Example action:

```
actions[IActionStHP]
  = Action(IChar1, // character
           EAnimation::StHP, // animation
           {}, // collision box (note that this is an std::optional)
           Hitbox({ // hitbox
               Hitbox::make_pair(1, {}),
               Hitbox::make_pair(2, {Box(0.0, 100.0, 150.0, 200.0)})}),
           Hitbox(), // hurtbox
           10, // damage
           0, // blockAdvantage
           0, // hitAdvantage
           9, // lockedFrames (number of frames before player can cancel)
           12, // animationLength
           ActionType::Other, // ActionType
           FVector(0, 0, 0), // velocity
           6, // specialCancelFrames
           {{Button::QCFP, HActionSpecial}}, // chains
           200.0 // knockdownDistance
           );
```

1. `character` is obvious.
2. `animation` fairly obvious.
3. `collision` is an std::optional. A value of `{}` means "None", which means the collision box will default to the character's default collision box defined in their `Character` object. The None value is good for most cases. If the character moves or takes a large step forward during the action, and `velocity` is set to 0, then `collsion` should instead be a `Hitbox` object where the boxes represent the collision boxes. The collision boxes should follow the character as they move.

   If the character moves with a constant speed during the action, `collision` can be left as `{}` and `velocity` can be set to a nonzero value instead. Logic will move the whole character and their boxes by `velocity` every frame. Note that currently, if `velocity` is 0, the player will always return to their starting position after the action ends regardless of how the `collision` boxes move.
4. `hitbox` defines all the boxes that, when colliding with the opponents hurt or collision box, actually result in a hit. A `Hitbox` is a list of pairs. Each pair is a pair of (end frame, {boxes...}). The end frame is the last frame that the boxes described in {boxes...} are active. The boxes become active when the previous pair becomes inactive. Example:

   ```
   Hitbox({
       // {} means no boxes. This is the first pair, so it's boxes
       // (which there happen to be none) become active on frame 0.
       // end frame is 1, so they are active only on frames 0 and 1.
       Hitbox::make_pair(1, {}),

       // This Hitbox contains one box. The previous pair ended on
       // frame 1, so this pair becomes active on frame 2. The end
       // frame for this pair is 2, so frame 2 is also its last active
       // frame. In other words, this pair is active for exactly one
       // frame: frame 2.
   ```

```
        Hitbox::make_pair(2, {Box(0.0, 100.0, 150.0, 200.0)})

        // The animation may be longer than 3 frames long, but we don't
        // have to specify empty hitboxes to cover the remaining
        // duration. We can stop at the last nonempty hitbox.
      }),
```

5. `hurtbox` is a `Hitbox` object defined just like `hitbox` was, but `hurtbox` specifies the boxes that when collided with the opponents hitbox will result in a hit.
6. `damage` is simply the amount of damage the move does.
7. `blockAdvantage` is the amount of "advantage" the attacker has when their attack is blocked. It can be negative to mean disadvantage.

   E.g. a value of 3 means that the attacker can act 3 frames earlier than the blocking player. Specifically, it means that the attacker's `lockedFrames` will expire 3 frames earlier than the blocking player comes out of block stun. If `specialCancelFrames` is less than `lockedFrames`, then the attack can cancel into a chain move even ealier than 3 frames.
8. `hitAdvantage` is the amount of "advantage" the attacker has when their attack lands.
9. `lockedFrames` is the number of frames before the player can act again.
10. `animationLength` is the length of the full animation. If `lockedFrames` is less than `animationLength`, then it means that the player can perform actions like walking before the full animation has played.

Note that all the remaining fields are optional.

11. `type` is an `ActionType`. Most moves will use `ActionType::Other` (default).
12. `velocity` is the amount that the player moves each frame that they are in this action. Defaults to 0.
13. `specialCancelFrames` is the number of frames before the player can perform a move listed in `chains`.
14. `chains` is an `std::map<enum Button, HAction>`. It maps "buttons" to actions. Here a "button" can be a full command like QCFP. E.g.

    ```
    {{Button::QCFP, HActionFJump}}
    ```

    means that if the player performs a QCFP after `specialCancelFrames` have passed, they will jump.

    ```
    {{Button::LP, HActionStLP}}
    ```

    means that if the player performs a LP after `specialCancelFrames` have passed, they will do a StLP.
15. `knockdownDistance` is the distance that the player who is hit gets launched over. A value of -1 (default) means no knockdown. a nonzero value means that the player who is hit will enter the knockdown action and fly `knockdownDistance` away.