Acts::Propagator API

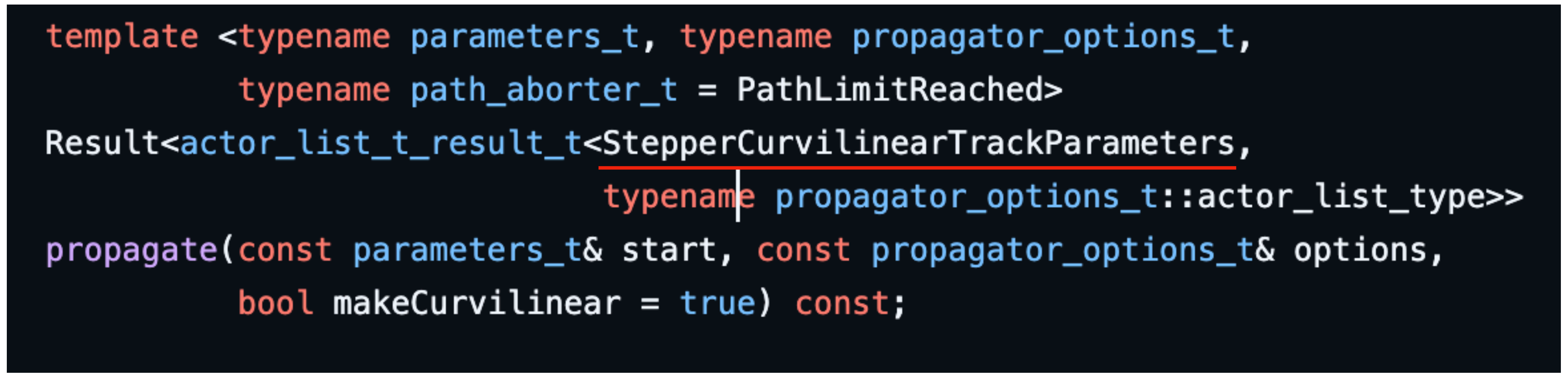
Joe Osborn
Brookhaven National Laboratory
January 28, 2025

Propagator API

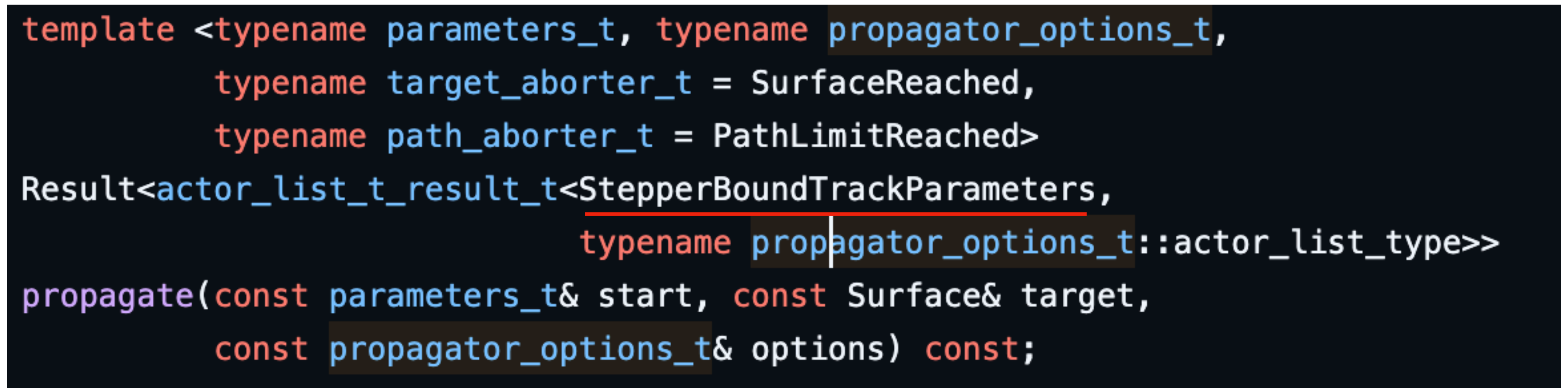
- There are effectively two ways to call the propagator machinery

  1. Propagation limited by abort condition or max number of propagation steps. Returns curvilinear parameters

  2. Propagation to a specific surface. Returns bound parameters on surface



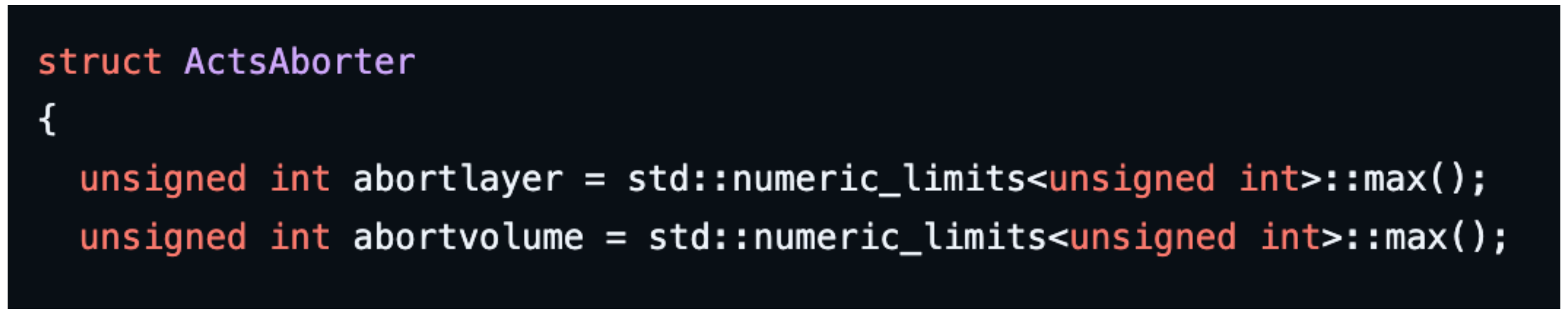
```
template <typename parameters_t, typename propagator_options_t,
          typename path_aborter_t = PathLimitReached>
Result<actor_list_t_result_t<StepperCurvilinearTrackParameters,
                             typename propagator_options_t::actor_list_type>>
propagate(const parameters_t& start, const propagator_options_t& options,
          bool makeCurvilinear = true) const;
```



```
template <typename parameters_t, typename propagator_options_t,
          typename target_aborter_t = SurfaceReached,
          typename path_aborter_t = PathLimitReached>
Result<actor_list_t_result_t<StepperBoundTrackParameters,
                             typename propagator_options_t::actor_list_type>>
propagate(const parameters_t& start, const Surface& target,
          const propagator_options_t& options) const;
```

Return Parameters

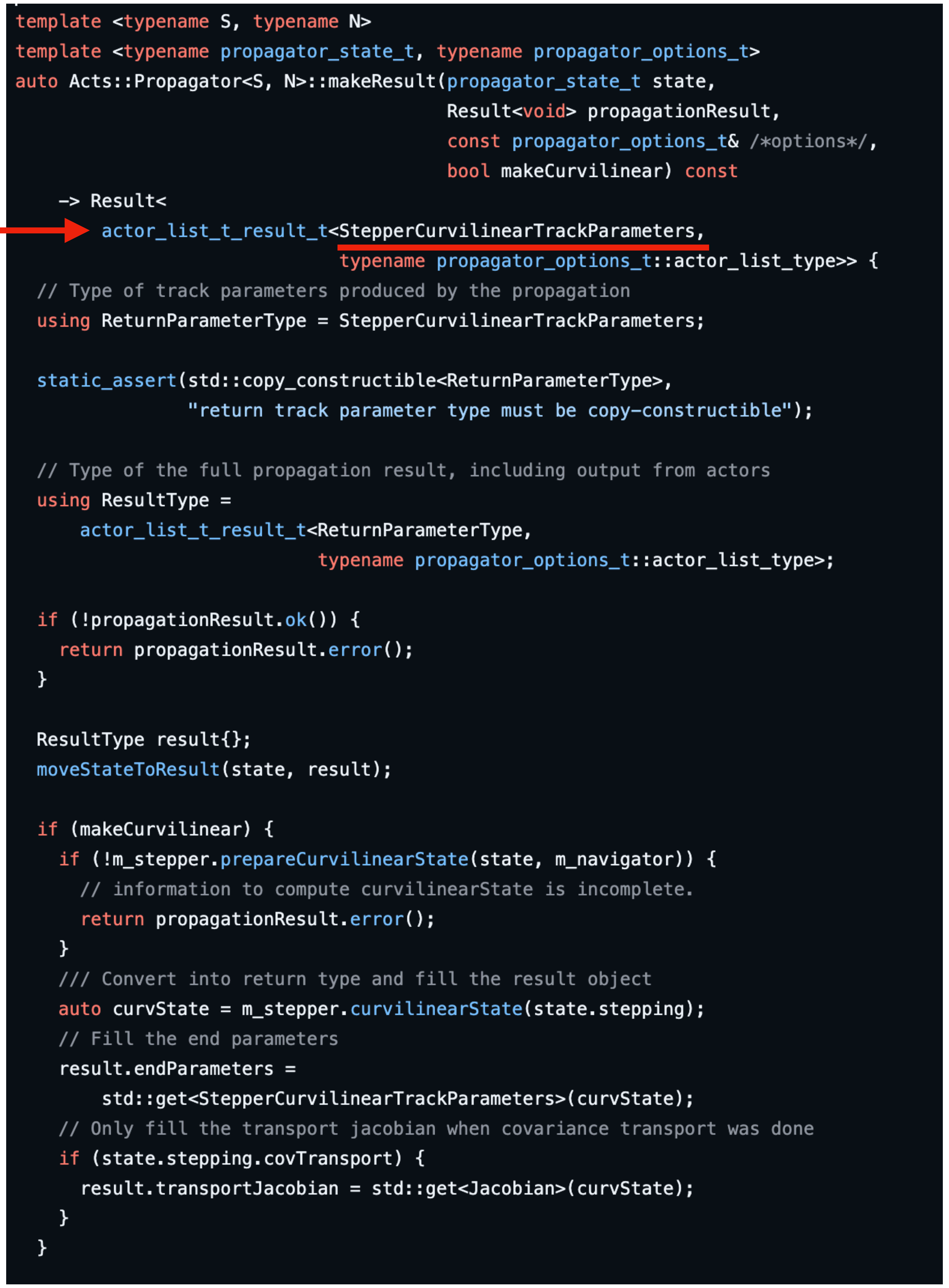
- We have a custom aborter that aborts in a user provided layer
  - Useful for determining track state when you want to know a track state where a measurement does not exist
- Problem - the track parameters returned are curvilinear! They aren't on the surface, so a position is not defined



```
struct ActsAborter
{
  unsigned int abortlayer = std::numeric_limits<unsigned int>::max();
  unsigned int abortvolume = std::numeric_limits<unsigned int>::max();
```

Paths Forward

- Not sure if there is a design choice or specific reason why the return type is fixed
- There is even a boolean in the argument list to makeCurvilinear, but no else to return bound parameters
- Regardless the type of return parameter is fixed, so adding an else {return boundParams;} does not work anyway
- Thoughts? Discussion?



```cpp
template <typename S, typename N>
template <typename propagator_state_t, typename propagator_options_t>
auto Acts::Propagator<S, N>::makeResult(propagator_state_t state,
                                        Result<void> propagationResult,
                                        const propagator_options_t& /*options*/,
                                        bool makeCurvilinear) const
    -> Result<
        actor_list_t_result_t<StepperCurvilinearTrackParameters,
                              typename propagator_options_t::actor_list_type>> {
  // Type of track parameters produced by the propagation
  using ReturnParameterType = StepperCurvilinearTrackParameters;

  static_assert(std::copy_constructible<ReturnParameterType>,
                "return track parameter type must be copy-constructible");

  // Type of the full propagation result, including output from actors
  using ResultType =
      actor_list_t_result_t<ReturnParameterType,
                            typename propagator_options_t::actor_list_type>;

  if (!propagationResult.ok()) {
    return propagationResult.error();
  }

  ResultType result{};
  moveStateToResult(state, result);

  if (makeCurvilinear) {
    if (!m_stepper.prepareCurvilinearState(state, m_navigator)) {
      // information to compute curvilinearState is incomplete.
      return propagationResult.error();
    }
    /// Convert into return type and fill the result object
    auto curvState = m_stepper.curvilinearState(state.stepping);
    // Fill the end parameters
    result.endParameters =
        std::get<StepperCurvilinearTrackParameters>(curvState);
    // Only fill the transport jacobian when covariance transport was done
    if (state.stepping.covTransport) {
      result.transportJacobian = std::get<Jacobian>(curvState);
    }
  }
```