Debugging

Debugging is a big part of any system and **SGOAP** wants to make that process as fluid as possible.

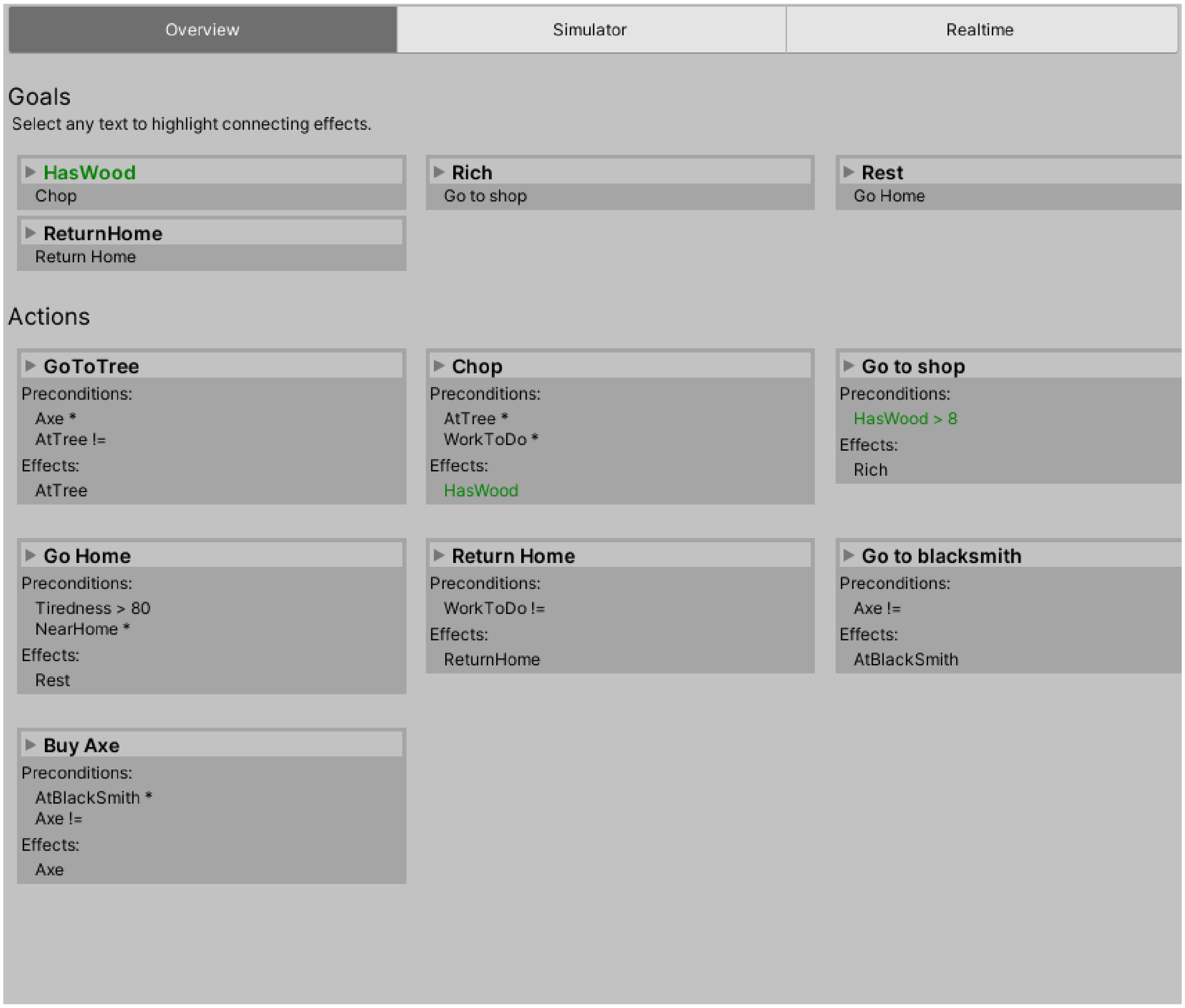
SGOAP ships with a debugger window

▼ Windows > Goap Agent Debugger

Overview Tab

The overview tab shows all your goals and actions. The key thing to do here is see the relationship between each actions.

> 💡 Click on any state to highlight similar state.



Goals

Select any text to highlight connecting effects.

▶ HasWood
Chop

▶ Rich
Go to shop

▶ Rest
Go Home

▶ ReturnHome
Return Home

Actions

▶ GoToTree
Preconditions:
Axe *
AtTree !=
Effects:
AtTree

▶ Chop
Preconditions:
AtTree *
WorkToDo *
Effects:
HasWood

▶ Go to shop
Preconditions:
HasWood > 8
Effects:
Rich

▶ Go Home
Preconditions:
Tiredness > 80
NearHome *
Effects:
Rest

▶ Return Home
Preconditions:
WorkToDo !=
Effects:
ReturnHome

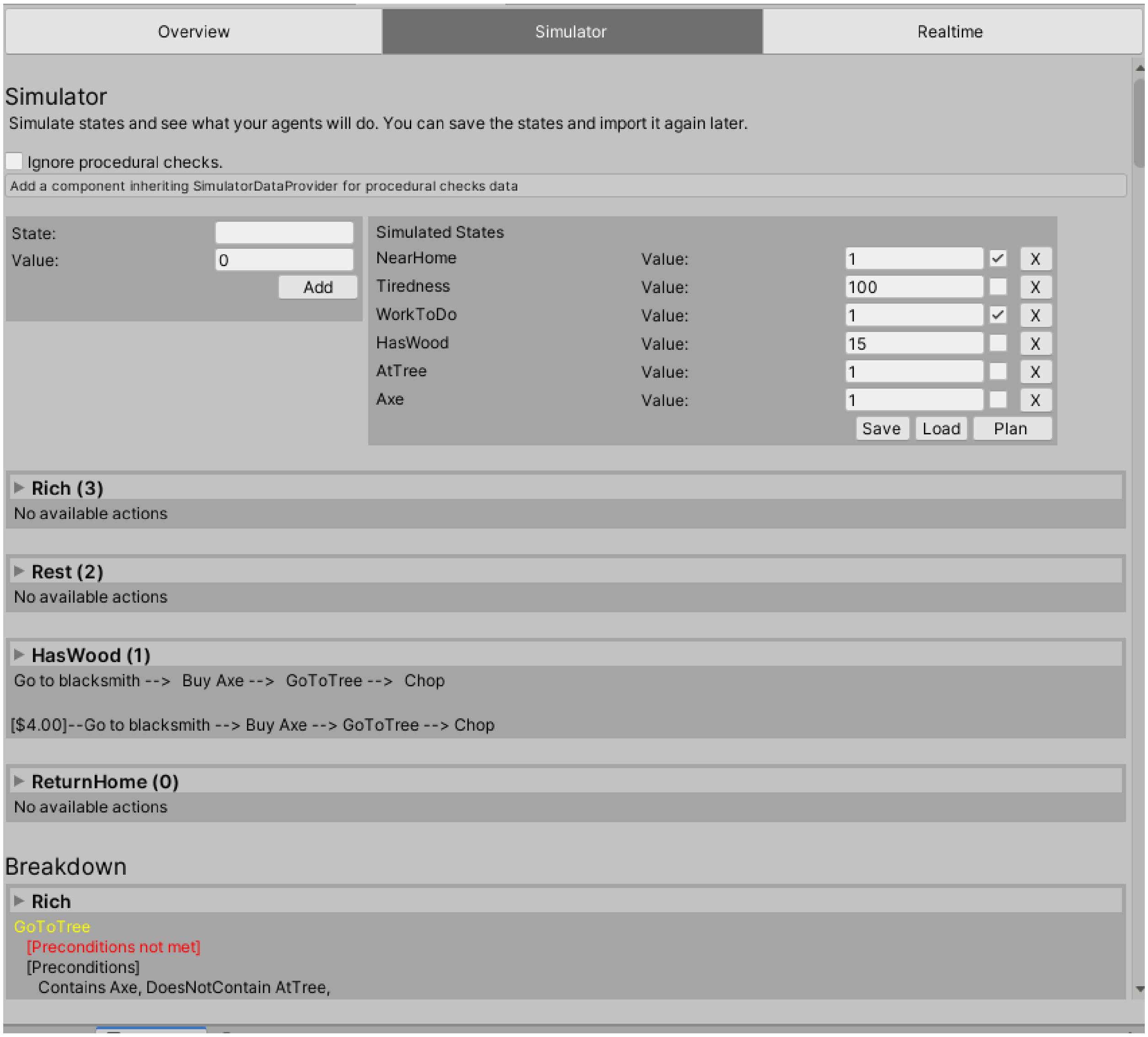
▶ Go to blacksmith
Preconditions:
Axe !=
Effects:
AtBlackSmith

▶ Buy Axe
Preconditions:
AtBlackSmith *
Axe !=
Effects:
Axe

Simulator

Simply press Plan to see what the Agent will do. But wait, how does it know? You can add simulated states, in another word, you can setup states that you want to see what the Agent will do in and press Plan.

💡 You can save and load these states for future use.



Simulator

Simulate states and see what your agents will do. You can save the states and import it again later.

☐ Ignore procedural checks.

Add a component inheriting SimulatorDataProvider for procedural checks data

State:

Value: 0

Simulated States

NearHome	Value:	1	☑
Tiredness	Value:	100	☐
WorkToDo	Value:	1	☑
HasWood	Value:	15	☐
AtTree	Value:	1	☐
Axe	Value:	1	☐

▶ **Rich (3)**

No available actions

▶ **Rest (2)**

No available actions

▶ **HasWood (1)**

Go to blacksmith --> Buy Axe --> GoToTree --> Chop

[$4.00]--Go to blacksmith --> Buy Axe --> GoToTree --> Chop

▶ **ReturnHome (0)**

No available actions

Breakdown

▶ **Rich**

GoToTree

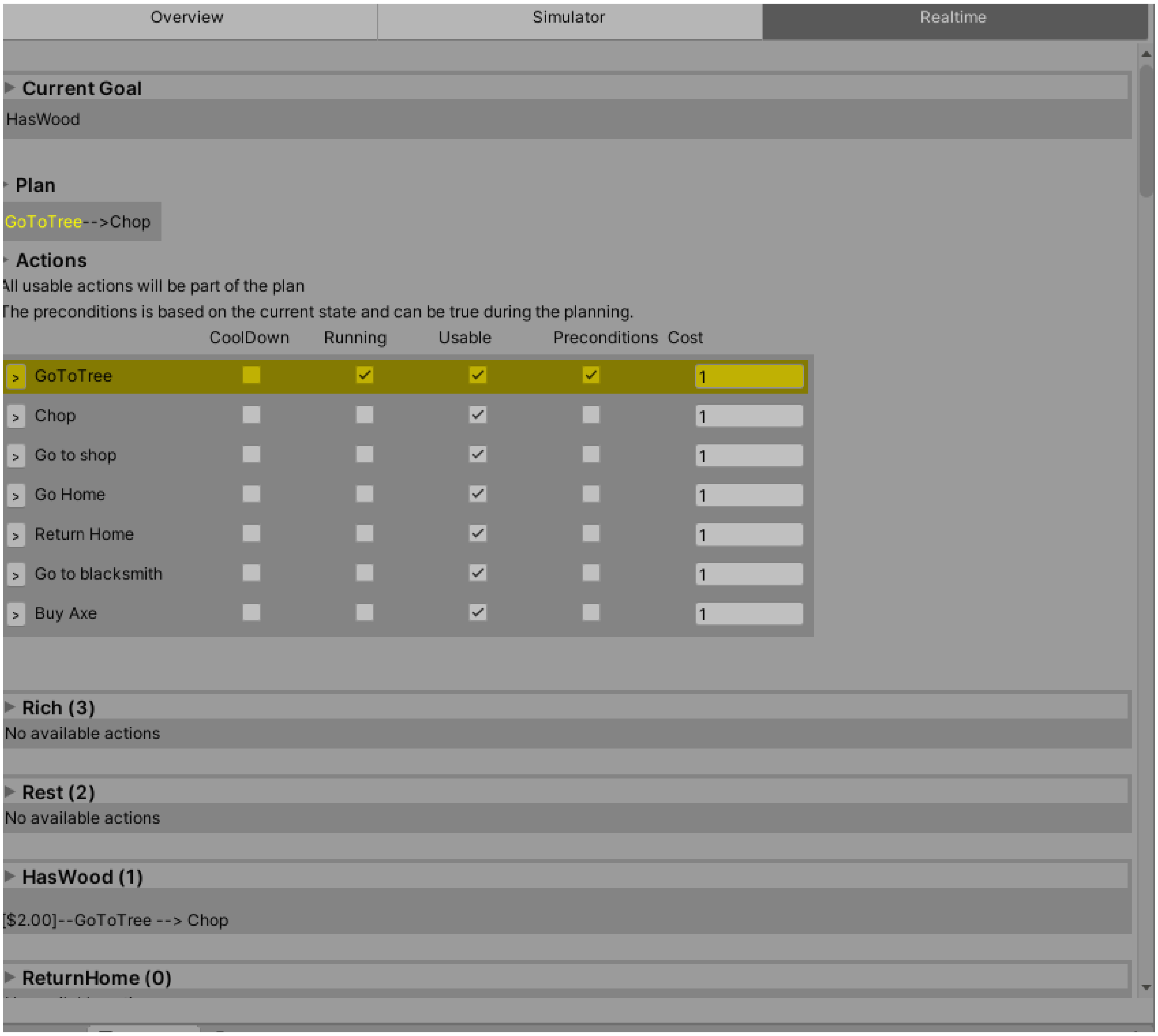
[Preconditions not met]

[Preconditions]

Contains Axe, DoesNotContain AtTree,

Realtime Tab

The Realtime Tab is a place to see what your Agent is thinking in real time! There are other handy usages such as modifying an action and stopping it. Changing the cost and see what will happen.



Current Goal

HasWood

Plan

GoToTree-->Chop

Actions

All usable actions will be part of the plan

The preconditions is based on the current state and can be true during the planning.

	CoolDown	Running	Usable	Preconditions	Cost
> GoToTree	☐	☑	☑	☑	1
> Chop	☐	☐	☑	☐	1
> Go to shop	☐	☐	☑	☐	1
> Go Home	☐	☐	☑	☐	1
> Return Home	☐	☐	☑	☐	1
> Go to blacksmith	☐	☐	☑	☐	1
> Buy Axe	☐	☐	☑	☐	1

Rich (3)

No available actions

Rest (2)

No available actions

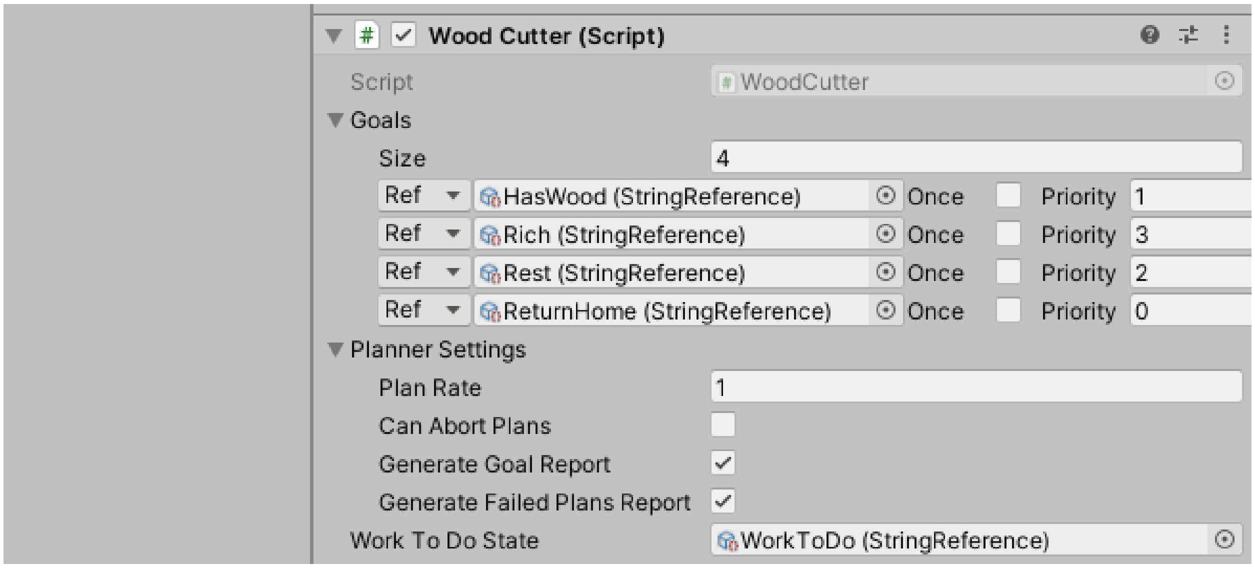
HasWood (1)

[$2.00]--GoToTree --> Chop

ReturnHome (0)

Extra Information

By default the full breakdown report is not enabled due to performance issues, but in the case you need a lot more information, go to your Agent, expand Planner Settings and check GenerateFailedPlansReport.



☑ Wood Cutter (Script)

Script: WoodCutter

Goals

Size: 4

Ref	HasWood (StringReference)	Once	□	Priority	1
Ref	Rich (StringReference)	Once	□	Priority	3
Ref	Rest (StringReference)	Once	□	Priority	2
Ref	ReturnHome (StringReference)	Once	□	Priority	0

Planner Settings

Plan Rate: 1

Can Abort Plans: □

Generate Goal Report: ☑

Generate Failed Plans Report: ☑

Work To Do State: WorkToDo (StringReference)