

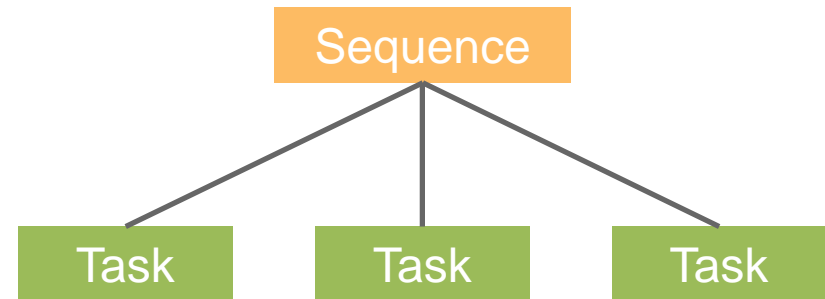
A 3D wireframe model of a roller coaster track, showing a complex layout with multiple loops, curves, and drops. The track is rendered in a light gray wireframe style, with some sections highlighted in black. The track starts on the left, goes up and over a loop, then down and around a curve, then up and over another loop, and finally down and around a curve before ending on the right. The track is surrounded by a grid of lines, suggesting a 3D coordinate system.

Sequencer Service

Define, Execute and Monitor Structured Tasks

■ Tasks

- check device state
 - connectivity
 - actual value vs target value
- power on/off a device
- change settings
 - set value of device A to X



■ Combine tasks into more complex **Sequences**

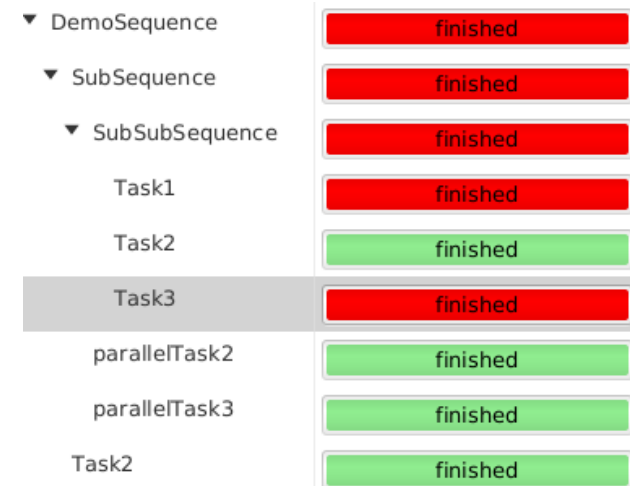
- Check availability of multiple devices
- set values for a set of devices, then do ... and then ...
- arbitrary complexity with nested sequences

Sequences and Tasks

Block	Cursor	Brea...	Progress	
▼ DeviceSetupDemo			finished	
▼ setupPowersupplies			finished	
▼ checkPowersupplies			finished	
checkPowersupplies(0): PS_A			finished	
checkPowersupplies(1): PS_B			finished	
checkPowersupplies(2): PS_C			finished	
▼ setupMagnets			finished	
▼ powerOnDevices			finished	
powerOnDevices(3): MAGNET_A			finished	
powerOnDevices(4): MAGNET_B			finished	
powerOnDevices(5): MAGNET_C			finished	
powerOnDevices(6): MAGNET_D			finished	
powerOnDevices(7): MAGNET_E			finished	
powerOnDevices(8): MAGNET_F			finished	
▼ setTargetValue			finished	
setTargetValue(9): MAGNET_A			finished	
setTargetValue(10): MAGNET_B			finished	

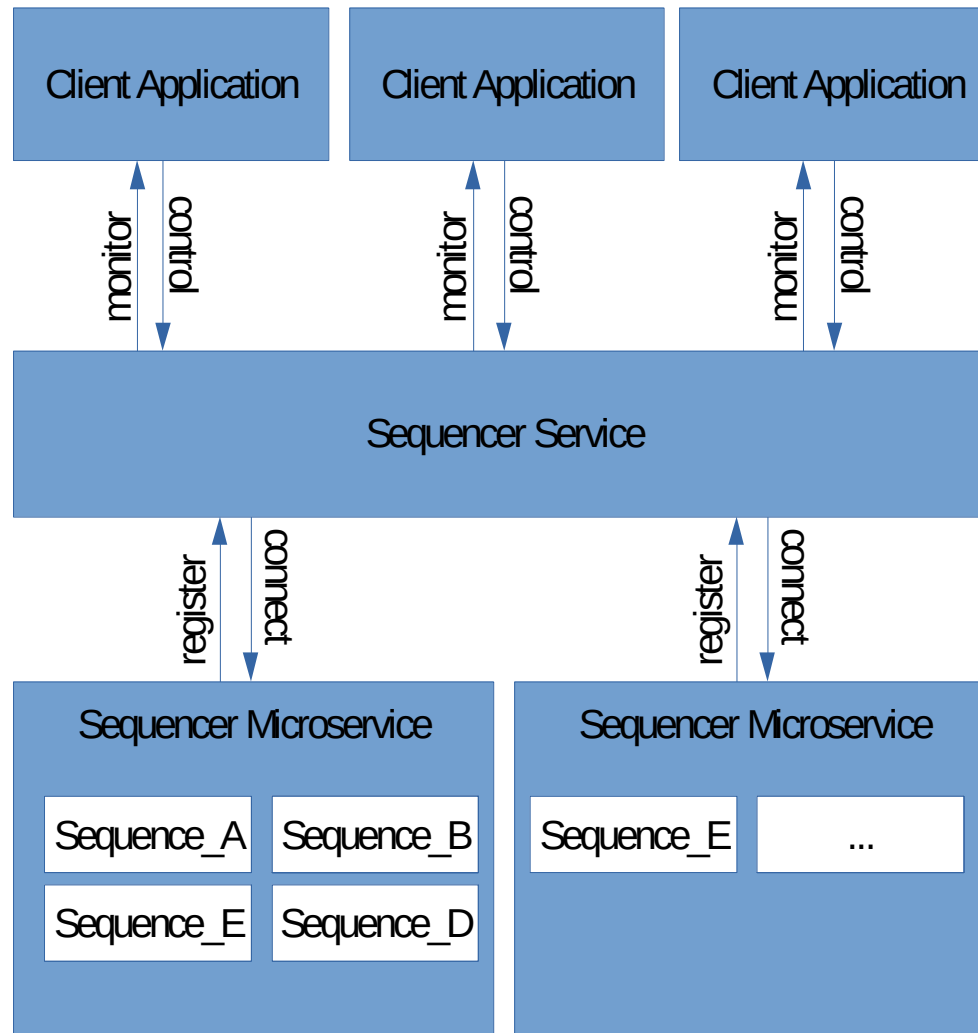
Automatically Expand/Collapse Show root

- System Setup
 - define sequences for repetitive setup activities
 - helps to prevent errors through automation
- Testing
 - define test cases as sequences
 - combine test cases into test scenarios
 - -> identify errors



- Framework for programming and deploying sequences
 - Sequences are written by or with the help of domain experts
 - Some programming skills (Java) required
- Environment for executing and monitoring Sequences
 - Once deployed sequences can be executed and monitored by operators
 - Provide a graphical user interface

Sequencer Service Prototype



Parameter Configuration



Available Instances

checkPowersupply x

- DemoSequence
- DeviceSetupDemo
- Executable Leafs Demo Mission (parametrized)
- LinearSequence
- ParallelSequence
- checkPowersupply**

Mission Instance

Search

Mandatory

checkDcMode

checkVoltageMax 10.0

checkCurrentMin 0.0

checkVoltageMin 0.0

device YR03BG1T

checkCurrentMax 5.0

accuracyLimit 0.001

Optional

Instantiate

Block	
▶ checkPowersupply	

Pause

Step Over

Step Into

Skip

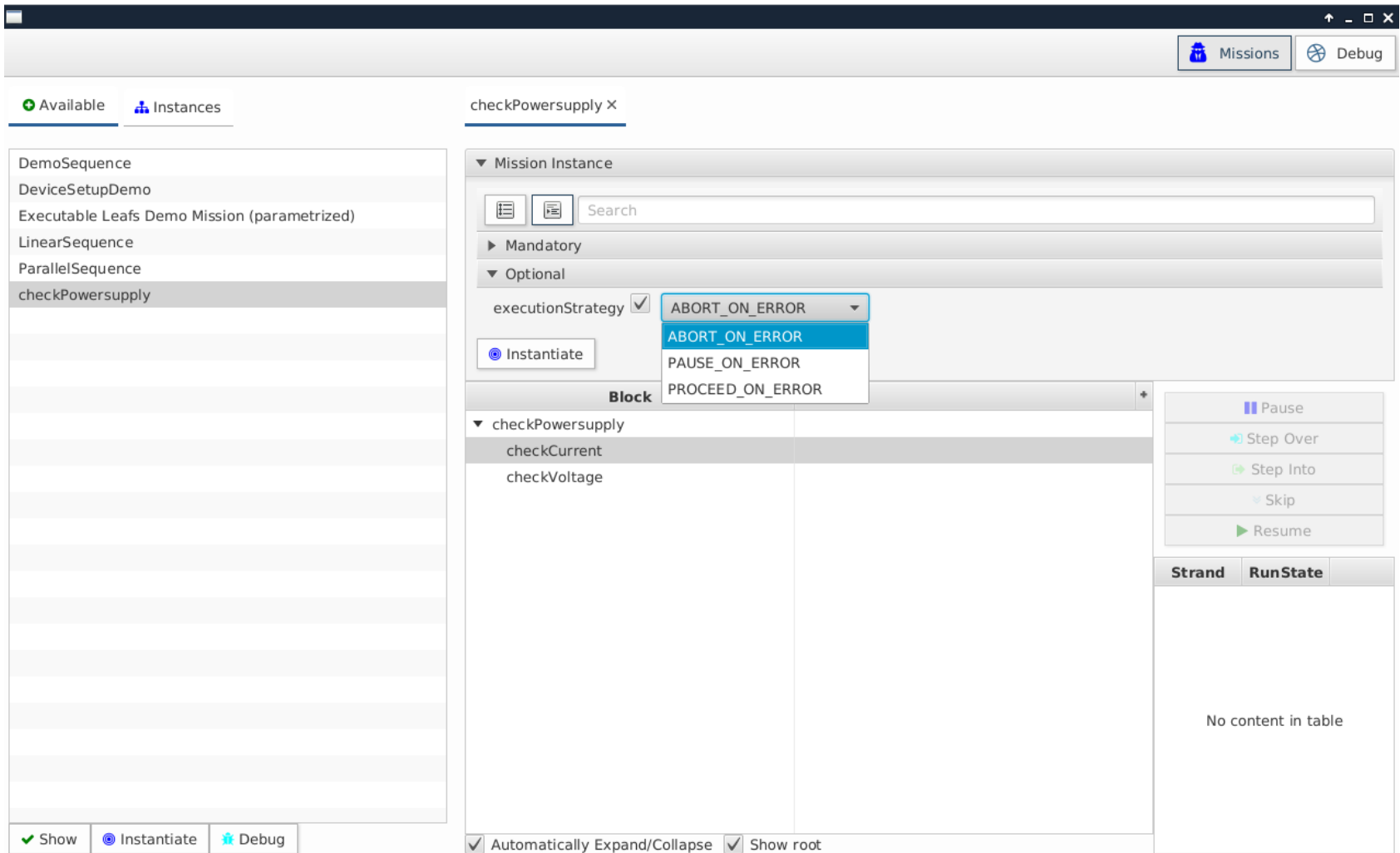
Resume

Strand	RunState
No content in table	

Show Instantiate Debug

Automatically Expand/Collapse Show root

Execution Strategies in Case of Errors

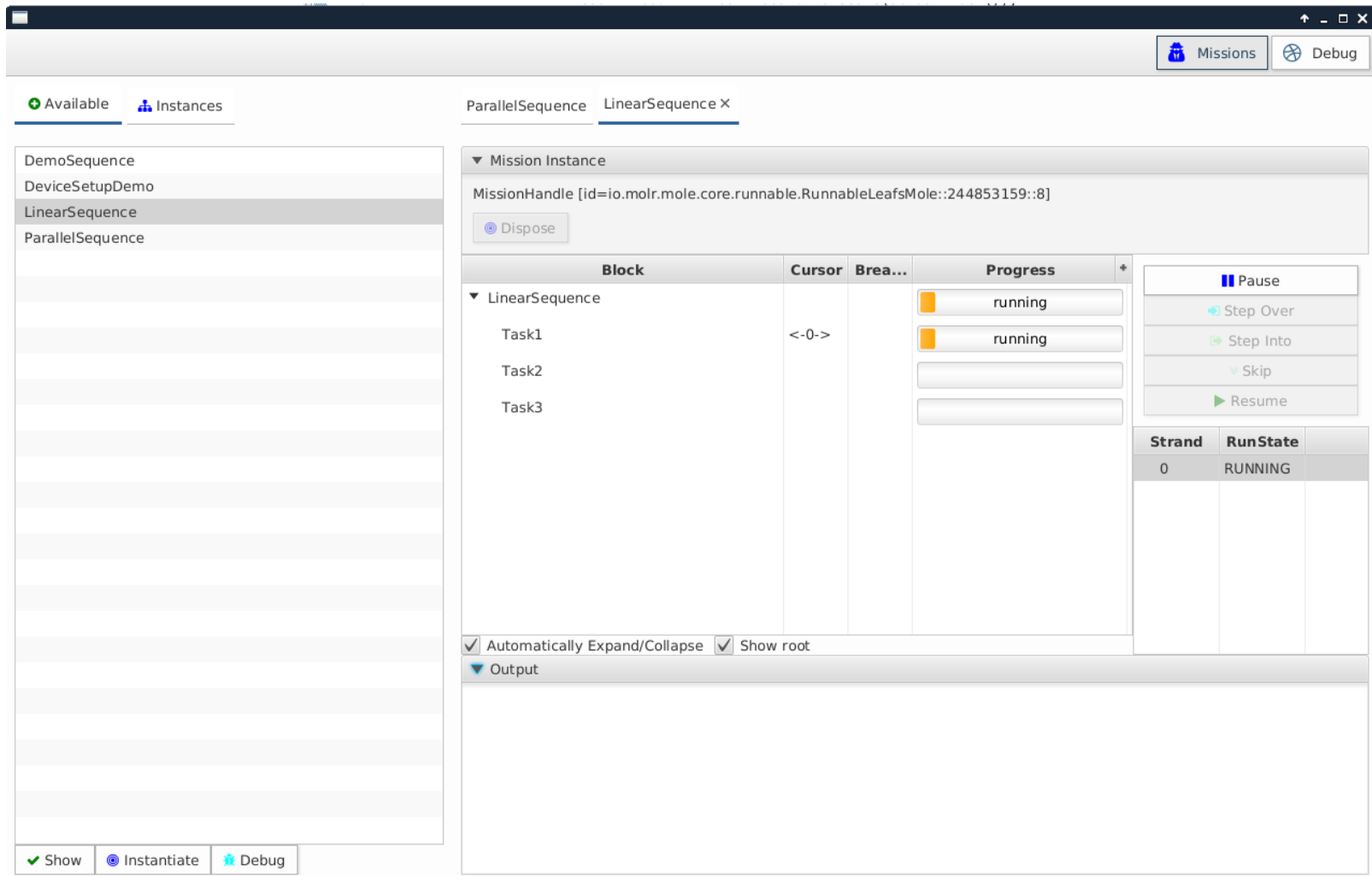


The screenshot shows a software interface for managing mission execution strategies. The main window is titled "checkPowersupply x" and contains several components:

- Left Panel:** A list of mission components including "DemoSequence", "DeviceSetupDemo", "Executable Leafs Demo Mission (parametrized)", "LinearSequence", "ParallelSequence", and "checkPowersupply". The "checkPowersupply" component is selected and highlighted.
- Top Right:** "Missions" and "Debug" buttons.
- Search Bar:** A search input field with a "Search" button.
- Execution Strategy:** A dropdown menu for "executionStrategy" with options: "ABORT_ON_ERROR", "PAUSE_ON_ERROR", and "PROCEED_ON_ERROR". The "ABORT_ON_ERROR" option is currently selected.
- Block List:** A table of blocks under the "checkPowersupply" mission instance:

Block
checkPowersupply
checkCurrent
checkVoltage
- Control Panel:** A set of control buttons: "Pause", "Step Over", "Step Into", "Skip", and "Resume".
- Table:** A table with columns "Strand" and "RunState". The table is currently empty, displaying "No content in table".
- Bottom:** A row of checkboxes: "Show", "Instantiate", "Debug", "Automatically Expand/Collapse", and "Show root".

Linear Execution



The screenshot shows a software interface for mission execution. The top bar includes "Missions" and "Debug" buttons. Below the bar, there are tabs for "Available" and "Instances". The left sidebar lists several mission sequences: DemoSequence, DeviceSetupDemo, LinearSequence (selected), and ParallelSequence. The main area displays a "Mission Instance" for "LinearSequence X" with the ID "MissionHandle [id=io.molr.mole.core.runnable.RunnableLeafsMole::244853159::8]". A "Dispose" button is visible. The central table shows the execution progress of the LinearSequence and its tasks:

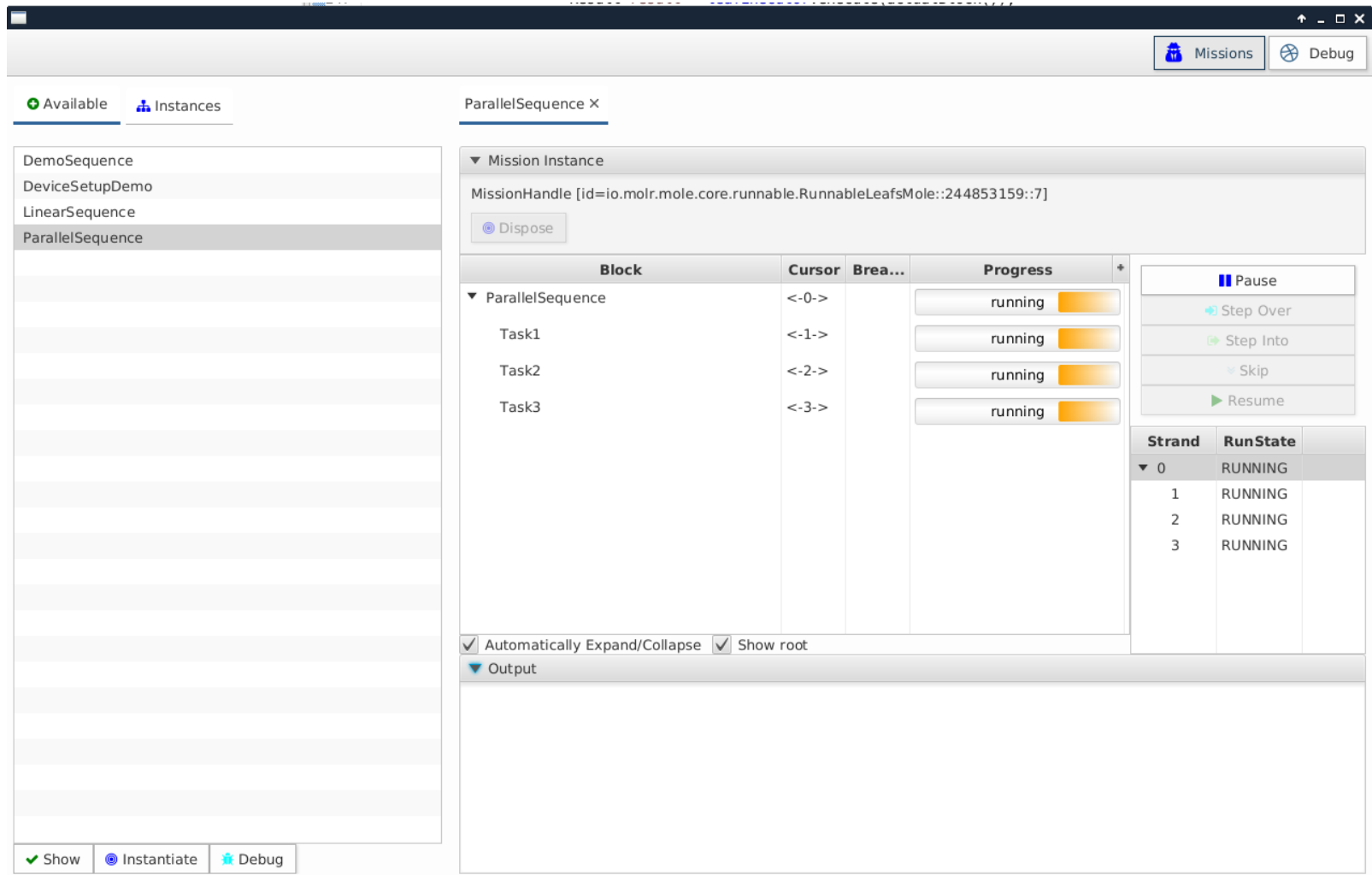
Block	Cursor	Brea...	Progress
LinearSequence			running
Task1	<-0->		running
Task2			
Task3			

On the right, there are control buttons: Pause, Step Over, Step Into, Skip, and Resume. Below these is a table showing the current state of the mission:





Strand	RunState
0	RUNNING

At the bottom, there are checkboxes for "Automatically Expand/Collapse" and "Show root", and an "Output" section. The bottom status bar includes "Show", "Instantiate", and "Debug" buttons.

Parallel Execution



The screenshot displays a software interface for managing parallel execution. On the left, a sidebar lists several sequences: DemoSequence, DeviceSetupDemo, LinearSequence, and ParallelSequence (which is currently selected). The main area is titled 'ParallelSequence X' and shows a 'Mission Instance' for 'MissionHandle [id=io.molr.mole.core.runnable.RunnableLeafsMole::244853159::7]'. Below this, a table lists the execution blocks and their progress:

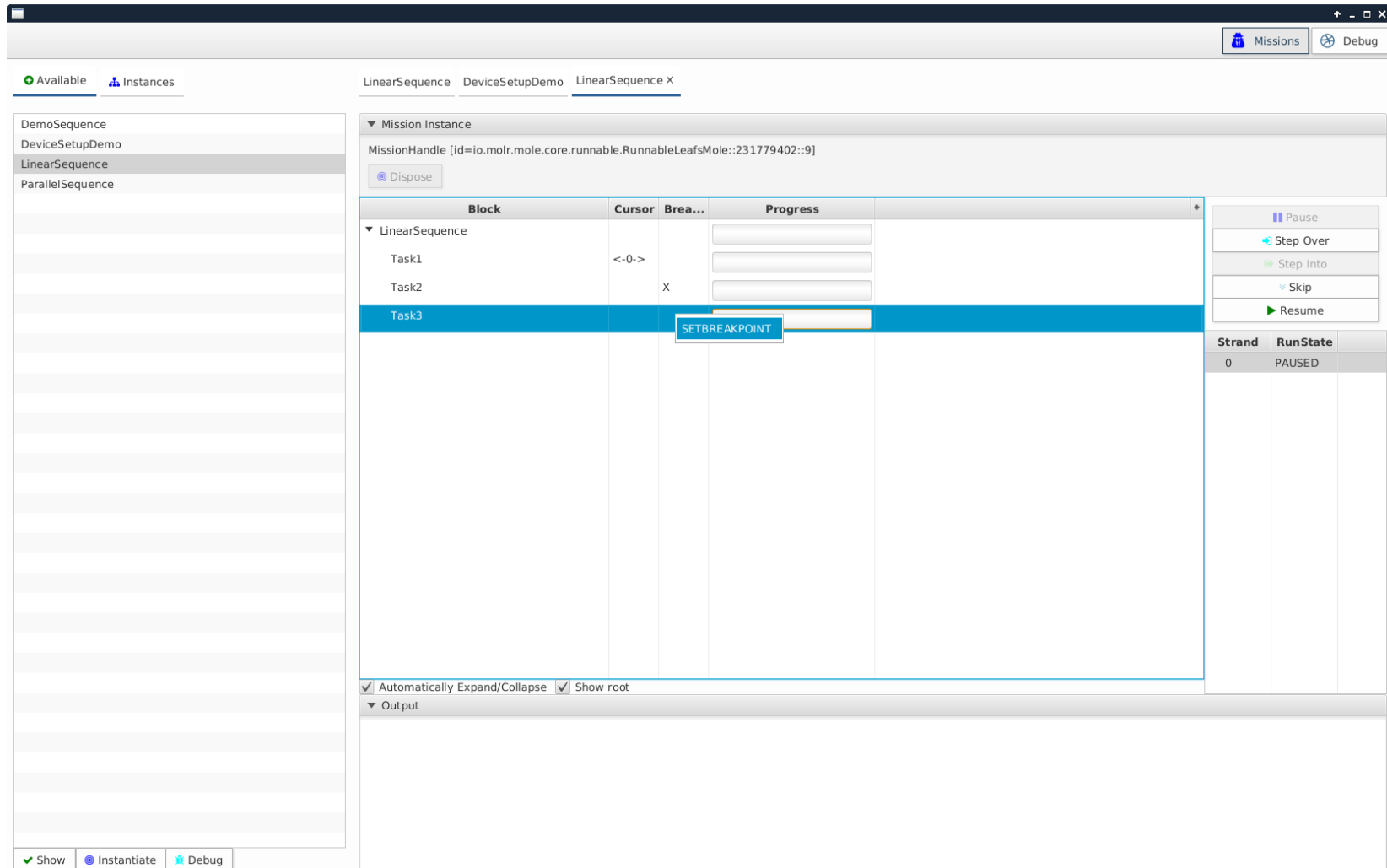
Block	Cursor	Brea...	Progress
ParallelSequence	<-0->		running 
Task1	<-1->		running 
Task2	<-2->		running 
Task3	<-3->		running 

To the right of the table is a control panel with buttons for 'Pause', 'Step Over', 'Step Into', 'Skip', and 'Resume'. Below these buttons is a table showing the state of different strands:

Strand	RunState
0	RUNNING
1	RUNNING
2	RUNNING
3	RUNNING

At the bottom of the main area, there are checkboxes for 'Automatically Expand/Collapse' and 'Show root', and an 'Output' section. At the very bottom of the interface, there are buttons for 'Show', 'Instantiate', and 'Debug'.

Breakpoints and Skipping

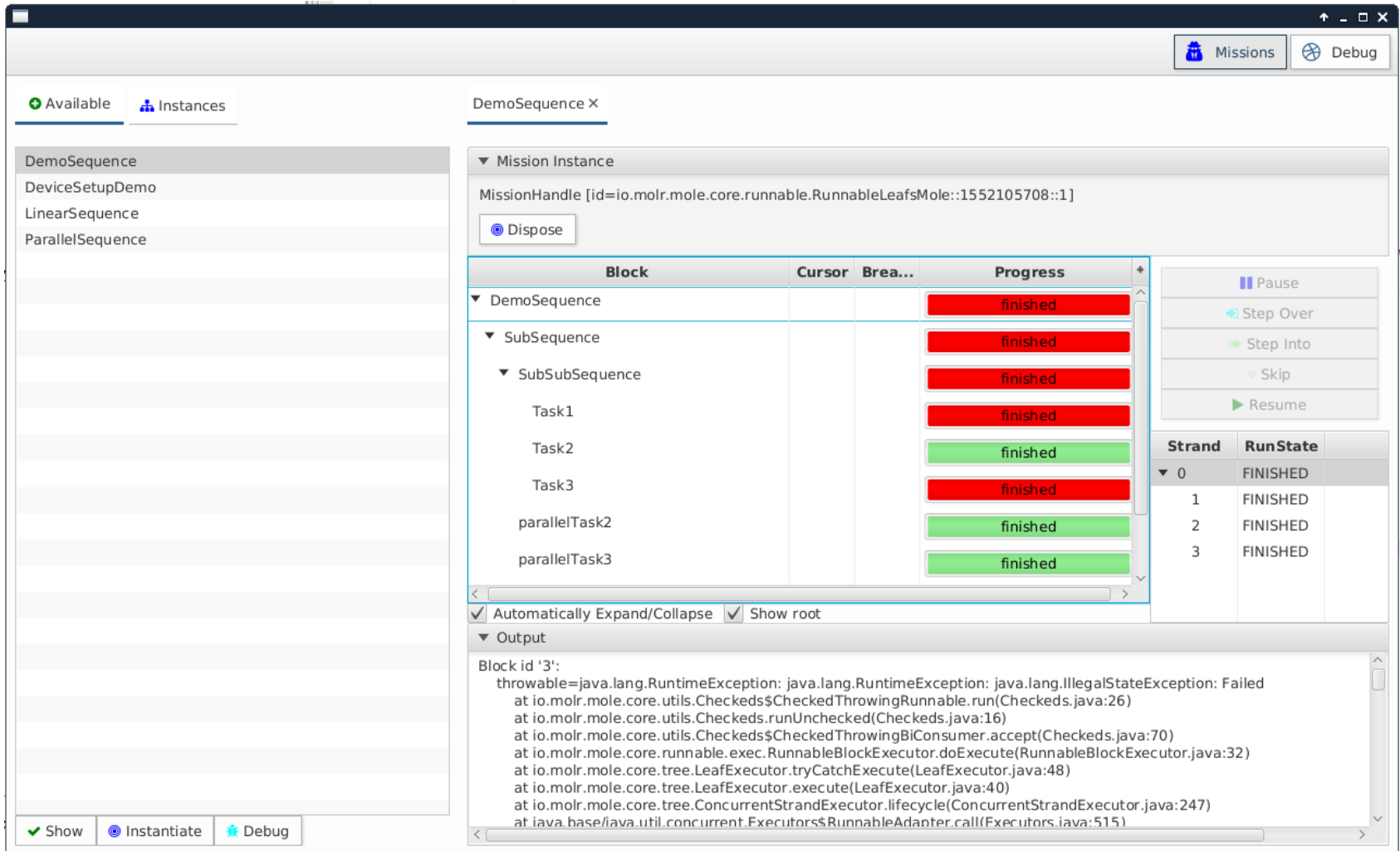


The screenshot shows a mission debugging interface. On the left is a tree view with the following items: DemoSequence, DeviceSetupDemo, LinearSequence (selected), and ParallelSequence. The main area displays a 'Mission Instance' for 'MissionHandle [id=io.moir.mole.core.runnable.RunnableLeafsMole::231779402::9]'. Below this is a table with columns: Block, Cursor, Brea..., and Progress. The table contains three rows: Task1, Task2, and Task3. Task3 is highlighted in blue, and a 'SETBREAKPOINT' button is overlaid on its 'Brea...' column. To the right of the table are control buttons: Pause, Step Over, Step Into, Skip, and Resume. Below the table is a table with columns 'Strand' and 'RunState', showing '0' and 'PAUSED' respectively. At the bottom, there are checkboxes for 'Automatically Expand/Collapse' and 'Show root', and an 'Output' section.

Block	Cursor	Brea...	Progress
Task1	<-0->		
Task2		X	
Task3		SETBREAKPOINT	

Strand	RunState
0	PAUSED

Result Monitoring



The screenshot displays a mission monitoring application window titled "DemoSequence X". The interface includes a top navigation bar with "Missions" and "Debug" buttons. Below this, there are tabs for "Available" and "Instances". The main area shows a tree view of mission blocks:

- DemoSequence
 - DeviceSetupDemo
 - LinearSequence
 - ParallelSequence

The "Mission Instance" section shows a "MissionHandle [id=io.molr.mole.core.runnable.RunnableLeafsMole::1552105708::1]" with a "Dispose" button. A table displays the progress of various blocks:

Block	Cursor	Brea...	Progress
DemoSequence			finished
SubSequence			finished
SubSubSequence			finished
Task1			finished
Task2			finished
Task3			finished
parallelTask2			finished
parallelTask3			finished

On the right side, there are control buttons: "Pause", "Step Over", "Step Into", "Skip", and "Resume". Below these is a table showing the state of different strands:

Strand	RunState
0	FINISHED
1	FINISHED
2	FINISHED
3	FINISHED

At the bottom, there is an "Output" section showing a stack trace for "Block id '3'":

```
Block id '3':
throwable=java.lang.RuntimeException: java.lang.RuntimeException: java.lang.IllegalStateException: Failed
at io.molr.mole.core.utils.Checkeds$CheckedThrowingRunnable.run(Checkeds.java:26)
at io.molr.mole.core.utils.Checkeds.runUnchecked(Checkeds.java:16)
at io.molr.mole.core.utils.Checkeds$CheckedThrowingBiConsumer.accept(Checkeds.java:70)
at io.molr.mole.core.runnable.exec.RunnableBlockExecutor.doExecute(RunnableBlockExecutor.java:32)
at io.molr.mole.core.tree.LeafExecutor.tryCatchExecute(LeafExecutor.java:48)
at io.molr.mole.core.tree.LeafExecutor.execute(LeafExecutor.java:40)
at io.molr.mole.core.tree.ConcurrentStrandExecutor.lifecycle(ConcurrentStrandExecutor.java:247)
at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:515)
```

At the bottom left, there are buttons for "Show", "Instantiate", and "Debug".

- How to configure sequences in a user friendly way
 - Human readable/writable configurations (XML, GUI)
 - Reusability of configurations
- Locks
 - How to ensure that sequence execution operating on devices do not interfere with each other
- Access Control
 - Authorize execution of specified sequences to certain users only