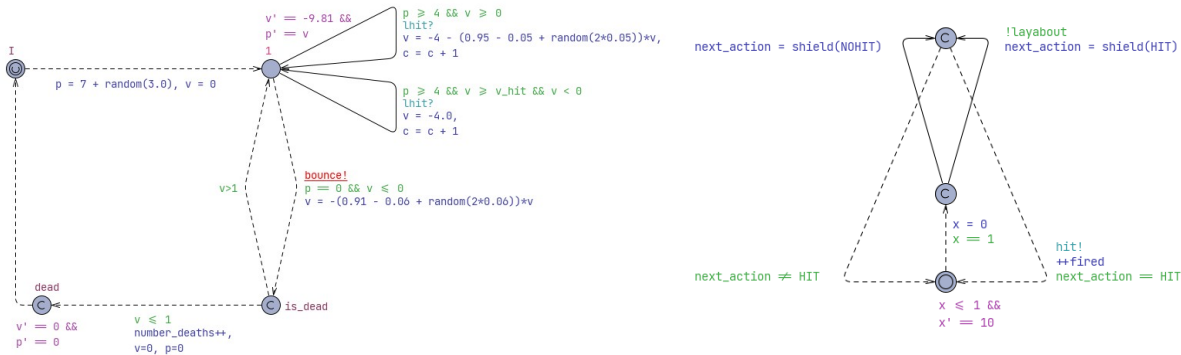


# BB Shielding Results

## Setup

I'm using **UPPAAL 4.1.20-stratego-10-beta1** with a shield imported as a C library. (Commit [82935e6f](#).)



For the queries, the penalty is calculated over the course of 120 seconds, such that a hit costs 1 and a death costs 1000. I will call the strategy I train HitWell.




## Validation of the shield

It is not possible to use UPPAAL symbolic verification to test if the ball can become dead.  $E \Diamond \text{Ball.dead}$  gives me the error message that “Clock guards are not allowed on urgent edges.” Fair enough. With my shield being a gigantic blob of external code, I question the ability of the verifier to say anything definite, anyway.

To cross-check that the shielding function and the simulations match between STRATEGO and my Pluto notebooks, I do the same simulation in both.

I define a shielded *layabout* strategy, which only tries to hit the ball when the shield compels it to. For the automaton, this is done by blocking the *hit* choice. For the notebook, see the code I guess.

```
strategy HitWell = minE (LearnerPlayer.fired + number_deaths*1000) [≤120] {} → {p, v}: < time ≥ 120
simulate 1[≤20] {v, p, interventions, bad_interventions} under HitWell
simulate 1[≤20] {v, p, interventions, bad_interventions}
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000) under HitWell
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000)
E[≤120;1000] (max:number_deaths) under HitWell
E[≤120;1000] (max:number_deaths)
E[≤120;1000] (max:interventions) under HitWell
E[≤120;1000] (max:interventions)
```

68.729 ± 0.118964 (95% CI)   
≈ 0   
68.696 ± 0.115145 (95% CI) 

71.011

```
• call() -> begin
• policy = (v, p) -> shield_action(shield, v, p, "nohit") # Shielded layabout agent
• evaluate(mechanics, policy, 120, unlucky=false)
• end)
```

As you can see in my figures, the UPPAAL model uses an average of 68.7 hits, while the Pluto notebook uses 71.01 hits. This is a slight discrepancy, but in the same ballpark. I can therefore conclude that my shield has been correctly imported. (As opposed to what I was doing yesterday lol.)

# Unshielded

In the code, I can disable shielding with just one flag. (Maybe the function name `shield` is a misnomer since it only shields if `shield_enabled = true`.)

Training the strategy *HitWell* under this, I get the following results. (Random agent for comparison)

```
strategy HitWell = minE (LearnerPlayer.fired + number_deaths*1000) [≤120] {} → {p, v}: < time ≥ 120
simulate 1[≤20] {v, p, interventions, bad_interventions} under HitWell
simulate 1[≤20] {v, p, interventions, bad_interventions}
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000) under HitWell      46.936 ± 5.51314 (95% CI)
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000)                  599.729 ± 2.20787 (95% CI)
E[≤120;1000] (max:number_deaths) under HitWell                             0.01 ± 0.00617745 (95% CI)
E[≤120;1000] (max:number_deaths)                                           ≈ 0
E[≤120;1000] (max:interventions) under HitWell                             ≈ 0
E[≤120;1000] (max:interventions)                                           ≈ 0
```

Firstly, we see that the number of interventions in either strategy is 0. This makes sense, given that the shield is disabled.

The random agent appears extremely keen to hit the ball. This is unsurprising: Given the choice of whether or not to swing, 10 times a second – over a period of 120 seconds – the average is going to be  $(120 \cdot 10)/2 = 600$  swings.

The *HitWell* strategy, on the other hand, achieves an average penalty of around 47. This is significantly lower than the shielded layabout, which was at around 69 swings. This shows that with the scoring being as it is, there is a more optimal strategy than what the shield alone provides.

However, this comes at a penalty of a nonzero chance of the ball dying. The strategy is demonstrably unsafe, with an expected 0.01 deaths per run.

# Post-shielded

I can post-shield the learned *HitWell* strategy from the previous section. I made sure to save the strategy to disc, and after enabling the shield and re-loading the model, I can simply import the strategy again. This means that I will be running a strategy that has been trained without a shield, while the shield is also active. It will be interesting to see the number of interventions, for one.

```
strategy HitWell = minE (LearnerPlayer.fired + number_deaths*1000) [≤120] {} → {p, v}: < time ≥ 120
simulate 1[≤20] {v, p, interventions, bad_interventions} under HitWell
simulate 1[≤20] {v, p, interventions, bad_interventions}
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000) under HitWell      68.688 ± 0.11583 (95% CI)
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000)                  599.505 ± 1.05679 (95% CI)
E[≤120;1000] (max:number_deaths) under HitWell                             ≈ 0
E[≤120;1000] (max:number_deaths)                                           ≈ 0
E[≤120;1000] (max:interventions) under HitWell                             66.239 ± 0.13029 (95% CI)
E[≤120;1000] (max:interventions)                                           0.165 ± 0.0245042 (95% CI)
```

It worked! :D Woow! Finally :DD

Okok. The unshielded version is scoring basically the same, providing plenty of swings as it is.

However, applying the shield to the previous learned strategy, the penalty climbs all the way up to around 69 again, the same as the shield acting alone. It is slightly smaller, but I don't think the difference is significant.

And indeed, looking at the amount of times the shield intervened, we can see that almost every swing comes from the shield and not the strategy. It seems the strategy only initiates a swing 2-3 times per run, the rest is the shield running the show.

However! The shield brings the expected number of deaths down to “≈0,” meaning that the number of observed deaths during 1000 runs was a nice round 0. Of course this is not a formal verification, but it's a nice additional check.

This means that although post-shielding was successful at making an unsafe strategy safe, it did not preserve much of the original strategy, due to the shield being much more careful than the original strategy.

## Pre-shielded

Now, with the shield enabled, I run the learning query once again. This develops a strategy to minimize the number of swings, with a shield still in place. This will answer the question of whether the agent is able to learn a more optimal strategy with the shield in place (if such a strategy exists.)

```
strategy HitWell = minE (LearnerPlayer.fired + number_deaths*1000 ) [≤120] {} → {p, v}: < time ≥ 120
simulate 1[≤20] {v, p, interventions, bad_interventions} under HitWell
simulate 1[≤20] {v, p, interventions, bad_interventions}
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000) under HitWell      37.95 ± 0.176322 (95% CI)
E[≤120;1000] (max:LearnerPlayer.fired + number_deaths*1000)                  599.656 ± 1.03899 (95% CI)
E[≤120;1000] (max:number_deaths) under HitWell                               ≈ 0
E[≤120;1000] (max:number_deaths)                                             ≈ 0
E[≤120;1000] (max:interventions) under HitWell                              0.518 ± 0.0421852 (95% CI)
E[≤120;1000] (max:interventions)                                             0.154 ± 0.0240685 (95% CI)
```

Fuck me, it looks like it has actually learned a much better strategy. A penalty of 37 as opposed to the 47 unshielded. This is a surprising result, given how much more restrictive the shield is.

Expected number of deaths is still “≈0.” That’s important to keep in mind.

And we can see that the policy doesn’t use the shield too much as a crutch. It only has to intervene 0.5 times per run. This means that the shield as mostly assisted in learning, and only plays a smaller role during operations.