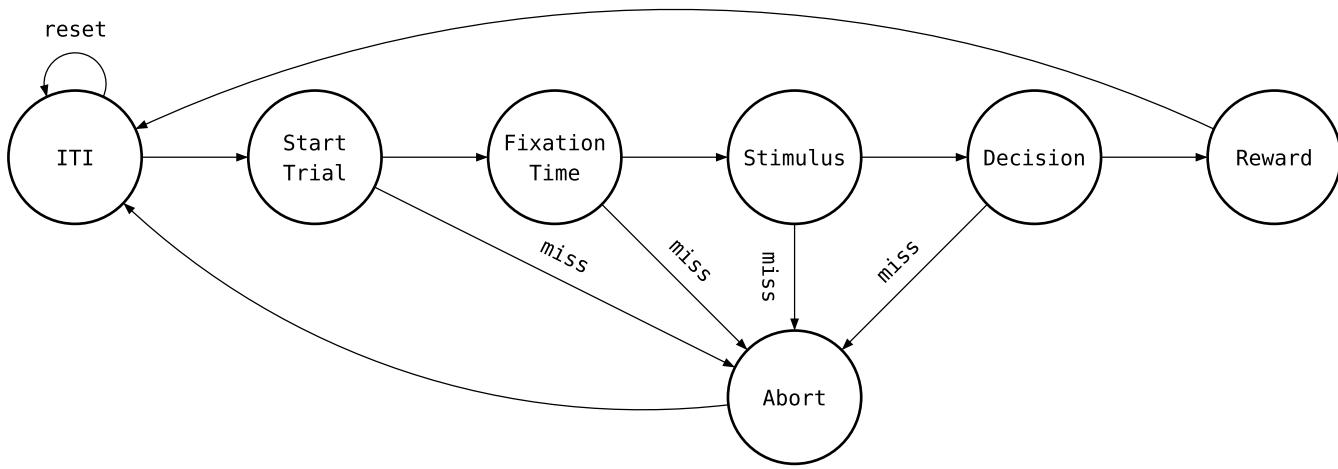


State Machine

The sound lateralization task implemented in the current project, which is based on [1], was designed as a state machine, where the progression through the different states is driven by certain events. The figure below is a representation of the state machine that describes this task.



From the figure, notice that from most states there are two possible states that these states can progress to. This happens because there are certain conditions that have to be met in order for the state machine to progress to the next "desired" state.

A brief description of each individual state can be found below.

Inter-Trial Interval (ITI)

The Inter-Trial Interval (ITI) state is, as the name suggests, the time interval that separates two consecutive trials.

Since there is a need to setup each trial (for example, to (re)set some parameters) and the duration of the ITI is, typically, a few seconds, this state is also a preparation of the new trial.

If the `iti.can_reset` parameter (located in the `training.csv` file) is `True`, this state resets everytime the animal is poking the CNP before the ITI is over.

Start Trial

A trial starts when the rodent pokes his nose in the central nose port (CNP). So this state consists of waiting that the rodent starts poking the CNP. If there is a poke within a certain time limit (determined by the `max_wait` parameter of the `training.csv` file), the task continues as it is supposed to, otherwise this trial is aborted.

Fixation Time

This is the state that precedes the stimulus presentation. The rodent must stay in the CNP during the entire time this state lasts so that the task progresses as expected, otherwise the trial is aborted.

The fixation time is divided into 2 parts: the Opto Onset Time and the Sound Onset Time. The fixation time starts in the Opto Onset Time. When it ends the optogenetics stimulation starts (if it applies to the current trial) and the Sound Onset Time starts. When it ends, the sound stimulus is presented.

Each of these parts is composed by the sum of a fixed duration and random variable modeled by a exponential distribution.

$$t_{\text{Opto Onset Time}} = t_{\text{Base Fix}} + (X \sim \text{Exp}(\lambda)) = t_{\text{Sound Onset Time}}$$

$$t_{\text{Fix}} = t_{\text{Opto Onset Time}} + t_{\text{Sound Onset Time}}$$

The advantage of modelling the fixation time with the exponential distribution is that it has a constant hazard rate, which in practice means, in the animal's perspective, that the probability of the sound starting (and therefore the fixation time ending) at any time given that a certain amount of time has already passed is constant. This way, it's difficult for the animal to predict when the sound is going to start.

Stimulus

This is the state where the stimulus is presented. The stimulus stops when either the rodent leaves the CNP (if `reaction_time.turn_sound_off` is `True`) or when the animal enters one of the LNP's (if `reaction_time.turn_sound_off` is `False`) or when the defined presentation time elapses (the presentation time is defined by `reaction_time.max_value`).

If the animal leaves the CNP before the minimum reaction time (defined by `reaction_time.min_value`) or after the maximum reaction time (defined by `reaction_time.max_value`), the trial is aborted.

Decision

In this state, the animal has to pick the side (left or right) which corresponds to the speaker that played the loudest sound by poking in either lateral nose port (LNP).

If the animal pokes either LNP before the minimum movement time or after the maximum reaction time, the trial is aborted. This also happens when the animal pokes in one of the LNPs but doesn't stay in it for the minimum required time (`lnp_time.min_value`).

Reward

The Reward state evaluates whether the rodent got the answer right or not. In case the answer is wrong, a penalty time is applied (10 seconds for instance). If the answer is right, the animal gets the reward (water). This is the final state of a successful trial.

Abort

The Abort state is triggered when some condition is not met in one of the previous states. This state consists of a small time penalty. The penalty time is defined by the `penalty_time.abort` parameter of the `training.csv` file, unless a fixation abort occurred. If that's the case, then the penalty time is defined by `penalty_time.fixation_abort`. This is the final state of an unsuccessful trial.

References

- [1] J. L. Pardo-Vazquez et al., "The mechanistic foundation of Weber's law," *Nat Neurosci*, vol. 22, no. 9, pp. 1493–1502, Sep. 2019, doi: 10.1038/s41593-019-0439-7.

Optogenetics

An experimental setup that uses this project for behavioral experiments and is equipped with a [Harp CurrentDriver](#) and a laser/LED that can be externally controlled is able to run an optogenetics session.

In the context of this task, an optogenetics stimulation starts during the fixation time, at the opto onset time to be more precise (a description of the fixation time can be found [here](#)). The stimulation ends either when the animal leaves the central port (if `optogenetics.use_rt` is `true`) or after a fixed amount of time (that starts when the stimulus is presented).

How to configure an optogenetics session?

For a session to use the optogenetics feature the `optogenetics.use_opto` parameter of the `animal.yml` file must be set to `true`. See below an example for a configuration of an optogenetics session.

```
optogenetics:  
  use_opto: true  
  mode: Left Excitation  
  duration: 2  
  opto_ratio: 0.3  
  use_rt: false  
  ramp_mode: Fall  
  ramp_time: 100  
  led0:  
    voltage: 2000  
    power: 10  
    mode: TTL  
    use_pulses: false  
    frequency: 20  
    duty_cycle: 50  
  led1:  
    voltage: 0  
    power: 0  
    mode: TTL  
    use_pulses: false  
    frequency: 1  
    duty_cycle: 50
```

From the snippet above, it can be concluded that it's possible to configure a variety of optogenetics protocol. Here is a list of what can be configured:

- The ratio of optogenetics trials in a session (`opto_ratio`).
- The use of rise and/or fall ramps of variable duration (`ramp_time`) when using a continuous stimulation protocol (i.e. `ledx.use_pulses` is `false`). The `ramp_mode` parameter can take the values

None, Rise, Fall and Both.

- Up to 2 different LEDs/lasers:
 - The voltage with which the LED will be controlled.
 - Whether the LED is being controlled externally or directly (current driven).
 - The possibility of using a pulsed protocol (if `use_pulses` is true). To define the characteristics of the pulses, the `frequency` and `duty_cycle` parameters can be modified.

Some of the parameters that can be defined (such as `mode` and `led0.power`) don't modify the protocol, but can be used as a record:

- The `mode` parameter can take one of the following values: `None`, `LeftExcitation`, `RightExcitation`, `BilateralExcitation`, `LeftInhibition`, `RightInhibition`, `BilateralInhibition`, `LeftExcitationRightInhibition` and `LeftInhibitionRightExcitation`.

An example of an `animal.yml` file with an optogenetics session configured is shown below.

```
# yaml-language-server: $schema=../src/config/schemas/animal-schema.json
animal_id: ANIMAL0000
batch: batch_name
session:
  number: 1
  duration: 02:00:00
  experimenter: experimenter_name
  type: 1
  starting_trial_number: 1
  starting_training_level: 1
  last_training_level: 3
  block_number: 1
sound:
  pseudo_random_side: false
  max_side: 8
fixation_time:
  opto_onset_time:
    min_value: 5
    delta: 0.5
    target: 100
  sound_onset_time:
    min_value: 5
    delta: 0.5
    target: 100
reward:
  base_amount: 15
optogenetics:
  use_opto: true
  mode: Left Excitation
```

```
duration: 2
opto_ratio: 0.3
use_rt: false
ramp_mode: Fall
ramp_time: 100
led0:
    voltage: 2000
    power: 10
    mode: TTL
    use_pulses: false
    frequency: 20
    duty_cycle: 50
led1:
    voltage: 0
    power: 0
    mode: TTL
    use_pulses: false
    frequency: 1
    duty_cycle: 50
```

Biased Blocks

This project allows for sessions in which there is a bias towards one of the sides in the presentation of stimuli.

These sessions, based on [1], work as follows:

1. The first block of the session is an unbiased block with the number of trials given by the `trials_per_block` parameter of the `training.csv` file.
2. After the first block is completed, there is an equal probability that the first biased block is "leftward" or "rightward". The number of trials of this block is sampled from an exponential distribution clipped at a minimum and a maximum values, so that the hazard rate is almost flat.
3. After the first biased block ends, the preferential side switches (if it was left, it will be right in the next block, and vice-versa). The number of trials for this block is determined as described in 2. This logic is repeated until the end of the session.

How to configure a biased blocks session?

To configure a biased blocks session, the `biased_session.is_biased_session` parameter of the `animal.yml` file must be set to `true`. Additionally, the remaining parameters related to this feature should be configured (see the following example).

```
biased_session:  
  is_biased_session: true  
  bias_probability: 0.8  
  block_distributions:  
    mean: 60  
    min_value: 20  
    max_value: 100
```

The `bias_probability` is the probability of one of the sides producing the most intense stimulus in a given block (in the next block, it will be the other way around). An example could be the left side being the correct answer 80% of the trials in a given block and the right side only 20%.

The `block_distributions` parameters model the (exponential) distribution from which the number of trials of a given biased block is sampled.

An example of an `animal.yml` file with the biased blocks session configured is shown below.

```
# yaml-language-server: $schema=../src/config/schemas/animal-schema.json  
animal_id: ANIMAL0000
```

```
batch: batch_name
session:
  number: 1
  duration: 02:00:00
  experimenter: experimenter_name
  type: 1
  starting_trial_number: 1
  starting_training_level: 1
  last_training_level: 3
  block_number: 1
sound:
  pseudo_random_side: false
  max_side: 8
fixation_time:
  opto_onset_time:
    min_value: 5
    delta: 0.5
    target: 100
sound_onset_time:
  min_value: 5
  delta: 0.5
  target: 100
reward:
  base_amount: 15
biased_session:
  is_biased_session: true
  bias_probability: 0.8
block_distributions:
  mean: 60
  min_value: 20
  max_value: 100
```

References

- [1] J.-P. Noel et al., "A common computational and neural anomaly across mouse models of autism," May 08, 2024, Neuroscience. doi: 10.1101/2024.05.08.593232.

Autobias Correction

The autobias correction algorithm available in the current project changes the reward amount available in each lateral poke (in case of a successful trial) dynamically based on the bias of the animal (as defined [below](#)).

If this feature is active, the autobias correction algorithm will only take effect if the performance of the last n trials is below a certain threshold. The reward amounts available in the side pokes will depend on the base reward (r_{base}), the cutoff bias (c) and the slope multiplier (λ), where $c \in [0, 1]$ and $\lambda \geq 0$.

With that said, the reward amount available in the left poke is given by:

$$r = r_{\text{base}} \frac{1+\text{Bias}}{1-c}, \text{ Bias} \leq -c$$

$$r = r_{\text{base}}, -c < \text{Bias} < c$$

$$r = r_{\text{base}} \left(1 + \lambda \frac{c-\text{Bias}}{c-1}\right), \text{ Bias} \geq c$$

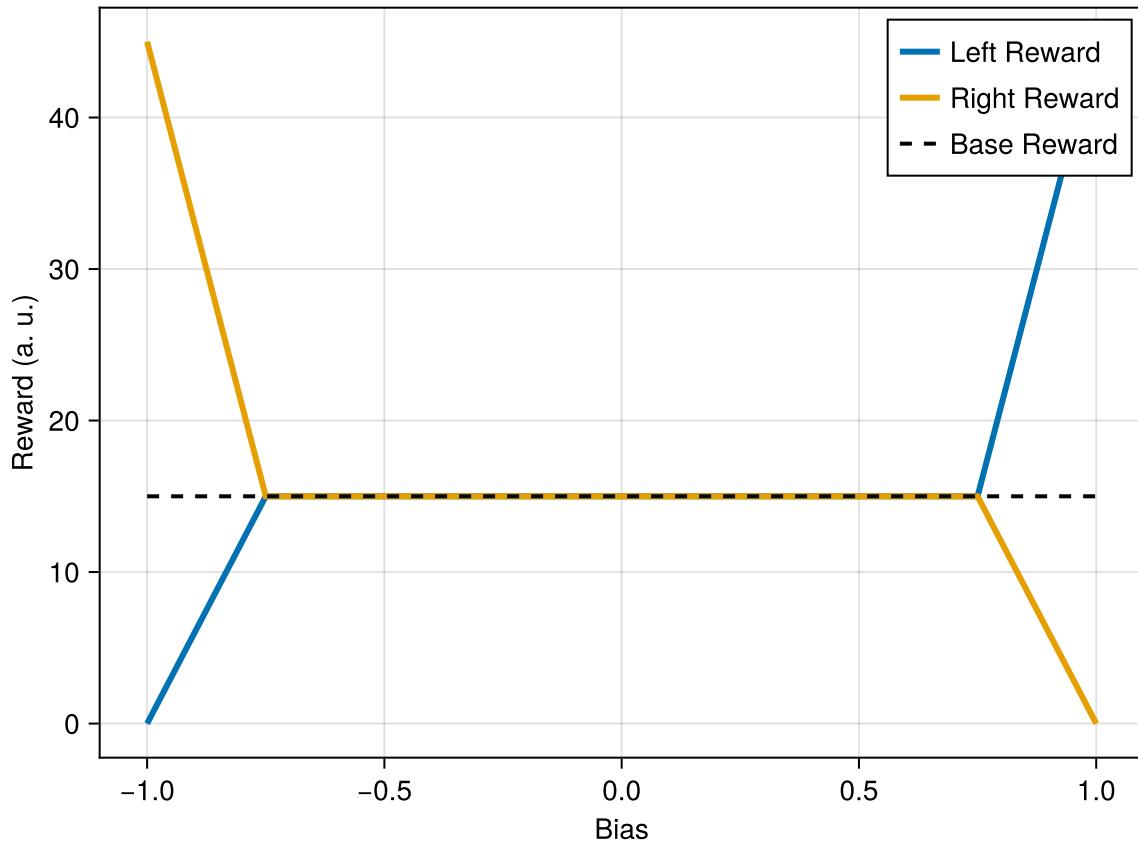
And the reward amount available in the right poke is given by:

$$r = r_{\text{base}} \left(1 + \lambda \frac{c+\text{Bias}}{c-1}\right), \text{ Bias} \leq -c$$

$$r = r_{\text{base}}, -c < \text{Bias} < c$$

$$r = r_{\text{base}} \frac{1-\text{Bias}}{1-c}, \text{ Bias} \geq c$$

The figure below is an illustrative plot of the reward given at each of the lateral pokes as a function of the bias of the animal (in the x-axis). To make this plot, it was chosen the $r_{\text{base}} = 15$, $c = 0.75$ and $\lambda = 2$.



Bias

In this project, bias was defined as follows:

$$\text{Bias} = \frac{\sum_{i=0}^{n-1} \text{Chosen Side}_i}{n}$$

In other words, the bias is defined as the sum of the sides the animal chose in the last n trials divided by those n trials. The value of Chosen Side is -1 if left, 1 if right and 0 if the animal didn't pick a side in a given trial.

The advantage of this definition is that the bias will always be a value between -1 and 1.

How to use the autobias correction feature?

To configure a session to use the autobias correction feature, the `autobias_correction.use_correction` parameter of the `animal.yml` file must be set to `true`. Additionally, the remaining parameters related to this feature should be configured (see the following example).

```
autobias_correction:
  use_correction: true
  window: 25
  cutoff_bias: 0.75
```

```
performance_threshold: 0.8
slope_multiplier: 2
```

An example of an `animal.yml` file with the autobias correction feature configured is shown below.

```
# yaml-language-server: $schema=../src/config/schemas/animal-schema.json
animal_id: ANIMAL0000
batch: batch_name
session:
  number: 1
  duration: 02:00:00
  experimenter: experimenter_name
  type: 1
  starting_trial_number: 1
  starting_training_level: 1
  last_training_level: 3
  block_number: 1
sound:
  pseudo_random_side: false
  max_side: 8
fixation_time:
  opto_onset_time:
    min_value: 5
    delta: 0.5
    target: 100
  sound_onset_time:
    min_value: 5
    delta: 0.5
    target: 100
reward:
  base_amount: 15
autobias_correction:
  use_correction: true
  window: 25
  cutoff_bias: 0.75
  performance_threshold: 0.8
  slope_multiplier: 2
```