```
paths.OptionalEnsemble(paths.AllOutXEnsemble(C7eq | alpha_r)),
trial_recrossings = recrossing.split(step.trials[0].trajectory)
```

```
trajectory = recrossing_steps[0].active[0].trajectory
```

```
paths.LengthEnsemble(1) & paths.AllInXEnsemble(alpha_r),
recrossing_steps = [step for step in storage.steps
```

```
# define function to identify accepted recrossings
```

```
paths.LengthEnsemble(1) & paths.AllInXEnsemble(C7eq)
```

```
return len(trial_recrossings) > 0 & step.accepted
```

```
storage = paths.AnalysisStorage("tps_file.nc")
```

```
recrossing = paths.SequentialEnsemble([
print len(recrossing_steps)  # output
```

```python
import openpathsampling as paths
```

```python
C7eq = storage.volumes['C7eq']
```

```
def accepted_recrossing(step):
```

```
alpha_r = storage.volumes['alpha_r']
```

```
# look at the first trajectory
```

```
if accepted_recrossing(step)]
```

```
# find all relevant MC steps
# load states from storage
# create the ensemble
```

])