```
// This is a Spec Explorer coordination script (Cord version 1.0).
// Here is where you define configurations and machines describing the
// exploration to be performed.

using SpecExplorer6.Sample;

/// Contains actions of the model, bounds, and switches.
config Main
{
    /// The two implementation actions that will be modeled and tested
    action abstract static void Accumulator.Add(int x);
    action abstract static int Accumulator.ReadAndReset();

    switch StepBound = 128;
    switch PathDepthBound = 128;
    switch TestClassBase = "vs";
    switch GeneratedTestPath = "..\\SpecExplorer6.TestSuite";
    switch GeneratedTestNamespace = "SpecExplorer6.TestSuite";
    switch TestEnabled = false;
    switch ForExploration = false;
}


/// This configuration provides a domain for
/// parameter in the previous one.
config ParameterCombination: Main
{
    action abstract static void Accumulator.Add(int x)
        where x in {0..2};
}

/// Constructs a machine from the model program.
/// Since the model is not finite, this machine explodes
/// and exploration is stopped by a bound.
/// Switch ForExploration makes the machine appear in Exploration Manager.
machine AccumulatorModelProgram() : Main where ForExploration = true
{
    construct model program from ParameterCombination
    where scope = "SpecExplorer6.AccumulatorModelProgram" //The value of the namespace switch can be a .Net ↙
     namespace or a fully-qualified class name.
}

/// Defines a scenario for slicing.
/// When explored on its own, this machine
/// leaves all its parameters unexpanded.
machine DoubleAddScenario() : Main where ForExploration = true
{
    //Omitting the parenthesis for an action invocation
    //is equivalent to setting all its parameters to _ (unknown).
    (Add(_); Add; ReadAndReset)*
}

/// Selects the slice of the model program
/// that matches the scenario. The model program
/// supplies all parameter and state data omitted from the scenario.
machine SlicedAccumulatorModelProgram() : Main where ForExploration = true
{
    DoubleAddScenario || AccumulatorModelProgram // (synchronized) parallel composition
}

/// Builds a machine resulting from a link coverage traversal
/// of the sliced model program. It can be explored or saved as a
/// C# test suite that can be run in a VSTS unit test project
/// (by pushing the Generate Test Code button in the Exploration
/// Manager toolbar). Most tests should fail, since the sample
/// implementation is empty.
```

```
machine AccumulatorTestSuite() : Main where ForExploration = true, TestEnabled = true
{
    construct test cases where strategy = "ShortTests" for SlicedAccumulatorModelProgram()
}
```