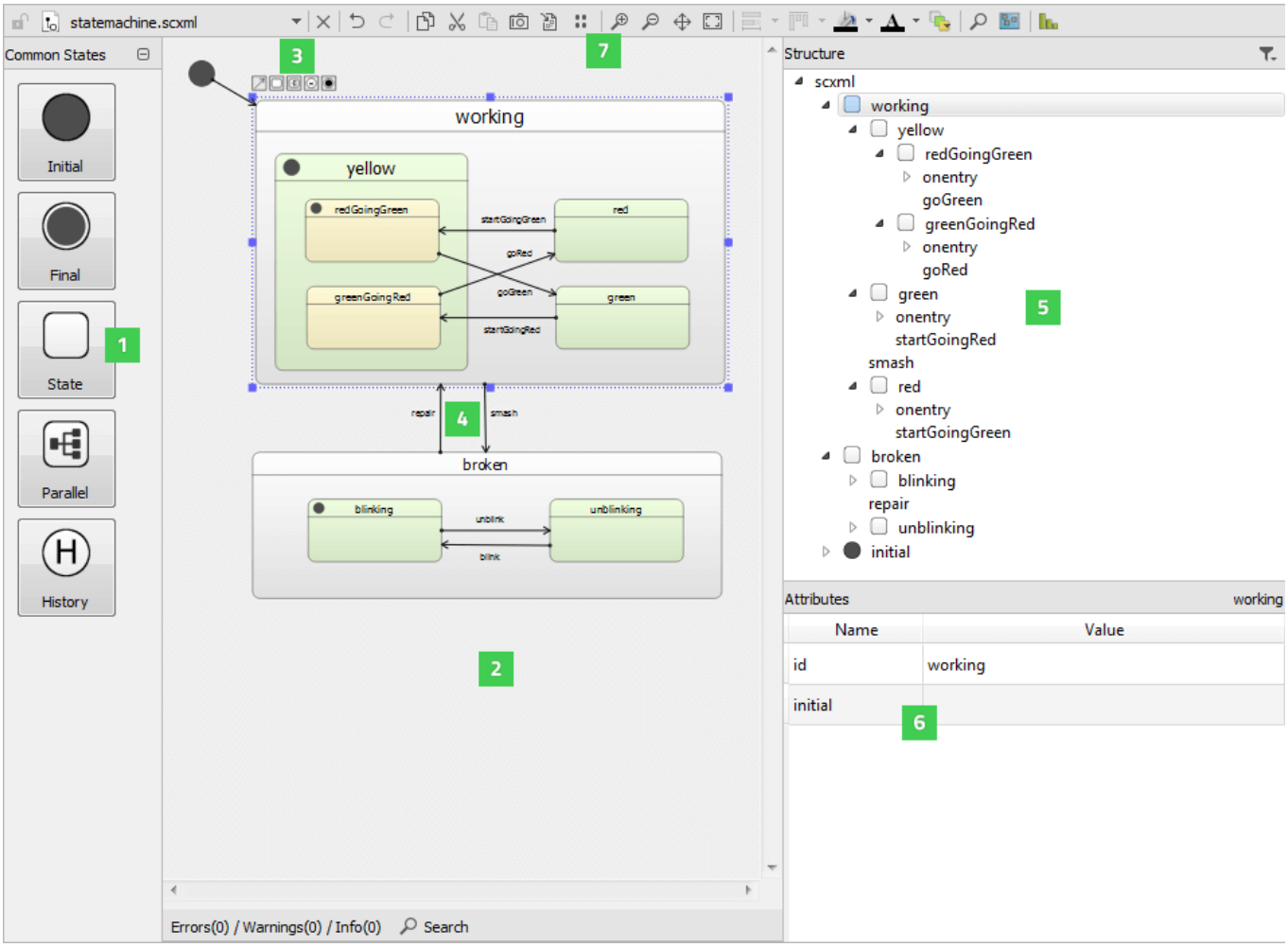
编辑状态图

状态图提供了一种图形方式来模拟系统对刺激的反应。这是通过定义系统可以处于的可能*状态*以及系统如何从一种状态移动到另一种状态（状态之间的转换）来完成*的*。事件驱动系统（如Qt应用程序）的一个关键特征是，行为通常不仅取决于上一个或当前*事件*，还取决于它之前的事件。使用状态图，此信息易于表达。

Qt Creator 提供了一个项目向导，用于将带有样板代码的状态图 XML （SCXML） 文件添加到项目中，以及一个用于编辑状态图的实验性 SCXML 编辑器。您可以使用 SCXML 编辑器向文件添加*状态*和*转换*。然后，您可以使用 Qt SCXML 模块中的类将文件创建的状态机嵌入到 Qt 应用程序中。



可以将状态从“**通用状态**”视图 （1） 拖放到状态编辑器 （2） 中。在状态编辑器中选择一个状态，然后使用工具按钮 （3） 创建过渡 （4） 及其*目标状态*。

您可以在“**结构**”视图 （5） 中查看状态图结构，并在“属性”视图 （6） 中指定所选状态或转换**的属性。**

您可以使用工具栏按钮 （7） 执行编辑、缩放、放大、导航和平移状态图等功能，以及截取屏幕截图和查看统

个状态图一次在状态编辑器中可见，请选择“（**适合视图**）”。

要在状态编辑器中查看大型状态图的特定部分，请选择“（**导航器**）”，然后在要查看的部分上移动导航器框架。

若要使用放大镜放大状态图的一部分，请选择“（**放大镜工具**）”。若要更快地移动放大镜工具，请按 **Alt** 键。

若要平移状态图，请选择“（**平移**）”。要加快平移速度，请按下 **Shift** 键。

若要在状态图中查看有关状态和转换数的统计信息，请选择“（**查看统计信息**）”。

若要从状态图进行搜索，请使用“搜索结果”。搜索将检查整个 SCXML 树中是否存在与搜索条件匹配的属性。

若要将状态图的当前可见部分另存为图像，请选择“（**保存屏幕截图**）”。若要将整个状态图另存为图像，请选择“（**将画布导出到图像**）”。

创建状态图

创建状态图：

1. 选择“**帮助**>**关于插件**>**建模**> **Scxml编辑器**。.
2. 选择**立即重新启动**以重新启动Qt创建器并加载插件。
3. Select **File** > **New File** > **Files and Classes** > **Modeling** > **State Chart** > **Choose** to create an empty state chart and to open it in the SCXML editor.
4. Drag and drop a state from the **Common States** view to the state editor.
5. Drag and drop child states to the initial state to create a *compound state* or use the tool buttons to create a transition from the selected state and its target state.
6. Select a state to edit its attributes in the **Attributes** view.
7. Select the transition line to add edge points to it.
8. To raise or send events, for example, use the context menu commands to add executable content to the and elements of states or to transitions.<onentry><onexit>

The following sections describe how to manage states, transitions, and executable content.

Managing States

When the state machine enters a state in response to an event, the state that it entered becomes the *active state*.

State charts are hierarchical, and therefore states can be nested inside other states, to create compound states.

In addition to basic states, you can create the following types of states:

- *Initial* state is the state the state machine enters when it starts.
- *Parallel state* contains child states that execute in parallel and are all active simultaneously. Events are processed independently by each child state and may trigger different transitions for each child.
- *Final state* enables a state machine to finish. When the state machine enters a top-level final state, it emits

- *History state* is a pseudo-state that represents the child state the parent state was in the last time the parent state was exited.

  Create a history state as a child of the state for which you want to record the current child state. When the state machine detects the presence of such a state at runtime, it automatically records the current (real) child state when the parent state is exited. A transition to the history state is in fact a transition to the child state that the state machine previously saved. The state machine automatically forwards the transition to the real child state.

You can add new states to the state chart in the following ways:

- Drag and drop states from the **Common States** view to the state editor.
- Select a state in the state editor, and then select the **State** tool button to create a transition and its target state.
- Copy and paste states within the SCXML editor or between the SCXML editor and the **Edit** mode.

You can drag states on top of other states to create compound states, or you can drag child states out of their parent state. To move child states within their parent, press down the **Ctrl** key while moving them.

You can use toolbar buttons to align states in the state editor, to adjust their size, and to change the default color scheme. Overlapping states are marked in red color.

To expand or collapse the state tree structure in the **Structure** view, double-click a state.

To view a child state of a nested state in more detail in the state editor, select **Zoom to State**.

To ensure that the state ids are unique within a compound state machine, select (**Toggle Full Namespace**). The name of the parent state is added to the names of the child states, separated by two colons (::). For example:

```
<state id="broken">
...
    <state id="broken::blinking">
    ...
    </state>
    <state id="broken::unblinking">
    ...
    </state>
</state>
```

Managing Transitions

Transitions define how a state reacts to *events* that are generated either by the state machine or external entities. When events occur, the state machine checks for a matching transition defined in the active state and moves to its target state.

To create a transition from the selected state to a new state, drag and release the mouse at the location where you want to add the target state. When you draw a transition to the center of another state, it points to the center of the state, but you can also draw a transition to the edge of the state.

To add edge points to transitions, select a transition line. Only two edge points are permitted for each line, and unnecessary edge points are removed automatically. To remove the selected edge point, select **Remove Point** in the

A transition label is automatically center-aligned, but you can drag it to another position.

To remove the selected transition, select **Remove** in the context menu.

Adding Executable Content

You can add *executable content* to a state chart to enable the state machine to modify its data model and to interact with external entities.

Use the context menu commands to add executable content to the and elements or to transitions:<onentry> <onexit>

- <raise> to raise events
- <send> to communicate with external entities
- <script> to run scripts
- <assign> to modify the data model
- <cancel> to cancel action execution
- <log> to record information in a log
- <if> to execute actions conditionally
- <foreach> to iterate over the items in a collection and execute an action for each of them

During a transition, the state machine executes the content that you specify for the element in the state it is leaving, then the content in the transition, and then the content for the element in the state it is entering. <onexit><onentry>

You can add attributes for the selected executable content in the **Attributes** view.

‹ Modeling

Building and Running ›