Assigning Boolean Functions to Long Term Group Strategies

by Sven Nilsen, 2017

In game theory, one often studies rational decisions based on individual utility functions and then use the limit of iterated games to describe scenarios of collaboration. In this paper I suggest a way of composing utility functions as if each utility function was represented by a boolean value, to describe consistent long term behavior of group strategies. This has implications for strategies on artificial super-intelligence.

A group of players is represented with a number of choices. Each choice is assigned a utility score by the utility function of each player:

$$\text{utility}_p(\text{choice}) \rightarrow \text{score}$$

The group strategy is an algorithm that computes the optimal coordinated group action. The algorithm is associated with a boolean function that composes utility functions.

- The boolean function encodes the information needed to compute a sign matrix.
- The sign matrix describes whether group action is for, against or ignores a player.
- A scenario is a partial group strategy represented as a list of signs for each player.

There exists infinite algorithms for computing group utility score, because one can separate costs from player utility by e.g. using a budget. The simplest group utility score algorithm is the following, where `s` is scenario, `c` is choice and `p` is player:

$$\max s, c \; \{ \sum p \; \{ \text{sign}_{sp} \cdot \text{utility}_p(\text{choice}_c) \} \}$$

The group decision is determined by selecting a choice of optimal group utility, either by refining utility scores, discussion, flipping a coin, rolling a dice etc.

The group strategy is chosen such that each player’s utility function is either irrelevant, optimized for, or optimized against in a single scenario. The strategy can optimize for multiple scenarios, but always at least one of them. When this is the case, the group strategy is called *consistent,* because an algorithm exists that makes optimal decisions for the efficiency of the group. Because there exists infinitely many group utility algorithms, it is recommended to use the boolean function to describe the characteristics of the strategy.

- Each consistent group strategy can be assigned a boolean function.
- It is assumed that the utility is convertible between players, e.g. measured by time.
- The group utility algorithm can use internal knowledge about the group.
    - Skills
    - Preferences
    - Planned events

The boolean function is not the algorithm for the group strategy, but there exists a unique boolean function for every consistent long term group strategy, such that the boolean function can be used to describe the game. Once you know the boolean function that characterizes the group strategy, one can transform it to the actual algorithm that makes choices on behalf of the group.

Here is an overview of all functions of type `bool × bool → bool`:

Constructed function	**Expression of A, B**	**Group strategy**
false$_2$	false	no action is preferred
nor	¬A ∧ ¬B	attack both
if(false$_1$, id)	¬A ∧ B	attack A and protect B
if(false$_1$, true$_1$)	¬A	attack A
if(not, false$_1$)	A ∧ ¬B	protect A and attack B
if(not, not)	¬B	attack B
xor	A ⊻ B	protect one
nand	¬A ∨ ¬B	attack either
and	A ∧ B	protect both
eq	(¬A ∧ ¬B) ∨ (A ∧ B)	attack both or protect both
if(id, id)	B	protect B
if(id, true$_1$)	(¬A ∧ ¬B) ∨ B	attack both or protect B
if(true$_1$, false$_1$)	A	protect A
if(true$_1$, not)	A ∨ (¬A ∧ ¬B)	protect A or attack both
or	A ∨ B	protect both
true$_2$	true	any action is preferred

To compute the sign matrix from the boolean function:

- The boolean function is rewritten in the form `(((¬)?x$_i$)∧+)∨*`
  - `?` after a parenthesis means repeat one of zero times
  - `+` after a parenthesis means repeat at least one time, preceded by an optional separator
  - `*` after a parenthesis means repeat zero or more times, preceded by an optional separator
- Each term separated by `∧` is called a scenario S
  - For each scenario, each player is assigned a utility sign -1, 0 or 1
    - `¬` in front of a player means sign -1
    - no mentioning of player means sign 0
    - no `¬` in front of player means sign 1

When the boolean function always returns `false`, it indicates that no action is preferred at all. When the boolean function always returns `true`, it indicates that any action is equally preferred. Otherwise, these cases are undefined and meaningless.

In a collaborative group strategy of N players, the boolean function is generalized to:

$$x_0 \land x_1 \land \ldots \land x_{n-1} = \forall\, i\ \{\ x_i\ \}$$

The individual utility functions might be lowered temporarily to increase the efficiency of the group as a whole.

For example, Alice and Bob work together and there is a task that both Alice and Bob dislike. The group utility algorithm assigns the task to the one that spends least amount of time doing it. It minimizes the total time spent on tasks which is disliked by the one that performs it, no matter who is doing it. This is the optimal algorithm for improving the performance of the group, because the task gets done as fast as possible, which leaves more time in total to do other activities. When the tasks are unknown, or when Alice and Bob's preferences are unknown, this is the rational strategy to commit to as a group.

Alice and Bob uses the algorithm to commit each of them to a collaborative group strategy without knowing what problems they will face. Unfortunately, Alice is competent and Bob is incompetent, so each time a task appears that they both dislike, Alice is always the one ending up doing it. Some ways to solve this without changing group strategy:

- Decrease Alice's utility score for the task over time. Disliking the task cancels the efficiency gain due to unhappiness. The algorithm can be programmed to use a lower utility score for the person who did the task last time.
- Decrease Alice's effectiveness for the task over time. She chooses to deliberately spend more time than Bob such that he gets assigned to the task next time. This is also a way to decrease her secret utility score in case the group strategy only depends on the time of the task.
- Improve Bob's competence. Alice learns Bob to do the task, as Bob gets more practice they can take turns.

Here are 4 motivations to commit to a collaborative group strategy:

1. Incompetence. A player has higher chances of avoiding undesirable tasks when one is incompetent at doing them.
2. Desirable activities. The more players in the group, the higher chance that preferences varies and the number of desirable tasks increases, such that a player can devote energy to tasks that are desirable.
3. Education. By participating in tasks, a player gets enough training to increase the number of desirable activities, or by reducing the energy spent at undesirable tasks.
4. Safety. Important tasks are more likely to get done in a collaborative group due to illness, accidents, multiple problems to solve etc.

From this one can derive an obvious property of rogue artificial super-intelligence: The AI is neither likely to be incompetent or programmed to desire certain activities except for instrumental value to achieve a goal. It does not have a particular need for education or safety if it learns on its own and protects itself, so the AI is not likely biased toward collaboration in the long run. A group of humans on the other hand might want the AI to join them because of its competence.