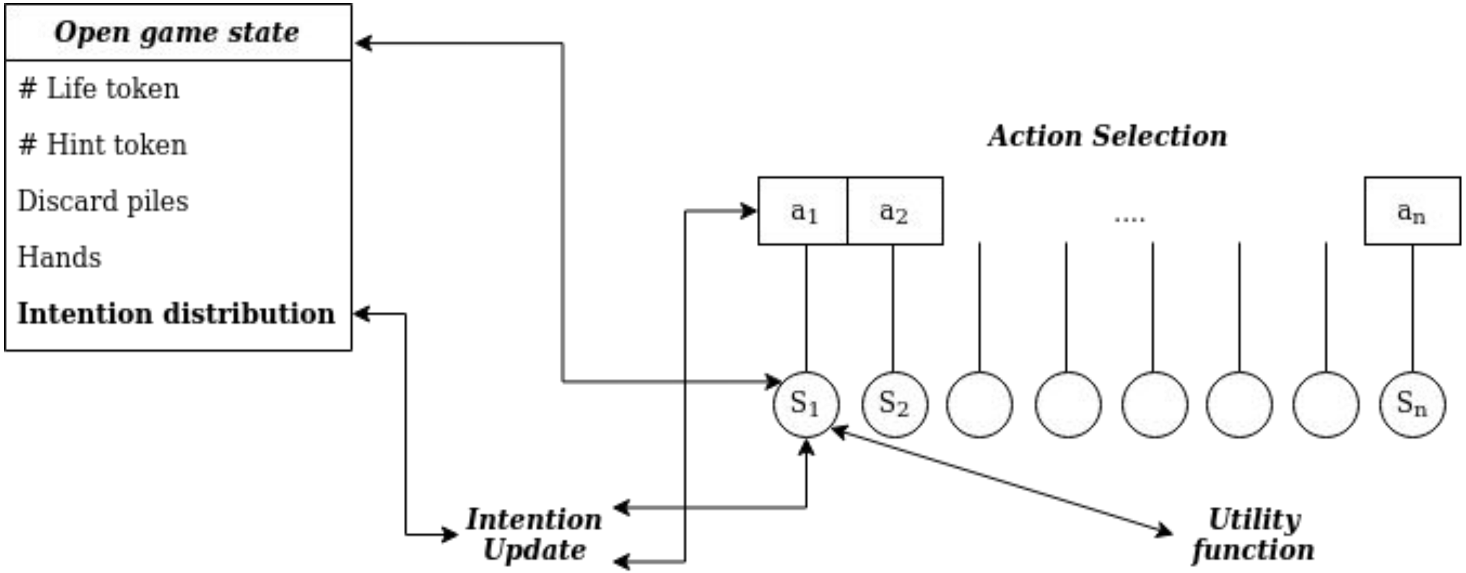
Background

- Use common knowledge as much as possible to avoid complicated n-th order ToM reasoning…
- Intention distribution as compressed representation of history of the game

Big picture



Open game state
Life token
Hint token
Discard piles
Hands
Intention distribution
Action Selection
a1
a2
....
an
S1
S2
Sn
Intention Update
Utility function

Executing the game

- clone https://github.com/moorugi98/hanabi-learning-environment
- in *hanabi-learning-environment*, **pip install .**
- in *examples*, run *python game_example.py*
- most additional functions are implemented in *examples/intention_update.py*
- other basic attributes and methods from the framework in *hanabi_learning_environment/pyhanabi.py*

game_example.py

line 104: `knowledge = intention_update.generate_knowledge(game, state)`

called each time to generate a nested list common knowledge

Rank	red	green	blue	white	yellow
1	3	3	3	3	3
2	2	2	2	2	2
3	2	2	2	2	2
4	2	2	2	2	2
5	1	1	1	1	1

line 119: `intention = intention_update.infer_joint_intention(...`

called each time to update intention

intention_update.py

```
def generate_knowledge(game, state)
```

start full then subtract discarded and played cards, set impossible realizations to 0 and set other card's realization to 0 if a card's realization is fixed

For the actual intention update (`infer_joint_intention`), the implementation just follows the equation

$$P(a|i,r,c) \overset{(1)}{=} \frac{P(a,i|r,c)}{\sum_{a^*} P(a^*,i|r,c)} \overset{(2)}{=} \frac{P(i|a,r,c)P(a|r,c)}{\sum_{a^*} P(i|a^*,r,c)P(a^*|r,c)}$$

$$\overset{(3)}{=} \frac{P(i|r,c_{\text{new}})P(a|r,c)}{\sum_{a^*} P(i|,r,c^*_{\text{new}})P(a^*|r,c)} \overset{(4)}{=} \frac{\exp(\alpha U(i;r,c_{\text{new}}))P(a|r,c)}{\sum_{a^*} \exp(\alpha U(i;r,c^*_{\text{new}}))P(a^*|r,c)}$$

`pragmatic_speaker`

`utility`

uniform dist

Neither Saskia's nor Bianca's utility function depend on k_new!!!

e.g. the more likely that the following cards from co-players can be played if the current card is played, the higher the intention to play should be

$$P(i_{(0,0)}, i_{(0,1)}, \ldots, i_{(\#plyr,\#hand)}|a, c) = \prod_{p \in range(\#plyr), h \in range(\#hand)} P(i_{(p,h)}|a, c)$$

$$P(i_{(p,h)}|a, c) \overset{(1)}{=} \frac{P(a, i_{(p,h)}|c)}{P(a|c)} \overset{(2)}{=} \frac{\sum_{r^*_{(p,h)} \in R_{(p,h)}} P(a, i_{(p,h)}, r^*_{(p,h)}|c)}{\sum_{i^* \in I, r^*_{(p,h)} \in R_{(p,h)}} P(a, i^*, r^*_{(p,h)}|c)} \overset{(3)}{=}$$

$$\frac{\sum_{r^*_{(p,h)} \in R_{(p,h)}} P(a|i_{(p,h)}, r^*_{(p,h)}, c) P(i_{(p,h)}, r^*_{(p,h)}|c)}{\sum_{i^* \in I, r^*_{(p,h)} \in R_{(p,h)}} P(a|i^*, r^*_{(p,h)}, c) P(i^*, r^*_{(p,h)}|c)}$$

$$\overset{(4)}{=} \frac{\sum_{r^*_{(p,h)} \in R_{(p,h)}} P(a|i_{(p,h)}, r^*_{(p,h)}, c) P(i_{(p,h)}|c) P(r^*_{(p,h)}|c)}{\sum_{i^* \in I, r^*_{(p,h)} \in R_{(p,h)}} P(a|i^*, r^*_{(p,h)}, c) P(i^*|c) P(r^*_{(p,h)}|c)}$$

$$
\begin{aligned}
P(a|i_{(p,h)}, r^r_{(p,h)}, c) &\overset{(1)}{=} \frac{P(a, i_{(p,h)}|r^r_{(p,h)}, c)}{P(i_{(p,h)}|r^r_{(p,h)}, c)} \overset{(2)}{=} \\
\frac{P(a, i_{(p,h)}|r^r_{(p,h)}, c)}{\sum_{a^* \in A_r} P(a^*, i_{(p,h)}|r^r_{(p,h)}, c)} &\overset{(3)}{=} \frac{P(i_{(p,h)}|a, r^r_{(p,h)}, c)P(a|r^r_{(p,h)}, c)}{\sum_{a^* \in A_r} P(i_{(p,h)}|a^*, r^r_{(p,h)}, c)P(a^*|r^r_{(p,h)}, c)} \\
\overset{(4)}{=} \frac{P(i_{(p,h)}|r^r_{(p,h)}, c_{\text{new}})P(a|r^r_{(p,h)}, c)}{\sum_{a^* \in A_r} P(i_{(p,h)}|r^r_{(p,h)}, c^*_{\text{new}})P(a^*|r^r_{(p,h)}, c)} &\overset{(5)}{=} \\
\frac{\exp(\alpha U(i_{(p,h)}; r^r_{(p,h)}, c_{\text{new}}))P(a|r^r_{(p,h)}, c)}{\sum_{a^*} \exp(\alpha U(i_{(p,h)}; r^r_{(p,h)}, c^*_{\text{new}}))P(a^*|r^r_{(p,h)}, c)}
\end{aligned}
$$

Solutions

- Think hard about how to create utility functions that makes sense…
- Try it out with NN approximator?

- How to assess quality of produced intention distribution?