points
## 4x3, fraction=1, 3 possible rows: 15 possible matrices map onto 4 distinct points
```



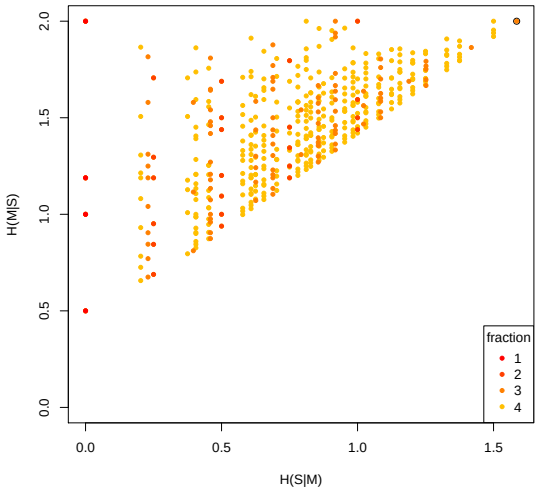
(a) A 2x2 system with only a single agent using WTA production (dark red dots) can only be in one of two possible states: a working signalling system (bottom left corner) or a broken one (top left corner). With two agents, a new possible system (where the two WTA-producing agents have contradictory associations) is added, which corresponds to the orange point in the top right corner. The black circle denotes the least optimal possible system for the signalling game.



(b) In a 3x3 system, even a single agent can exhibit a pooling equilibrium, as can be seen in the middle of the left axis.



(c) A 3x4 system, i.e. one more signal than meanings is available for use: perfect communication is possible despite there being synonymy. Since any individual agent is expected to use WTA production, only populations with 2 or more agents can exhibit synonymy (i.e. according to our assumptions synonymy can only occur across and not within individuals).



(d) A 4x3 system, i.e. at least one signal will have to be used to convey two meanings.

Figure 1: Comparing the possible positions in entropy state space for 2x2 and 3x3 signalling games.

More plots of larger signalling/meaning spaces

```
# this'll take a while
spaceplot(4,4,5:1)
```

```
## 4x4, fraction=5, 56 possible rows: 455126 possible matrices map onto 4356 distinct points
## 4x4, fraction=4, 35 possible rows: 73815 possible matrices map onto 1166 distinct points
## 4x4, fraction=3, 20 possible rows: 8855 possible matrices map onto 235 distinct points
## 4x4, fraction=2, 10 possible rows: 715 possible matrices map onto 43 distinct points
## 4x4, fraction=1, 4 possible rows: 35 possible matrices map onto 5 distinct points
```

```
text(0, 2, '4', pos=4)
text(0, log2(3)*3/4, '3,1', pos=4)
text(0, 1, '2,2', pos=4)
text(0, 0.5, '2,1,1', pos=4)
text(0, 0, '1,1,1,1', pos=4)
```

```
spaceplot(4,4,2:1)
```

```
## 4x4, fraction=2, 10 possible rows: 715 possible matrices map onto 43 distinct points
## 4x4, fraction=1, 4 possible rows: 35 possible matrices map onto 5 distinct points
```

```
spaceplot(5,5,2:1)
```

```
## 5x5, fraction=2, 15 possible rows: 11628 possible matrices map onto 110 distinct points
## 5x5, fraction=1, 5 possible rows: 126 possible matrices map onto 7 distinct points
```

```
spaceplot(6,6,2:1)
```

```
## 6x6, fraction=2, 21 possible rows: 230230 possible matrices map onto 256 distinct points
## 6x6, fraction=1, 6 possible rows: 462 possible matrices map onto 11 distinct points
```

```
# better not
# spaceplot(7,7,2:1)
```

```
spaceplot(3,4,2:1)
```

```
## 3x4, fraction=2, 10 possible rows: 220 possible matrices map onto 19 distinct points
## 3x4, fraction=1, 4 possible rows: 20 possible matrices map onto 3 distinct points
```

```
spaceplot(4,5,2:1)
```

```
## 4x5, fraction=2, 15 possible rows: 3060 possible matrices map onto 50 distinct points
## 4x5, fraction=1, 5 possible rows: 70 possible matrices map onto 5 distinct points
```

```
spaceplot(5,6,2:1)
```

```
## 5x6, fraction=2, 21 possible rows: 53130 possible matrices map onto 125 distinct points
## 5x6, fraction=1, 6 possible rows: 252 possible matrices map onto 7 distinct points
```

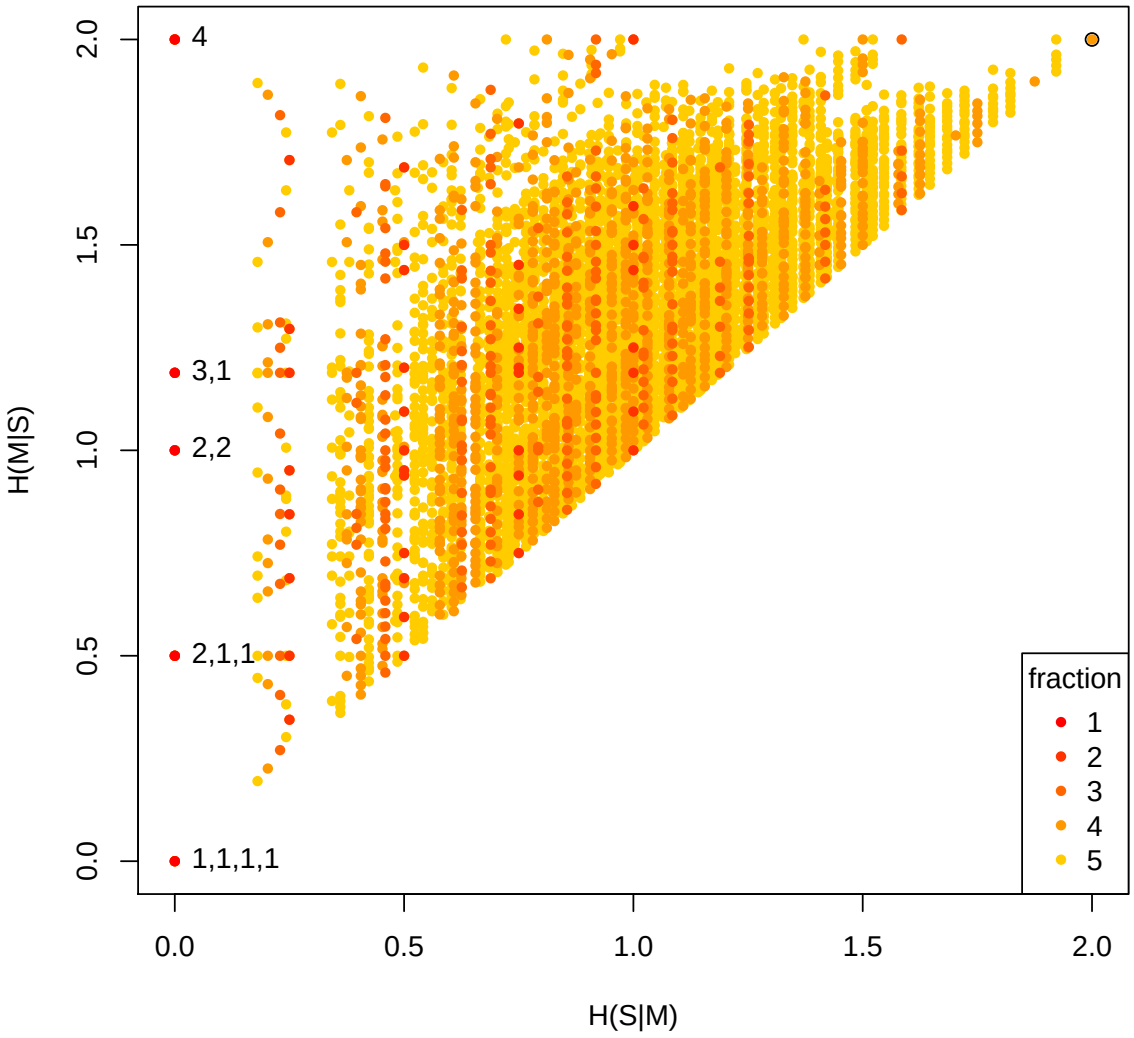
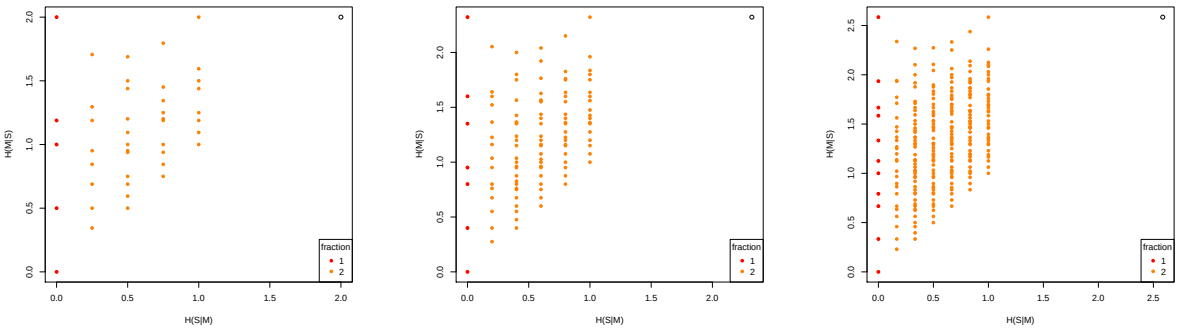
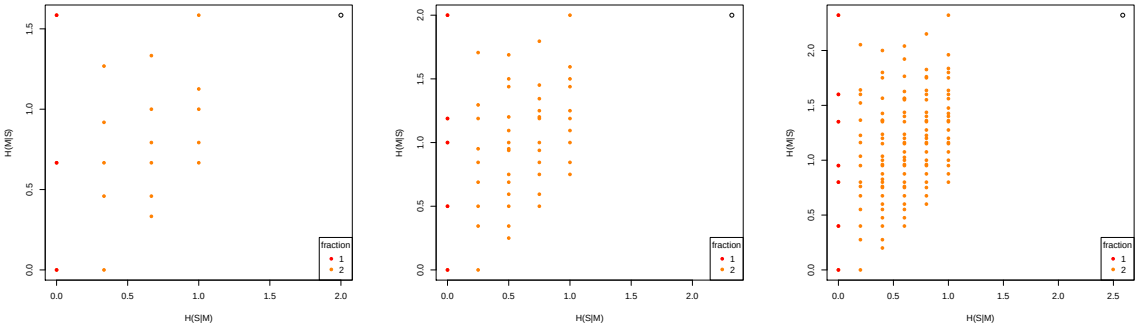


Figure 2: Entropy state space plot for a 4×4 signalling game. The numbers on the left indicate the different pooling equilibria: one signal used for all four meanings at the very top, a pool of three meanings and one unambiguous signal below, two ambiguous signals with two possible meanings each in the middle, followed by having only one pair of homonyms (i.e. one ambiguous signal) and finally a completely unambiguous communication system in the bottom left corner. While the proximity of points does not directly indicate that states are indeed neighbouring states (i.e. that they can be reached by changing one agent's signalling system), some of the patterns can in fact be interpreted as trajectories between states. See in particular the 'arches' with which the pooling equilibria along the left axis are connected with each other. The pooling equilibrium 3,1 can be changed to either a 2,2 or a 2,1,1 equilibrium, and both these trajectories (which are the intermediate states in which individual agents production systems are changed, leading to temporary misalignment) are visible in the outline of arches drawn by the group of points to the right. Note also how it takes at least four (contradicting) WTA agents to be able to occupy the most inefficient spot for a 4×4 signalling game.



(a) 4x4 (b) 5x5 (c) 6x6



(d) 3x4 (e) 4x5 (f) 5x6

Figure 3: Visualising the increasing density of the state space as more agents (or, equivalently, less categorical agents) are added.