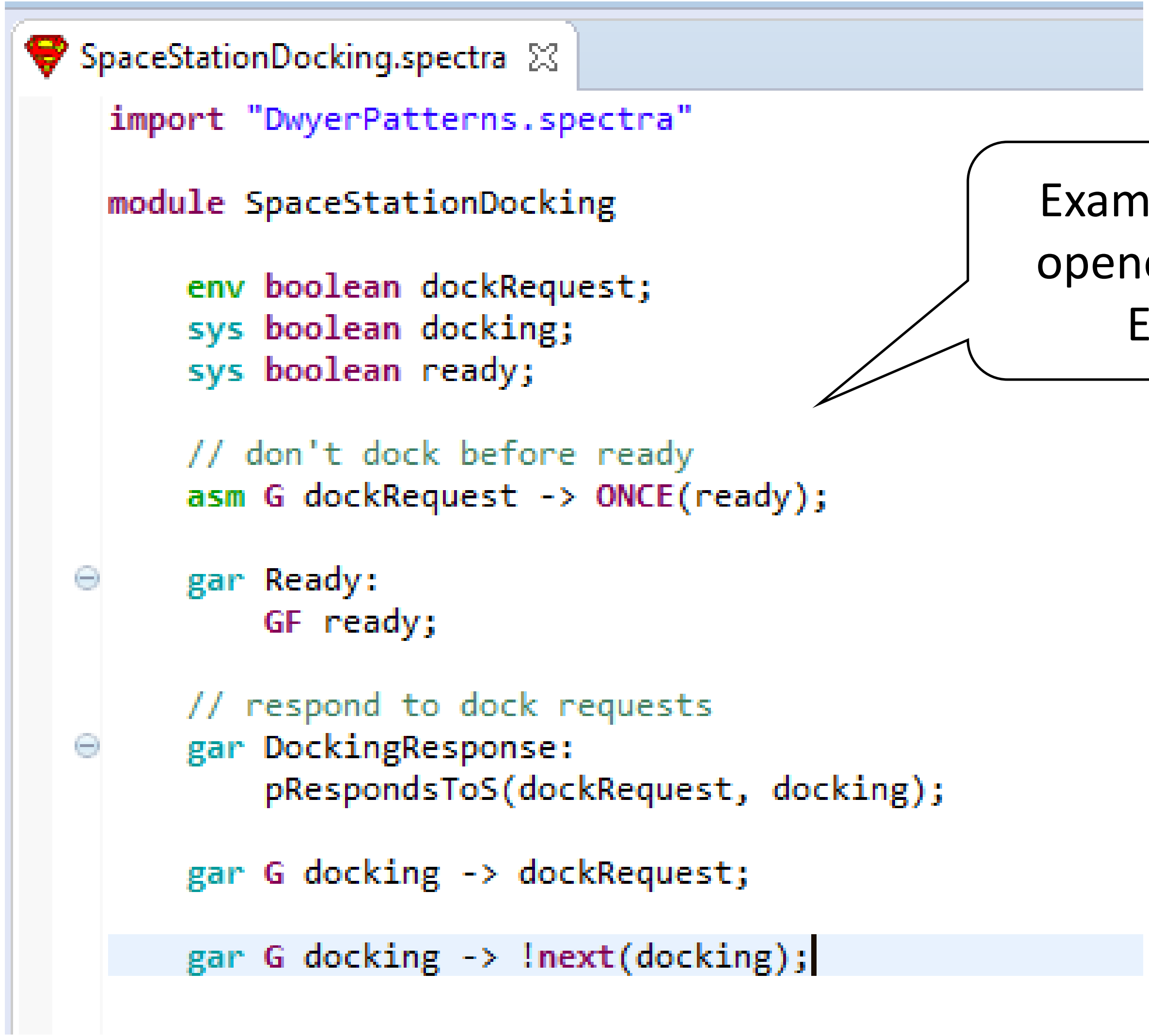
A Symbolic Justice Violations Transition System for Unrealizable GR(1) Specifications

JVTS Tool Session Example

Aviv Kuvent, Shahar Maoz and Jan Oliver Ringert

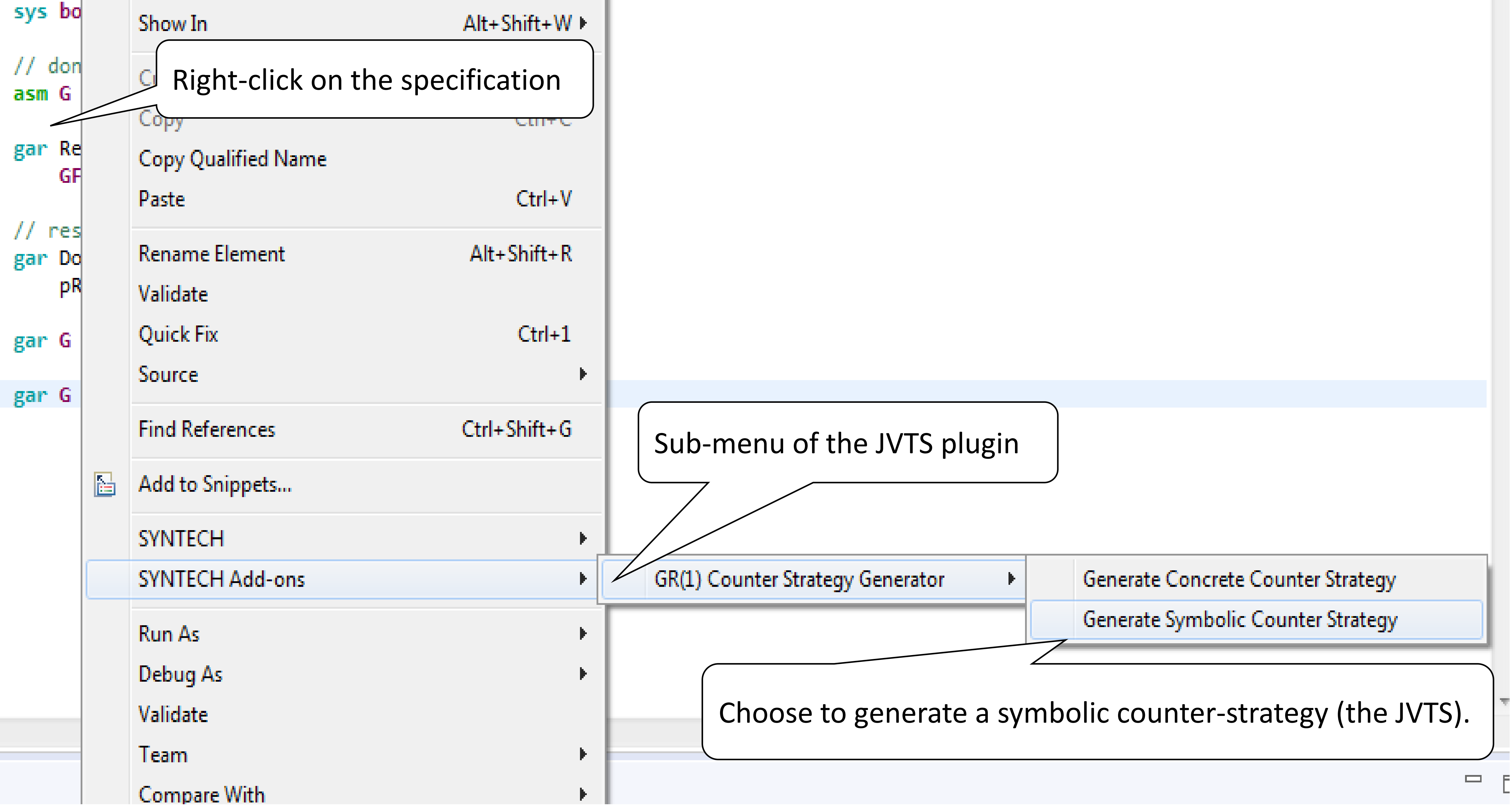
Running Example Specification



```
import "DwyerPatterns.spectra"

module SpaceStationDocking

    env boolean dockRequest;
    sys boolean docking;
    sys boolean ready;

    // don't dock before ready
    asm G dockRequest -> ONCE(ready);

    gar Ready:
        GF ready;

    // respond to dock requests
    gar DockingResponse:
        pRespondsToS(dockRequest, docking);

    gar G docking -> dockRequest;

    gar G docking -> !next(docking);
```

Example Specification opened in the Spectra Eclipse editor

Generate Symbolic Counter Strategy

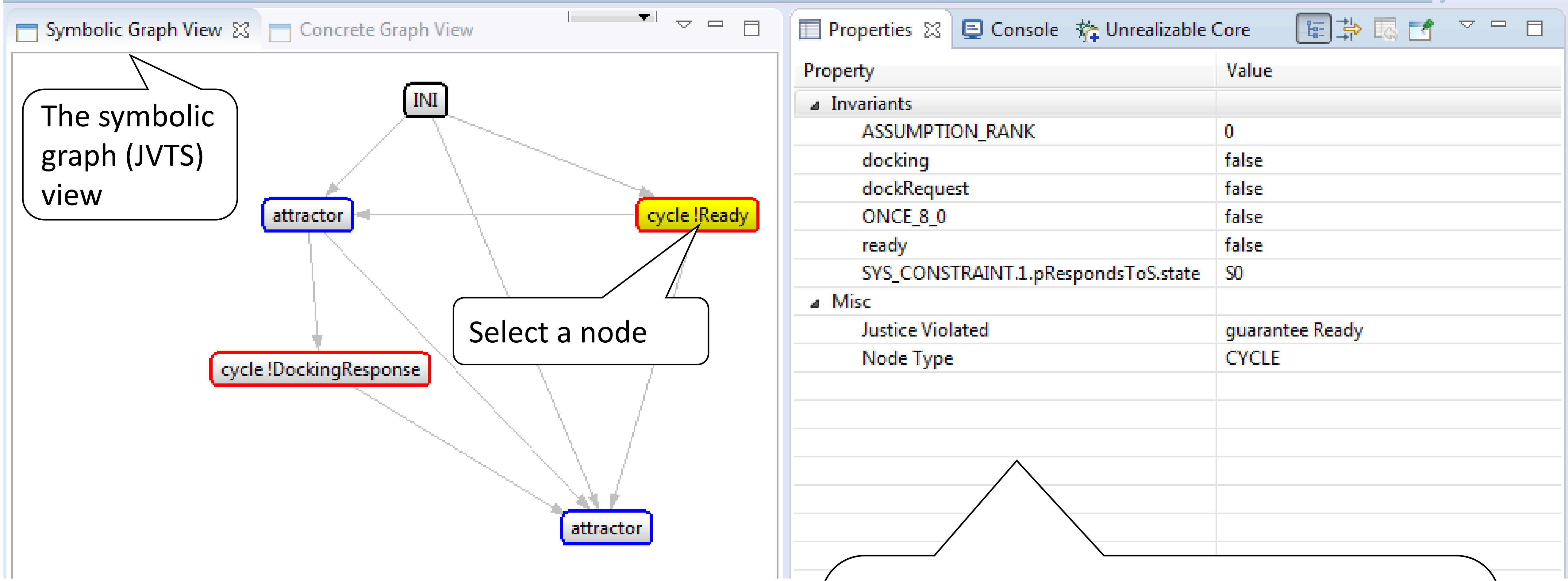


```
sys bo

// don
asm G

gar Re
    GF

// res
gar Do
    pR

gar G

gar G
```

Right-click on the specification

Sub-menu of the JVTS plugin

Choose to generate a symbolic counter-strategy (the JVTS).



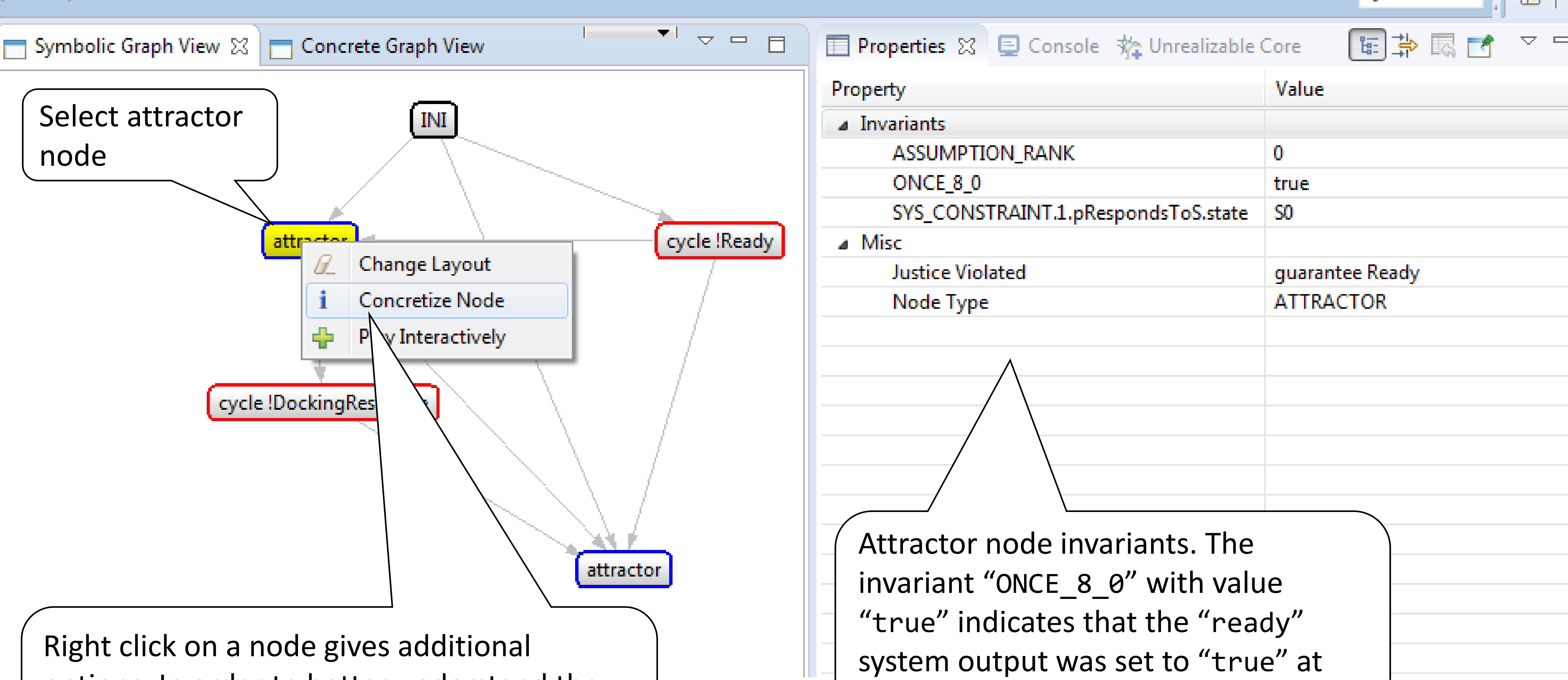
Property	Value
Invariants	
ASSUMPTION_RANK	0
docking	false
dockRequest	false
ONCE_8_0	false
ready	false
SYS_CONSTRAINT.1.pRespondsToS.state	S0
Misc	
Justice Violated	guarantee Ready
Node Type	CYCLE

The symbolic graph (JVTS) view

Select a node

Properties of the selected state, including invariants of the node.
This is a cycle node where the system violates its justice guarantee `Ready: GF ready` by maintaining `ready = false`

Concretizing Nodes

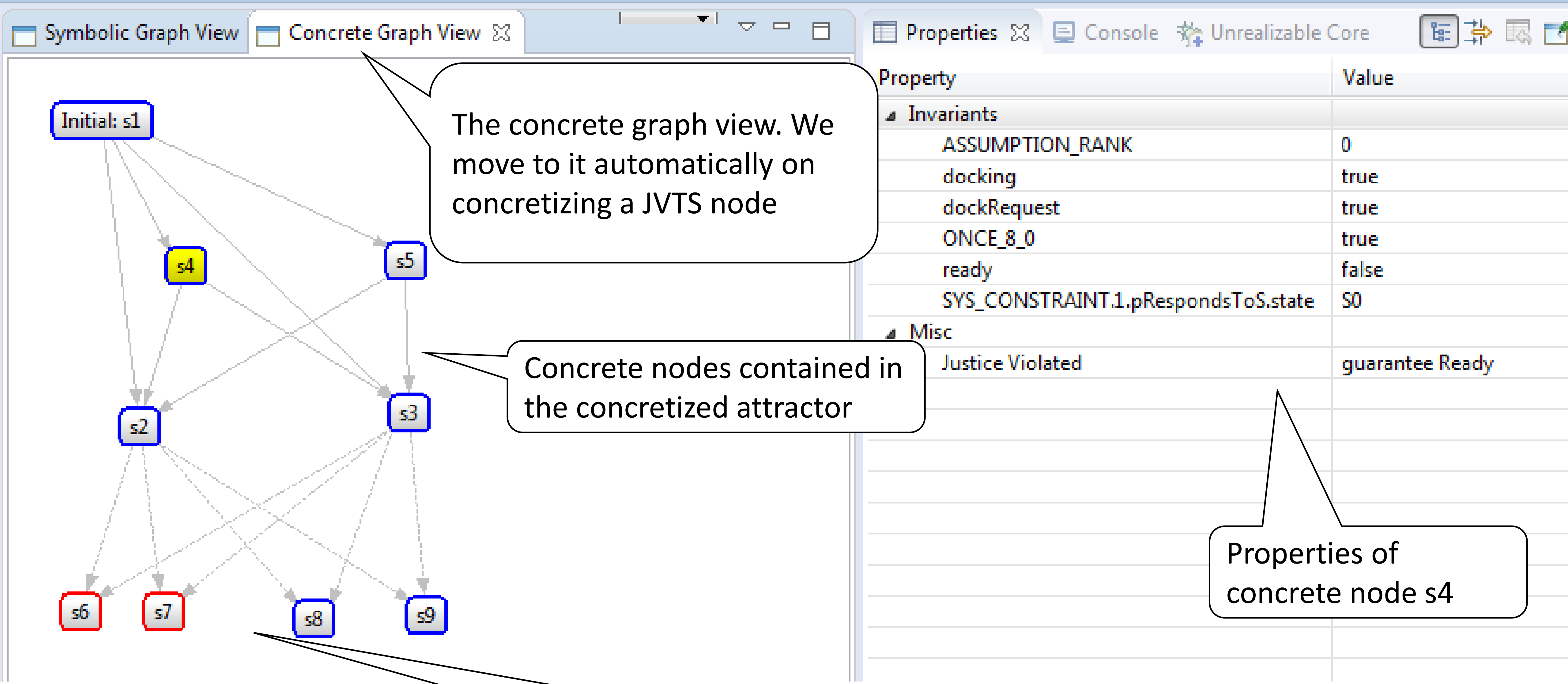


Property	Value
Invariants	
ASSUMPTION_RANK	0
ONCE_8_0	true
SYS_CONSTRAINT.1.pRespondsToS.state	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR

Select attractor node

Right click on a node gives additional options. In order to better understand the flow attracting to the lower cycle, we choose to concretize this node

Attractor node invariants. The invariant “ONCE_8_0” with value “true” indicates that the “ready” system output was set to “true” at some point in the past



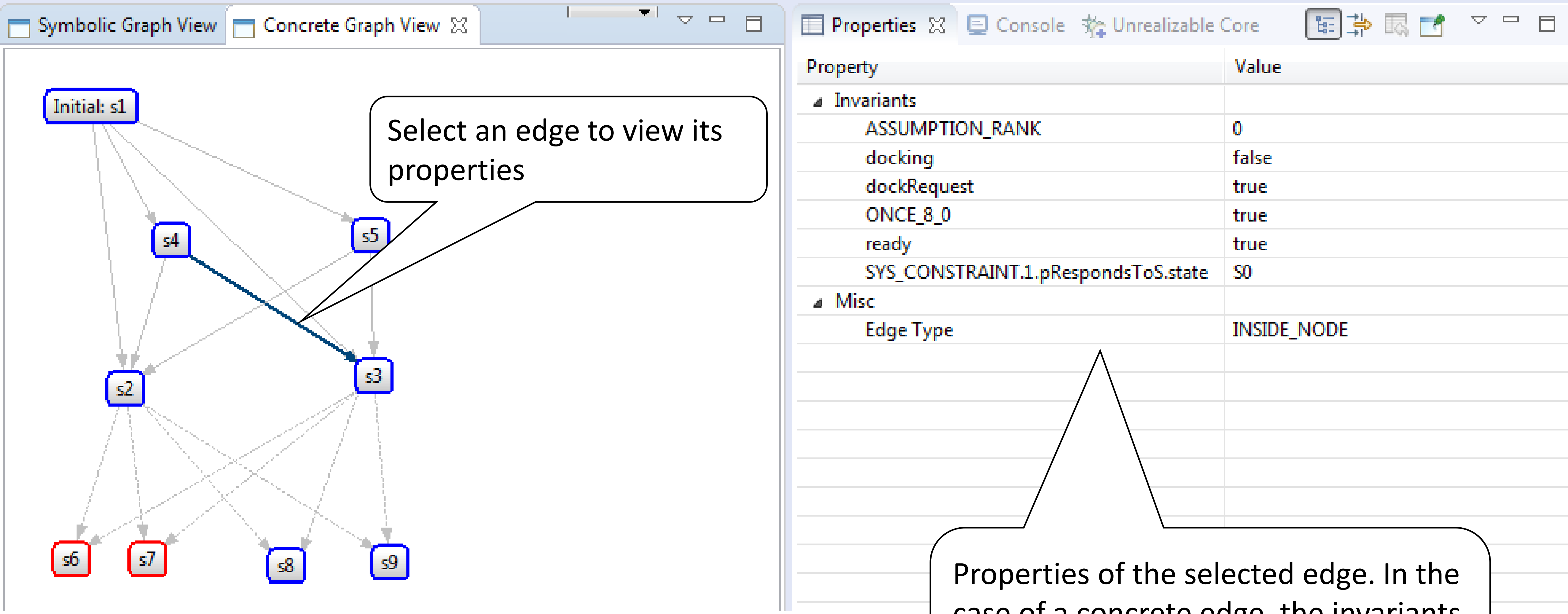
Property	Value
Invariants	
ASSUMPTION_RANK	0
docking	true
dockRequest	true
ONCE_8_0	true
ready	false
SYS_CONSTRAINT.1.pRespondsToS.state	S0
Misc	
Justice Violated	guarantee Ready

The concrete graph view. We move to it automatically on concretizing a JVTS node

Concrete nodes contained in the concretized attractor

Properties of concrete node s4

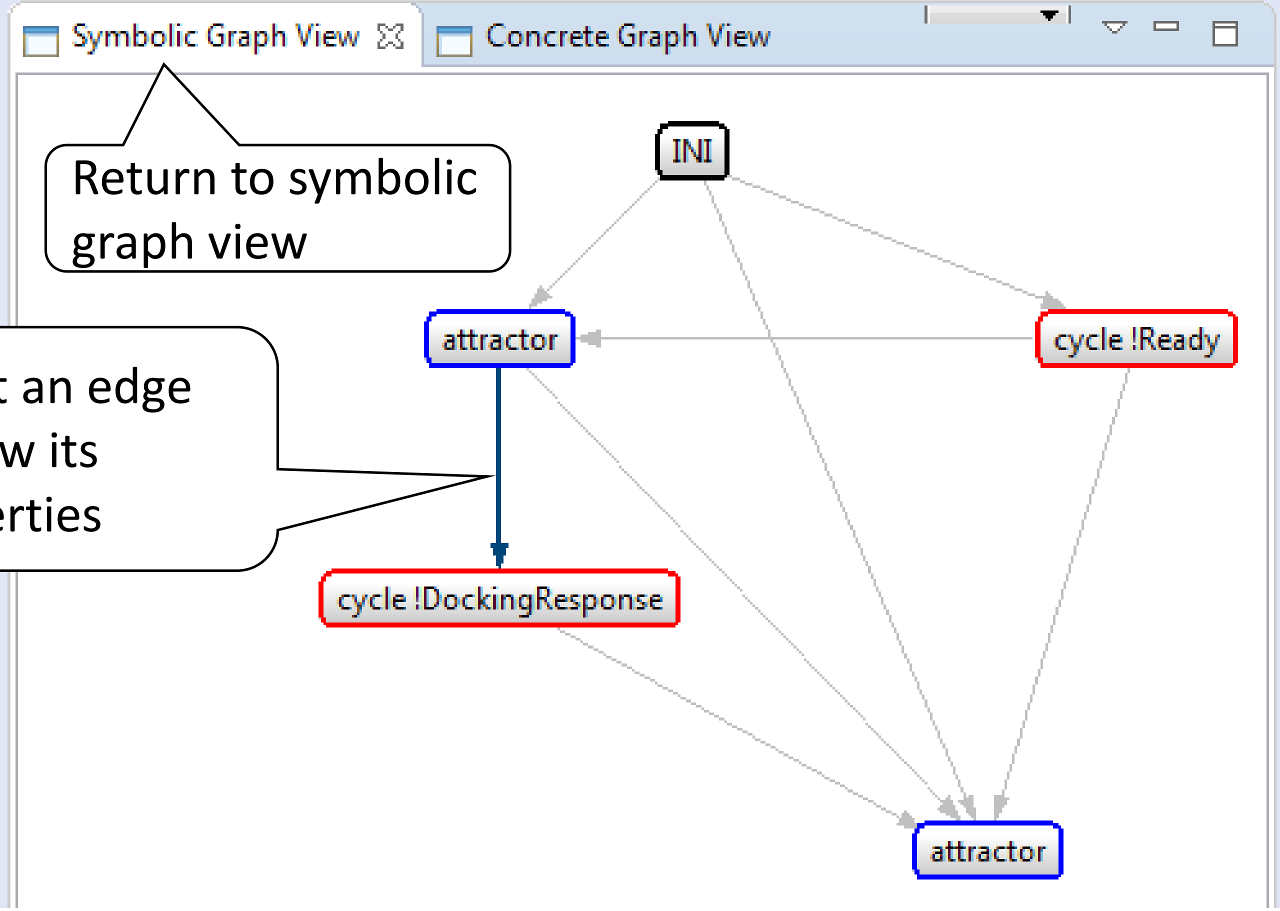
Bottom row are nodes reachable in other JVTS nodes, identified by the dotted edges leading to them



Property	Value
Invariants	
ASSUMPTION_RANK	0
docking	false
dockRequest	true
ONCE_8_0	true
ready	true
SYS_CONSTRAINT.1.pRespondsToS.state	S0
Misc	
Edge Type	INSIDE_NODE

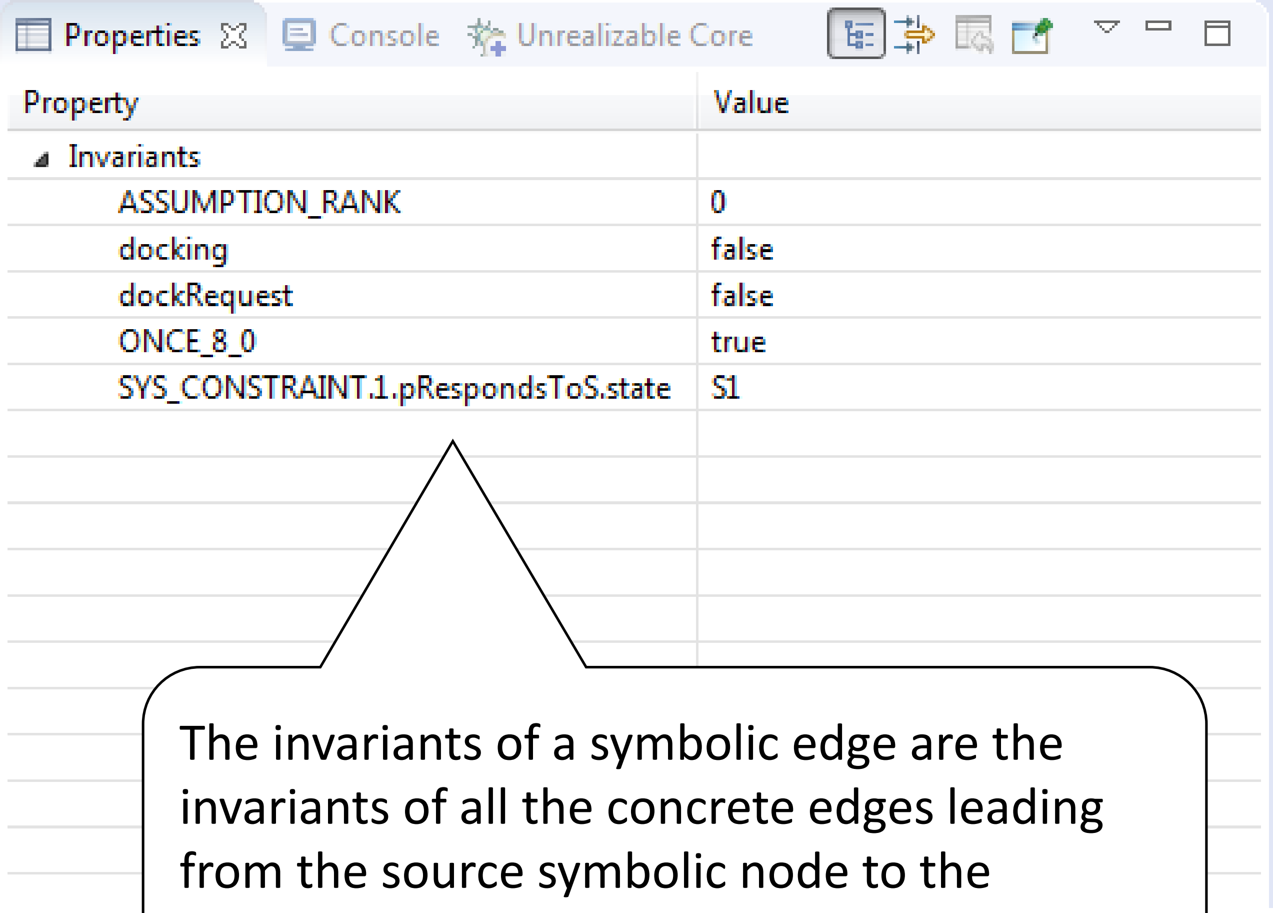
Select an edge to view its properties

Properties of the selected edge. In the case of a concrete edge, the invariants on it will be the invariants of the destination node



Return to symbolic graph view

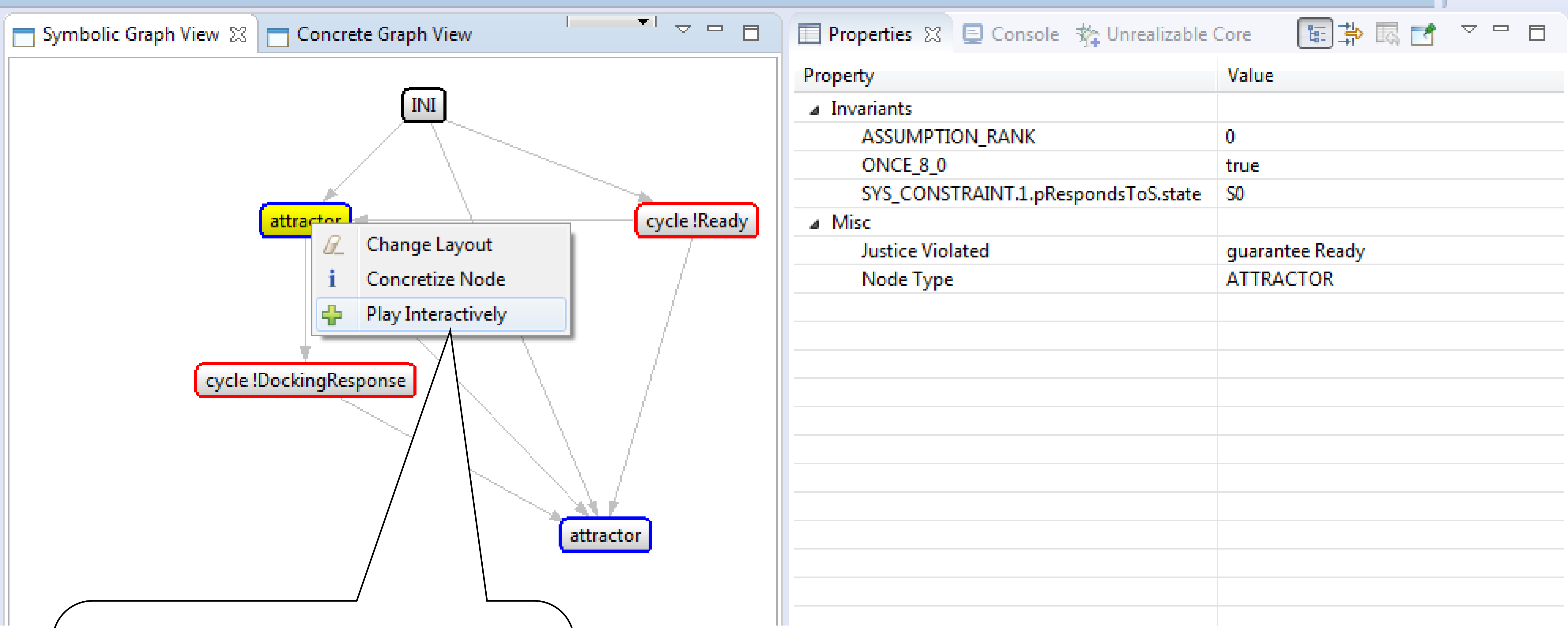
t an edge
w its
erties



Property	Value
Invariants	
ASSUMPTION_RANK	0
docking	false
dockRequest	false
ONCE_8_0	true
SYS_CONSTRAINT.1.pRespondsToS.state	S1

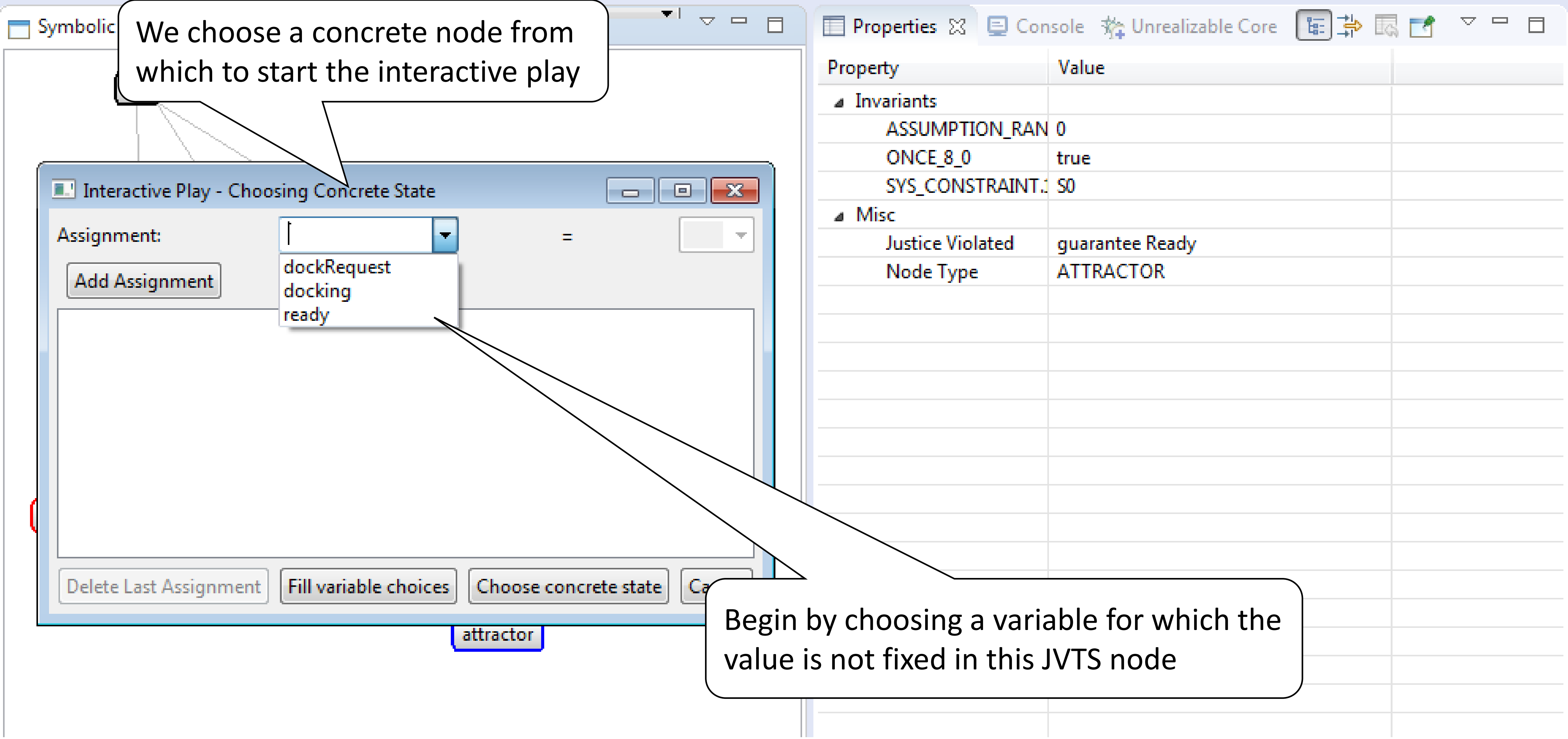
The invariants of a symbolic edge are the invariants of all the concrete edges leading from the source symbolic node to the destination symbolic node

Playing Interactively



Property	Value
Invariants	
ASSUMPTION_RANK	0
ONCE_8_0	true
SYS_CONSTRAINT.1.pRespondsToS.state	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR

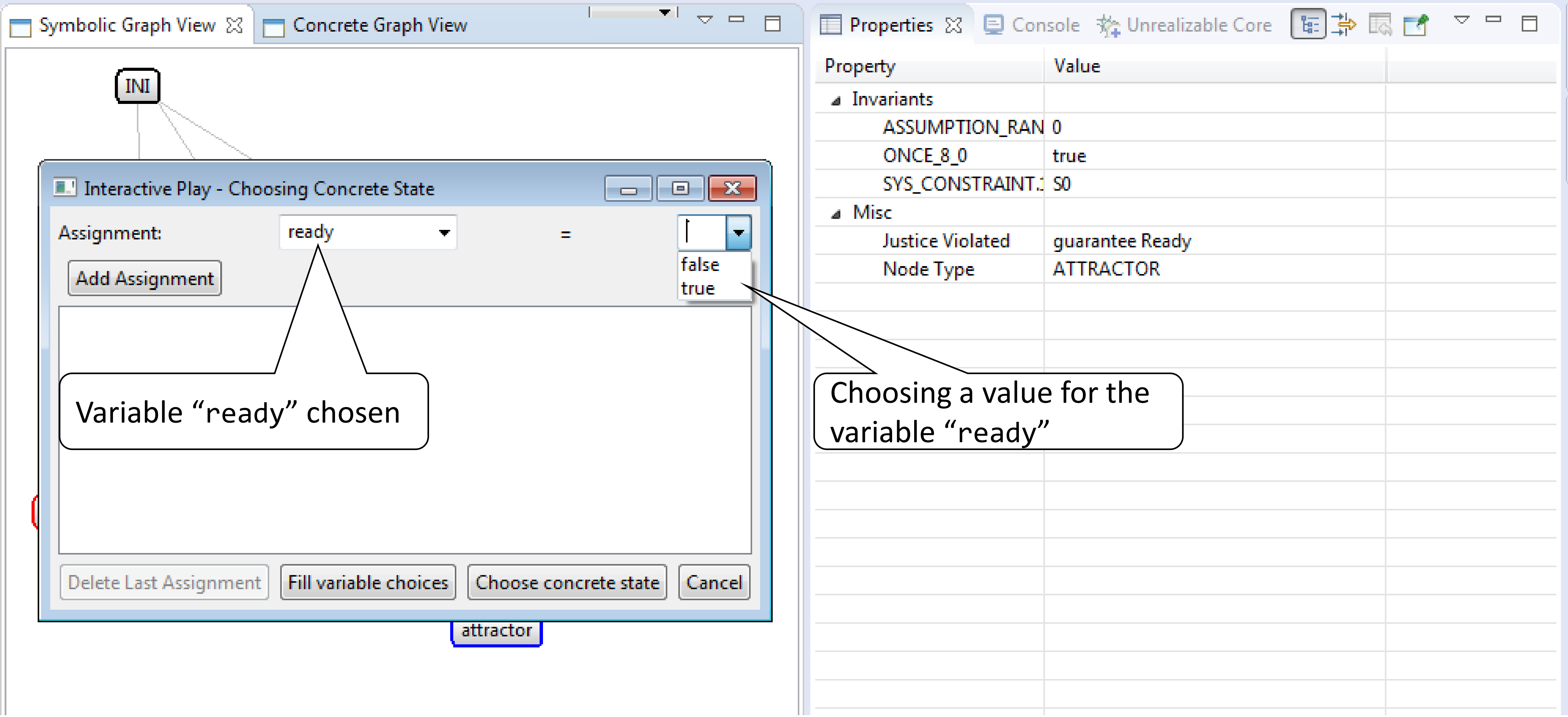
Right click on the attractor. Select to perform interactive play to better understand the flow of the counter-strategy represented by this JVTS



We choose a concrete node from which to start the interactive play

Begin by choosing a variable for which the value is not fixed in this JVTS node

Property	Value
Invariants	
ASSUMPTION_RAN	0
ONCE_8_0	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR



Interactive Play - Choosing Concrete State

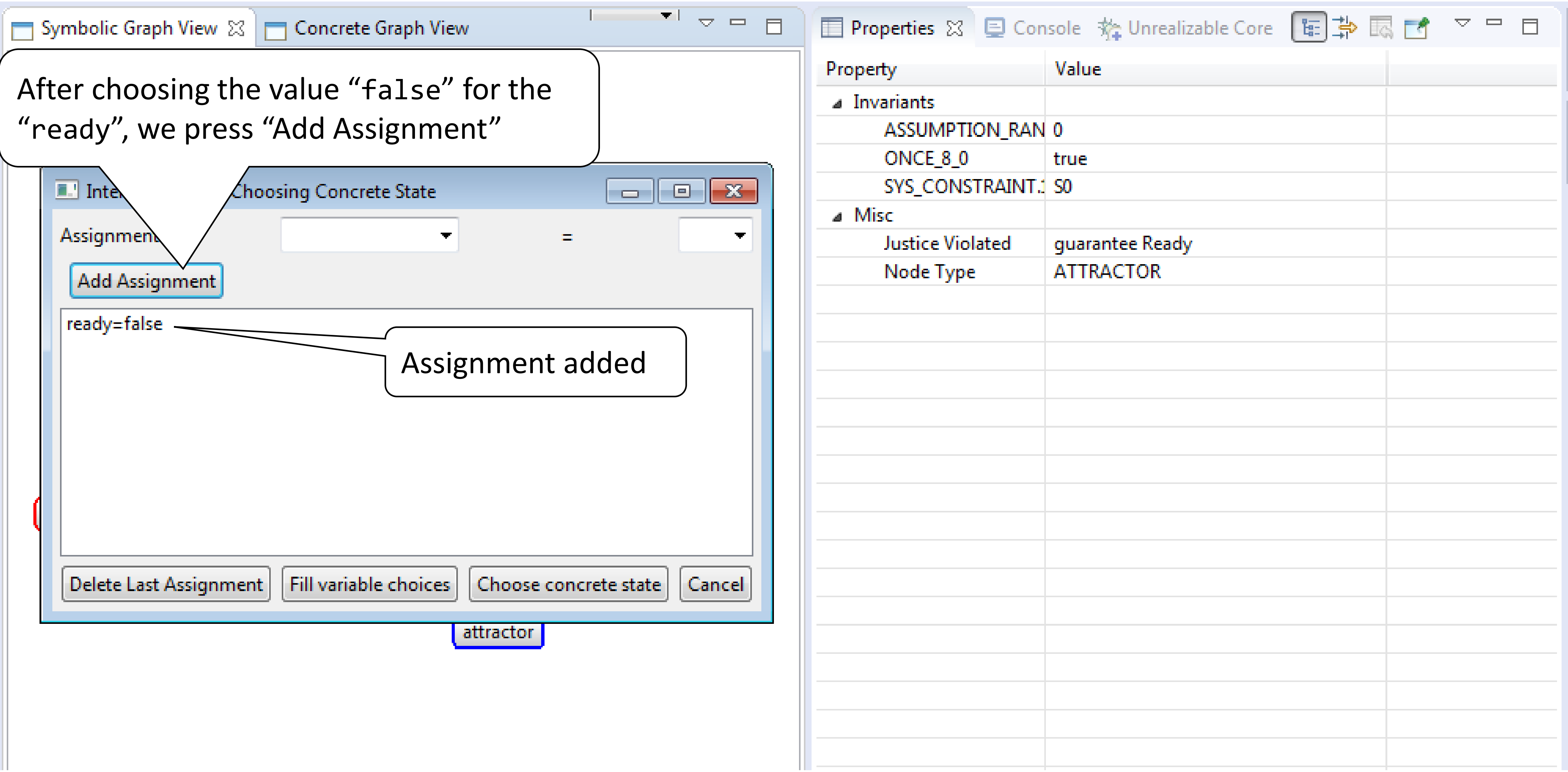
Assignment: ready =

false
true

Property	Value
Invariants	
ASSUMPTION_RAN	0
ONCE_8_0	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR

Variable “ready” chosen

Choosing a value for the variable “ready”

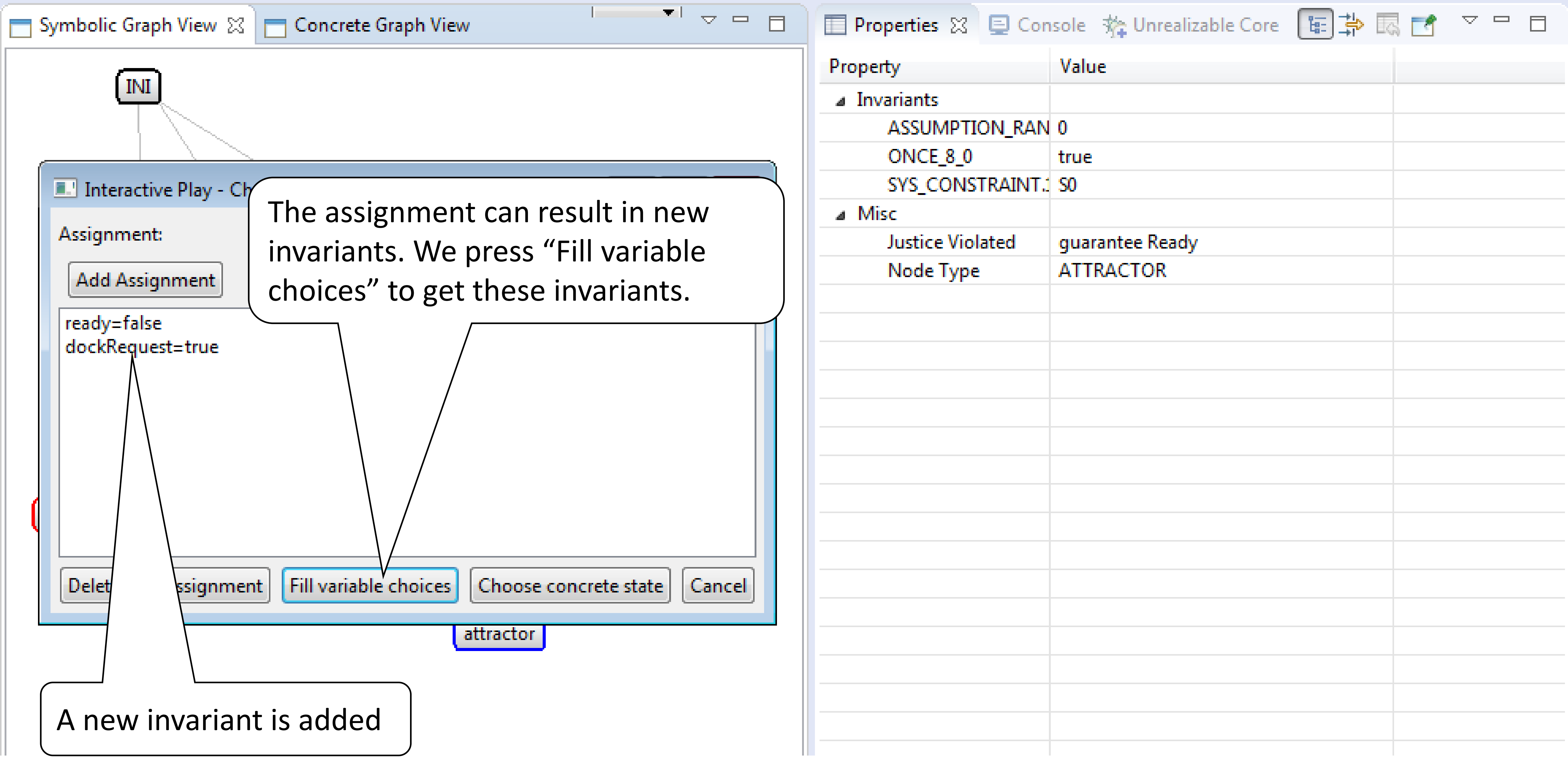


After choosing the value “`false`” for the “`ready`”, we press “Add Assignment”

ready=false

Assignment added

Property	Value
Invariants	
ASSUMPTION_RAN	0
ONCE_8_0	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR



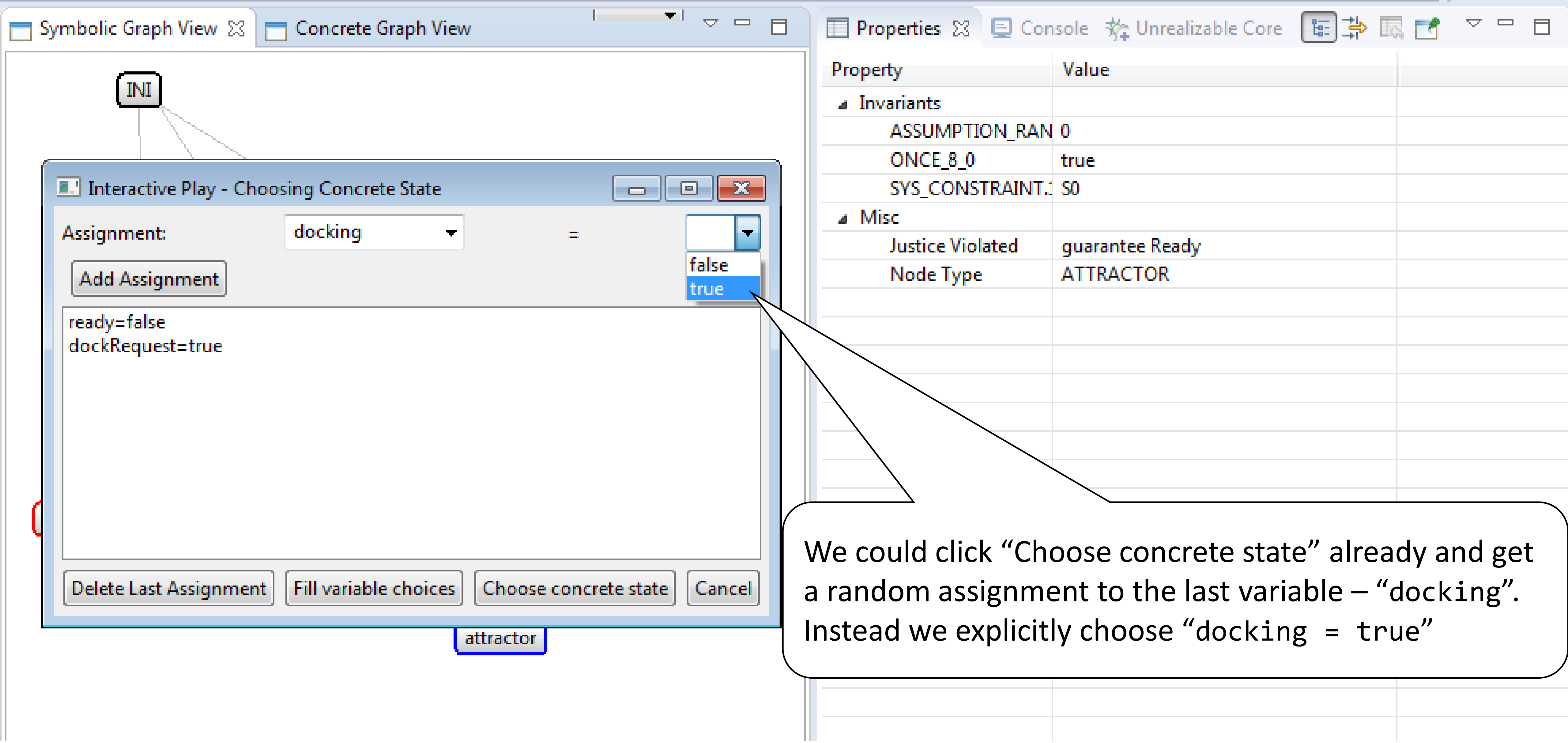
Assignment:

```
ready=false
dockRequest=true
```

Property	Value
Invariants	
ASSUMPTION_RAN	0
ONCE_8_0	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR

The assignment can result in new invariants. We press “Fill variable choices” to get these invariants.

A new invariant is added

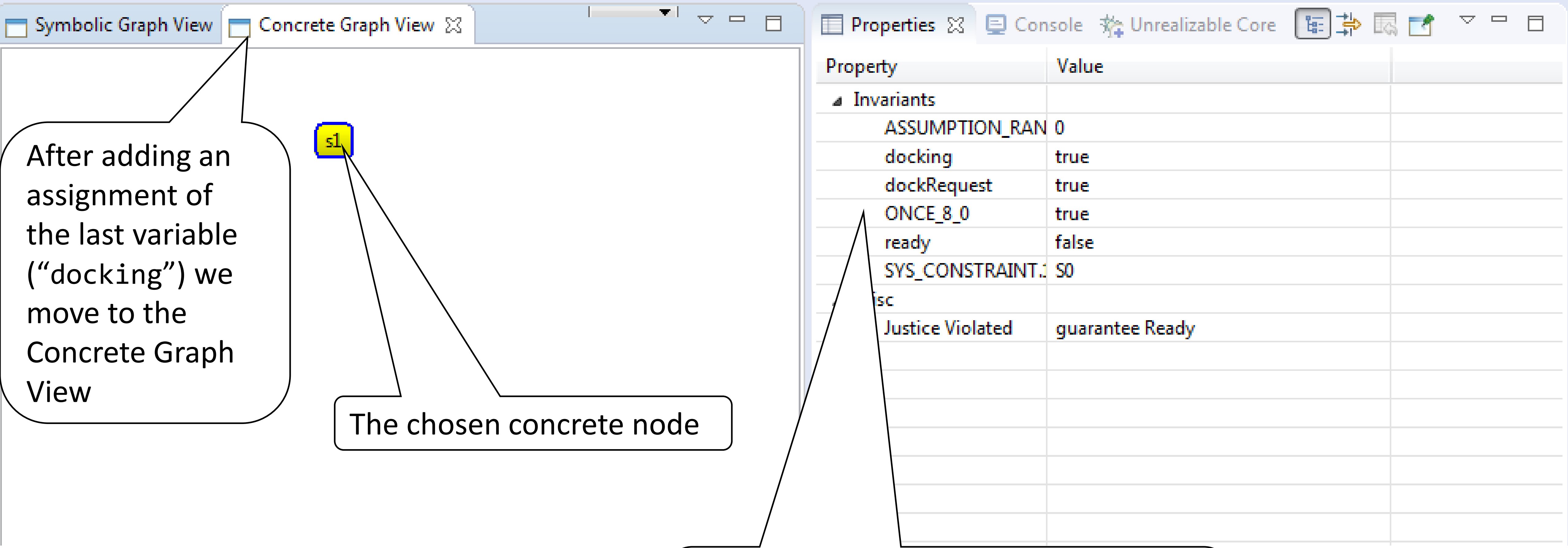


Assignment: docking =

```
ready=false
dockRequest=true
```

Property	Value
Invariants	
ASSUMPTION_RAN	0
ONCE_8_0	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR

We could click “Choose concrete state” already and get a random assignment to the last variable – “`docking`”. Instead we explicitly choose “`docking = true`”

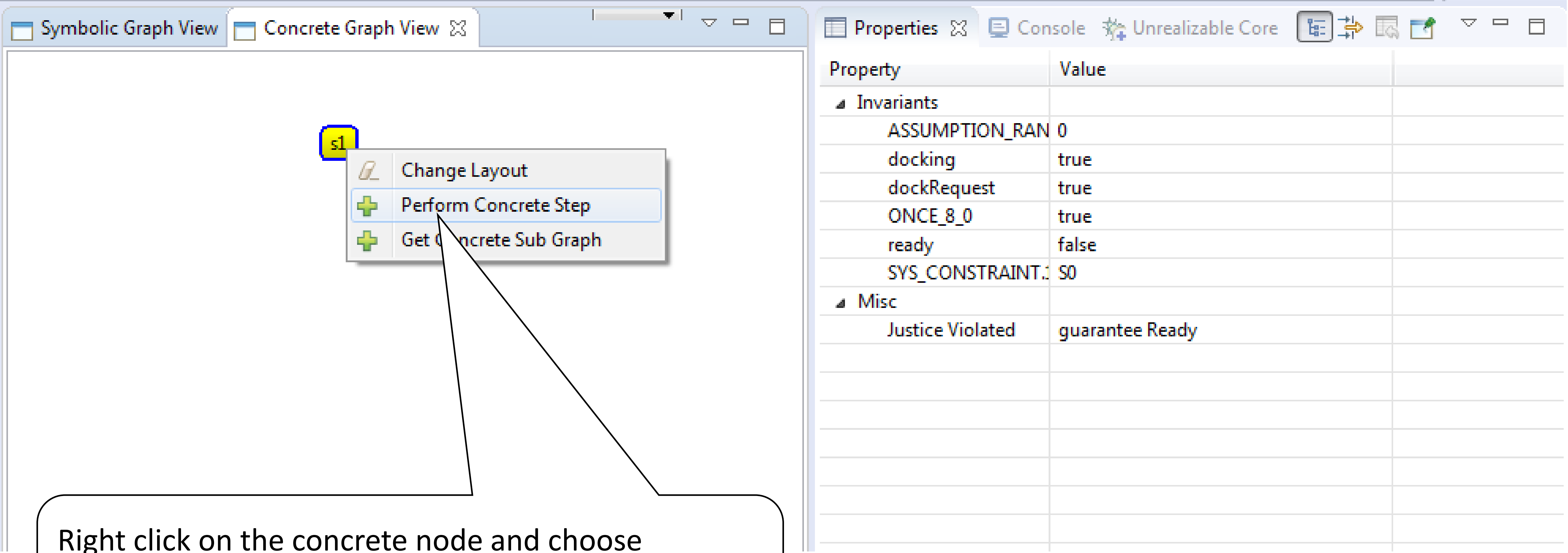


Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	true
dockRequest	true
ONCE_8_0	true
ready	false
SYS_CONSTRAINT.:	S0
isc	
Justice Violated	guarantee Ready

After adding an assignment of the last variable (“docking”) we move to the Concrete Graph View

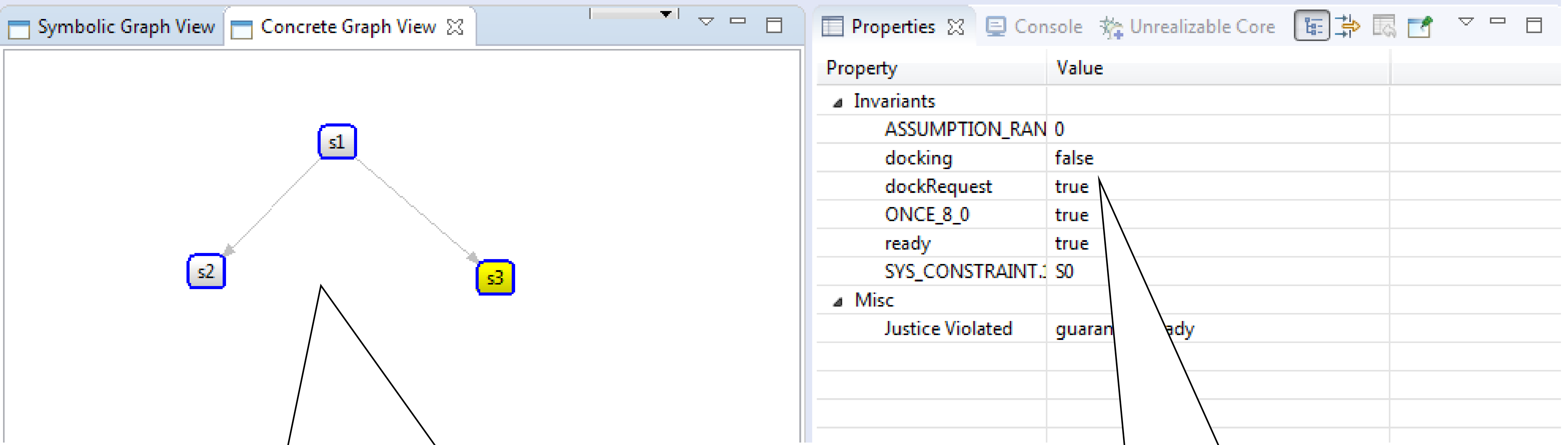
The chosen concrete node

From the invariants we see that this node is equivalent to node “s4” when we perform the “concrete node” operation



Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	true
dockRequest	true
ONCE_8_0	true
ready	false
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready

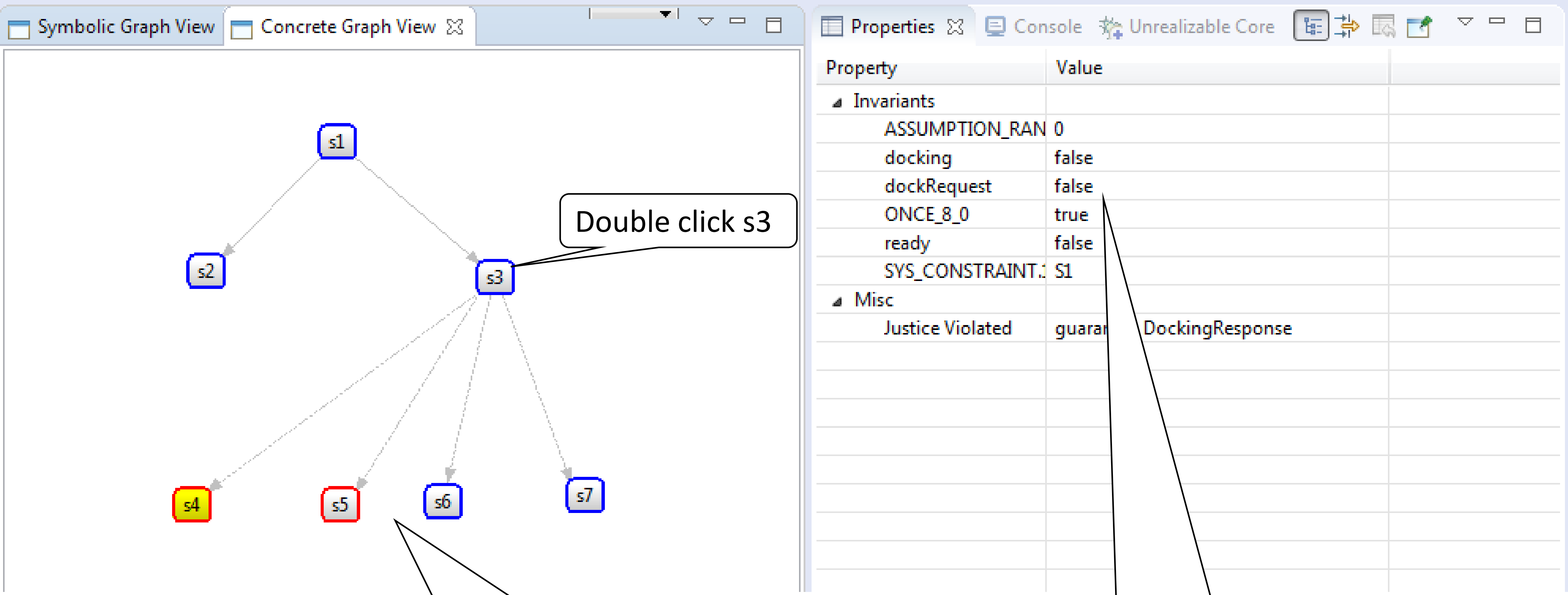
Right click on the concrete node and choose “Perform Concrete Step” to get s1 successors. Can also double-click on s1



Property	Value
◢ Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	true
ONCE_8_0	true
ready	true
SYS_CONSTRAINT.:	S0
◢ Misc	
Justice Violated	guaran ady

All s1 successors. If the number of successors was greater than 5, a pop up window similar to the interactive play window would open, requesting the engineer to choose assignments to get a specific successor.

s3 invariants. We see that docking is “`false`” (satisfying the safety of `G docking -> !next(docking)`), and dockRequest is “true”

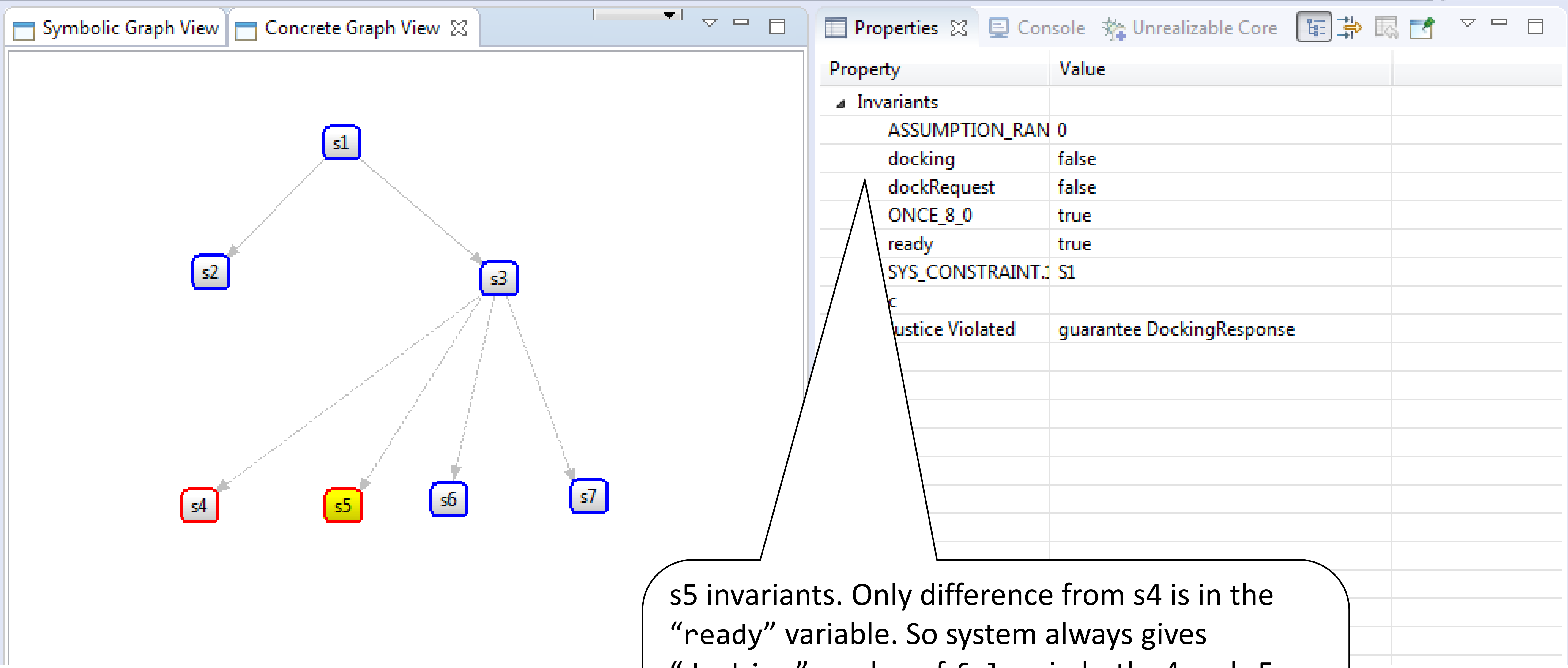


Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_8_0	true
ready	false
SYS_CONSTRAINT.:	S1
Misc	
Justice Violated	guarar DockingResponse

Double click s3

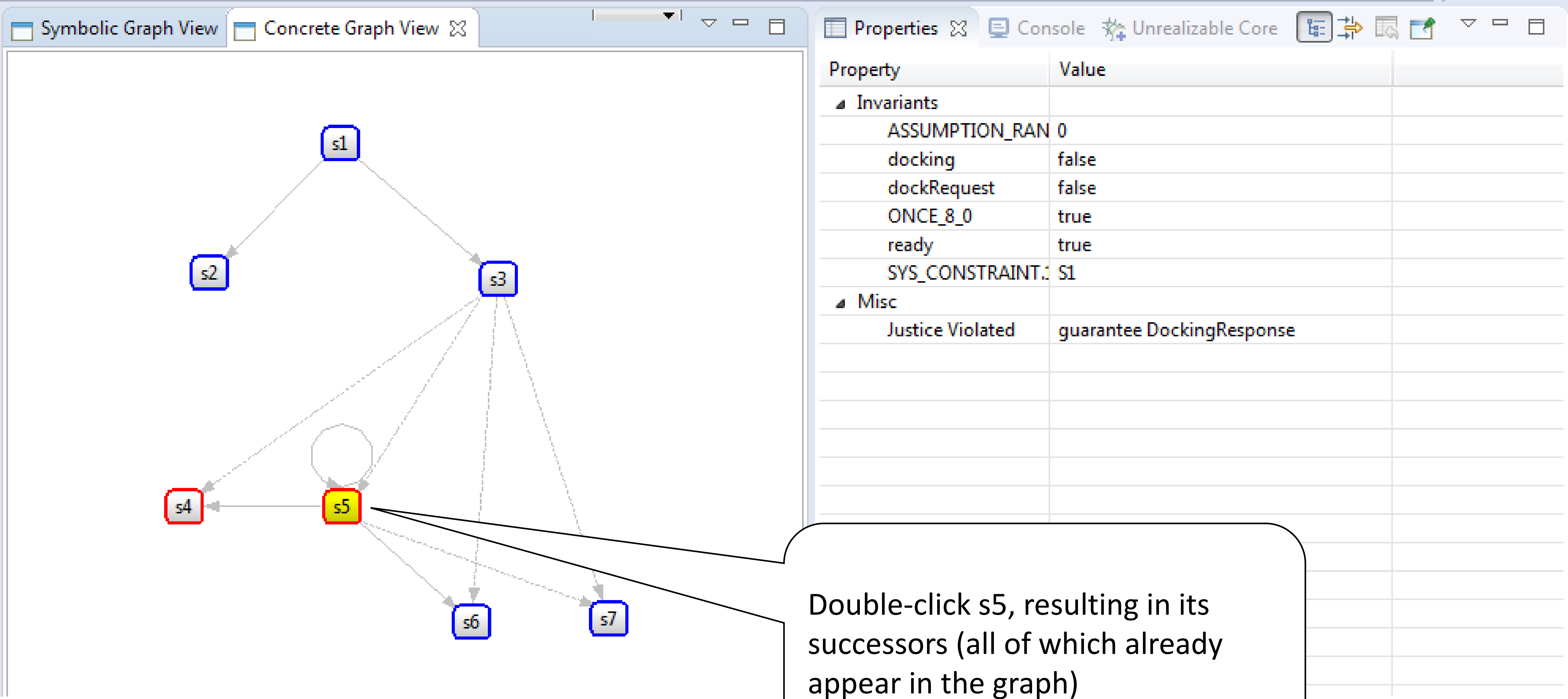
Successors of s3. Dotted edges indicate they're outside the attractor containing s3. Red nodes are contained in a cycle JVTS node

Invariants of s4. We see that `dockRequest` is now `false`, when it previously was `true` (s3).



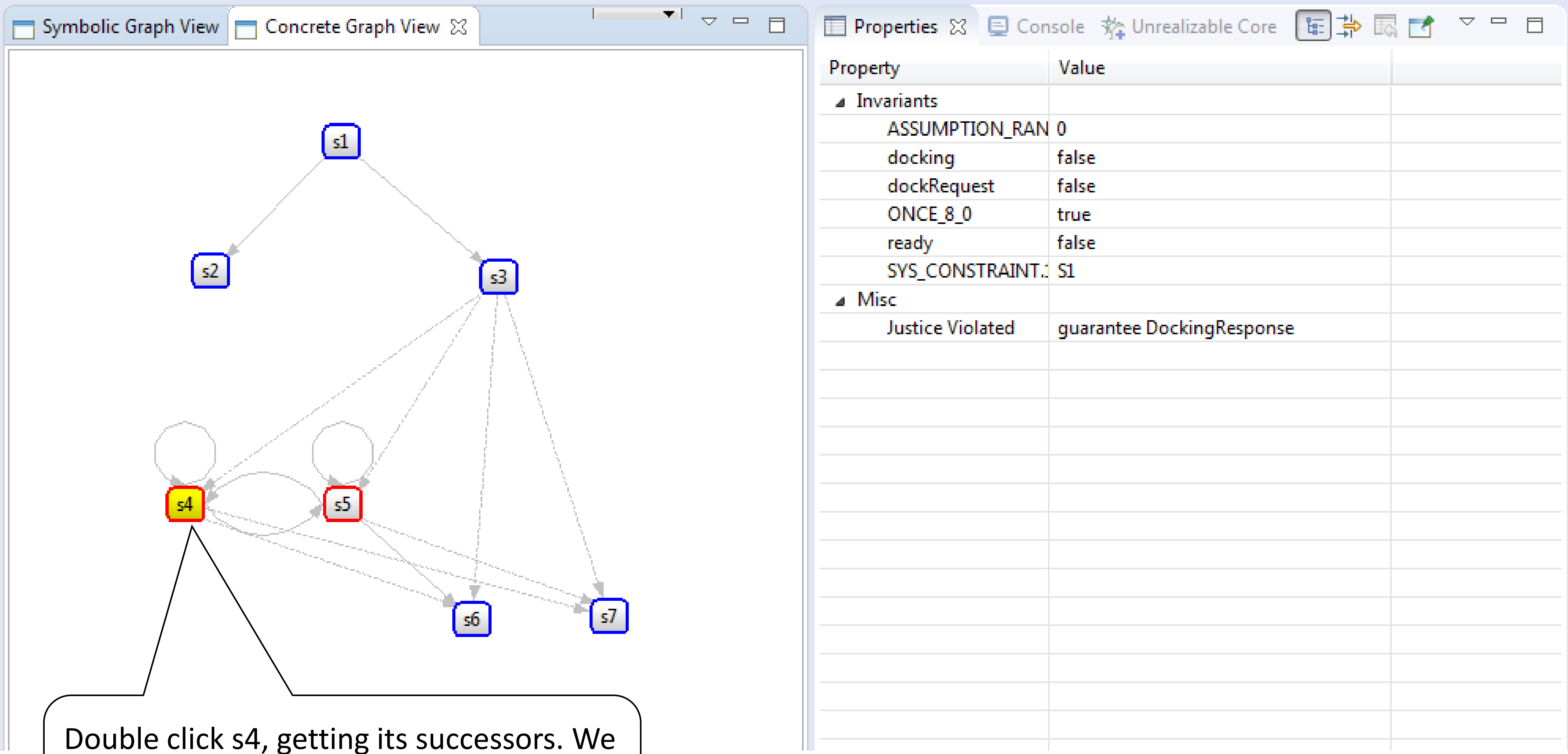
Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_8_0	true
ready	true
SYS_CONSTRAINT.:	S1
c	
ustice Violated	guarantee DockingResponse

s5 invariants. Only difference from s4 is in the “ready” variable. So system always gives “docking” a value of false, in both s4 and s5, violating the justice guarantee “DockingResponse” which ensures every “dockRequest” will eventually have a “docking” granted.



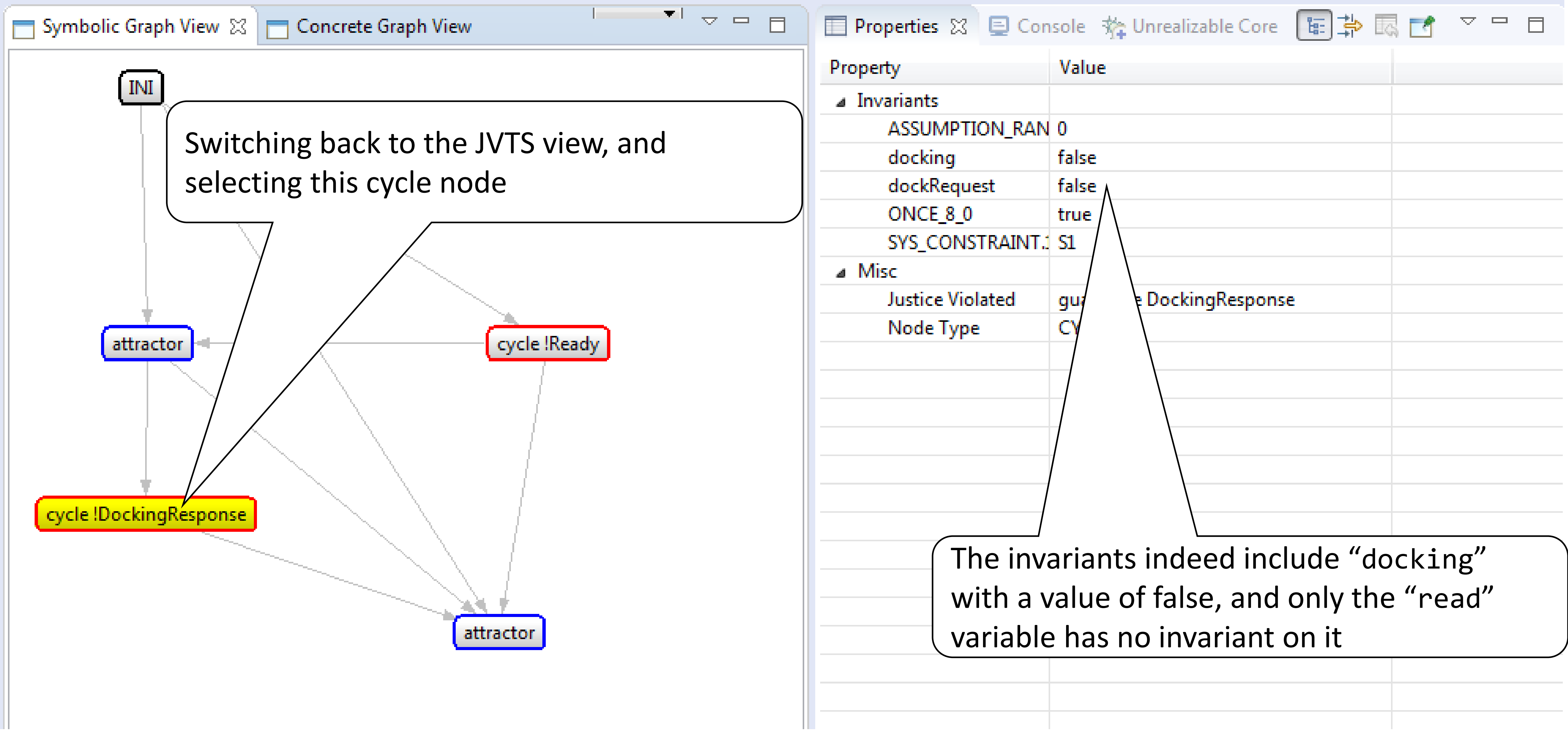
Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_8_0	true
ready	true
SYS_CONSTRAINT.:	S1
Misc	
Justice Violated	guarantee DockingResponse

Double-click s5, resulting in its successors (all of which already appear in the graph)



Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_8_0	true
ready	false
SYS_CONSTRAINT.:	S1
Misc	
Justice Violated	guarantee DockingResponse

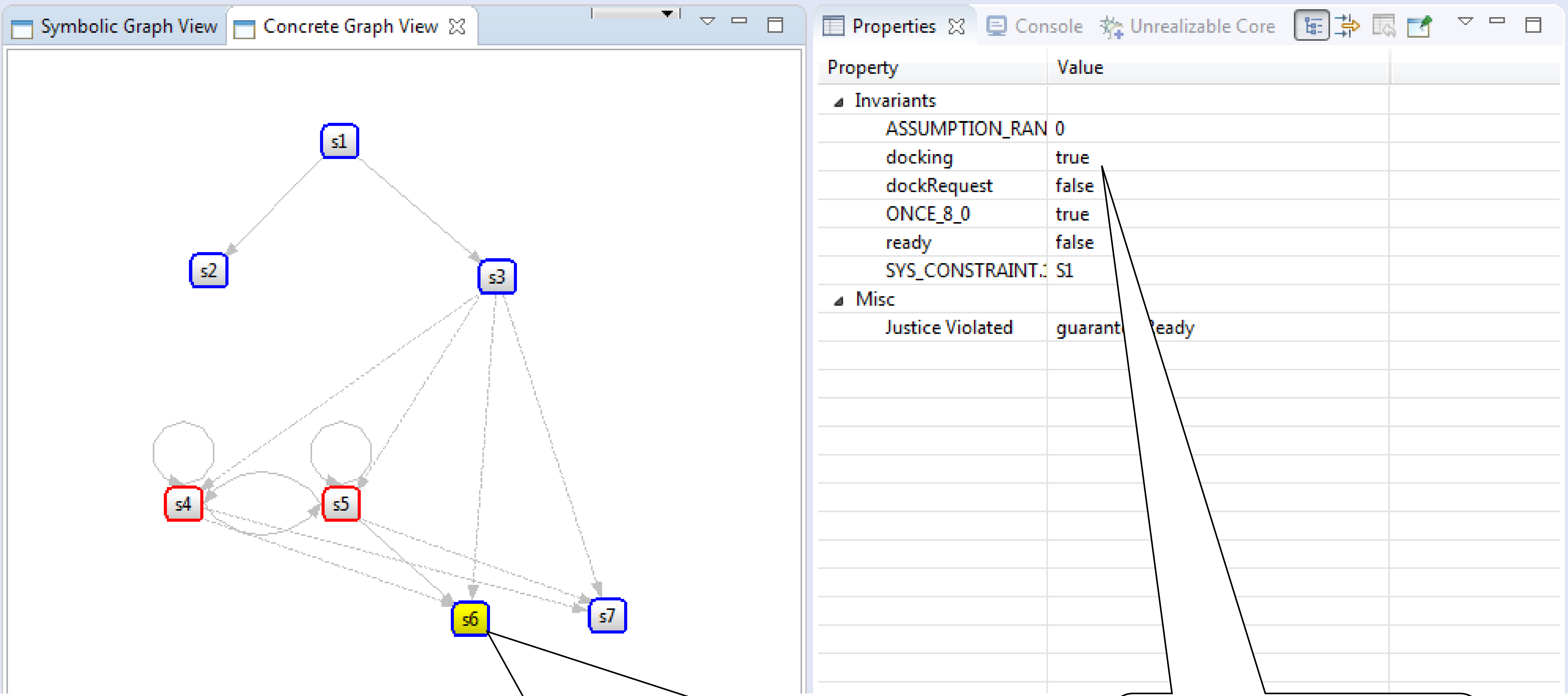
Double click s4, getting its successors. We see the cycle JVTS node is comprised only of s4 and s5



Switching back to the JVTS view, and selecting this cycle node

Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_8_0	true
SYS_CONSTRAINT.:	S1
Misc	
Justice Violated	gu e DockingResponse
Node Type	CY

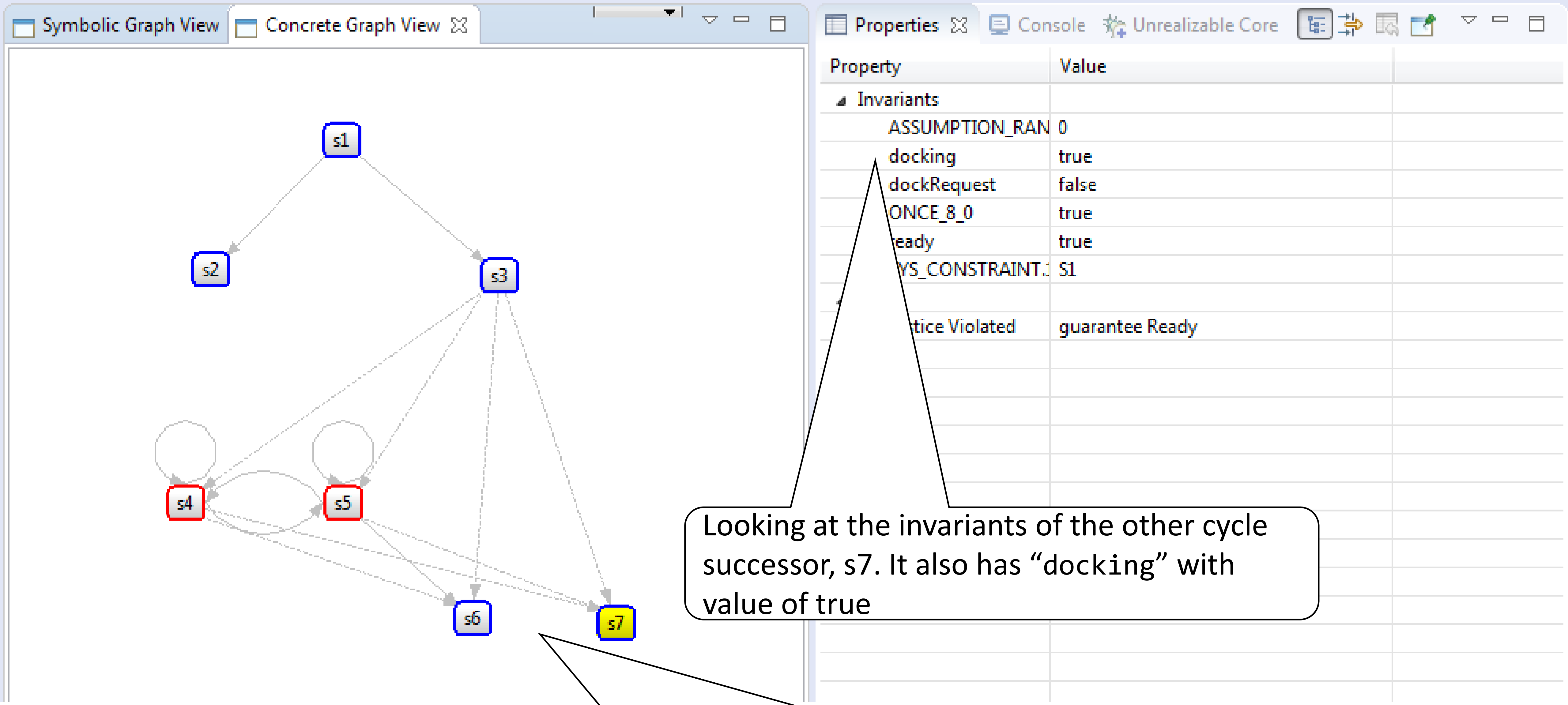
The invariants indeed include "docking" with a value of false, and only the "read" variable has no invariant on it



Property	Value
◢ Invariants	
ASSUMPTION_RAN	0
docking	true
dockRequest	false
ONCE_8_0	true
ready	false
SYS_CONSTRAINT.	S1
◢ Misc	
Justice Violated	guarant[illegible]eady

To understand why system does not set “docking” to true in order to satisfy the justice guarantee, we look at s6 invariants (one of the cycle successors)

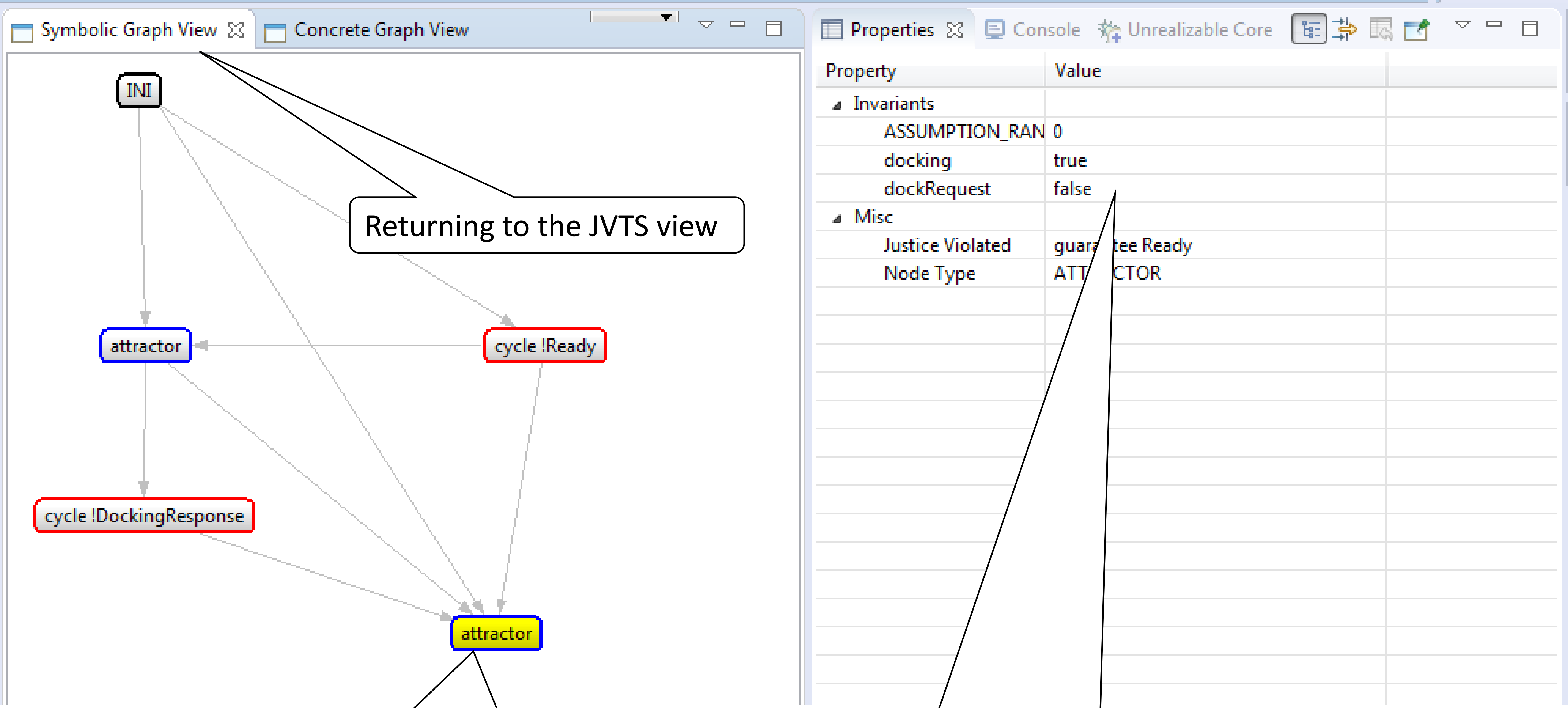
s6 invariants include “docking” with a value of `true`.



Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	true
dockRequest	false
ONCE_8_0	true
eady	true
YS_CONSTRAINT.:	S1
tice Violated	guarantee Ready

Looking at the invariants of the other cycle successor, s7. It also has “docking” with value of true

Double clicking on s6 or s7 does not result in any successors – they are dead-end state. From the variable assignments on them we see that they violate the safety guarantee `G docking -> dockRequest`



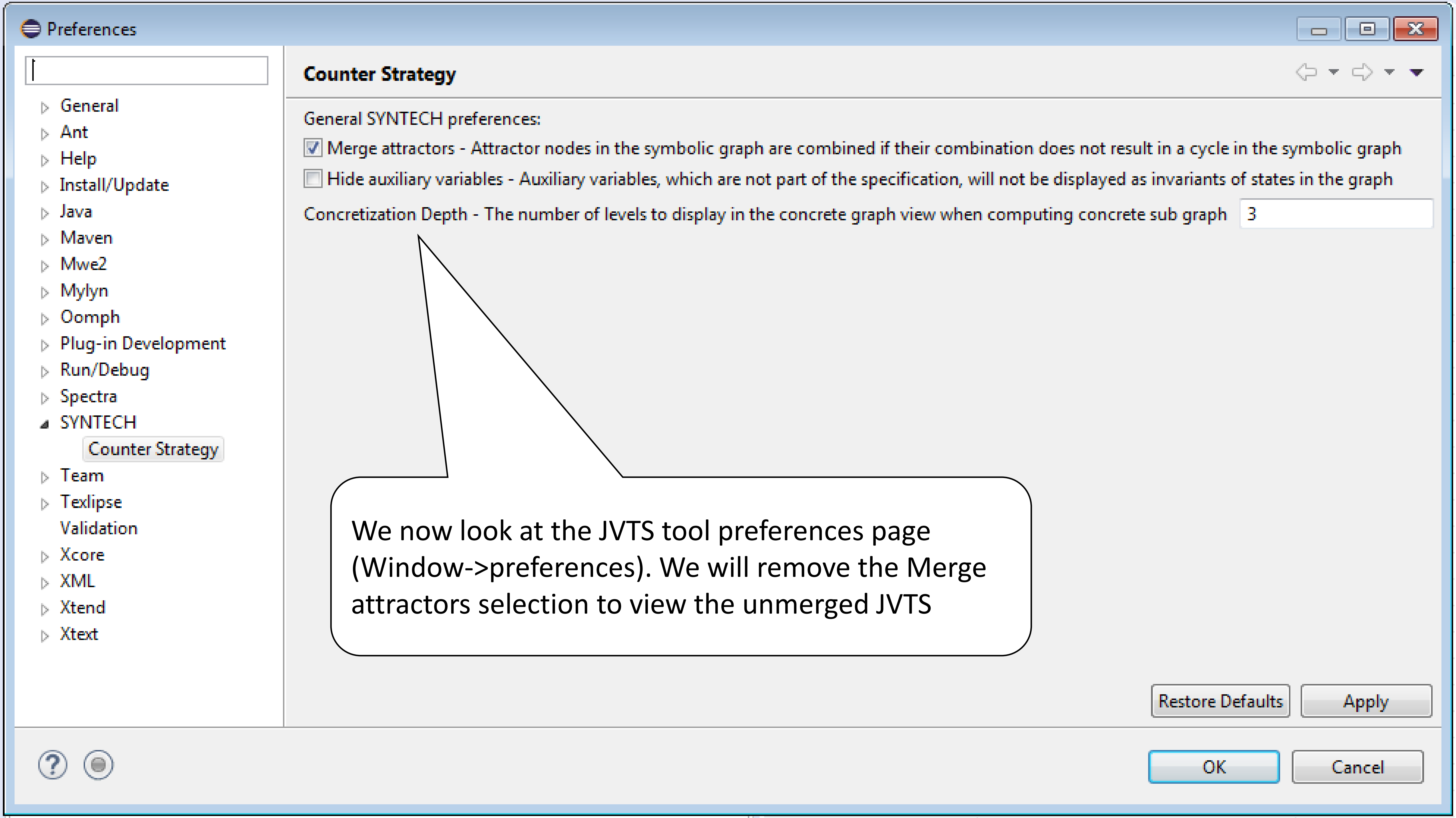
Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	true
dockRequest	false
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTOR

Returning to the JVTS view

Selecting the dead-end attractor

The dead-end attractor invariants, showing the invariants that result in the safety guarantee violation

Preferences: Merging Attractors



Counter Strategy

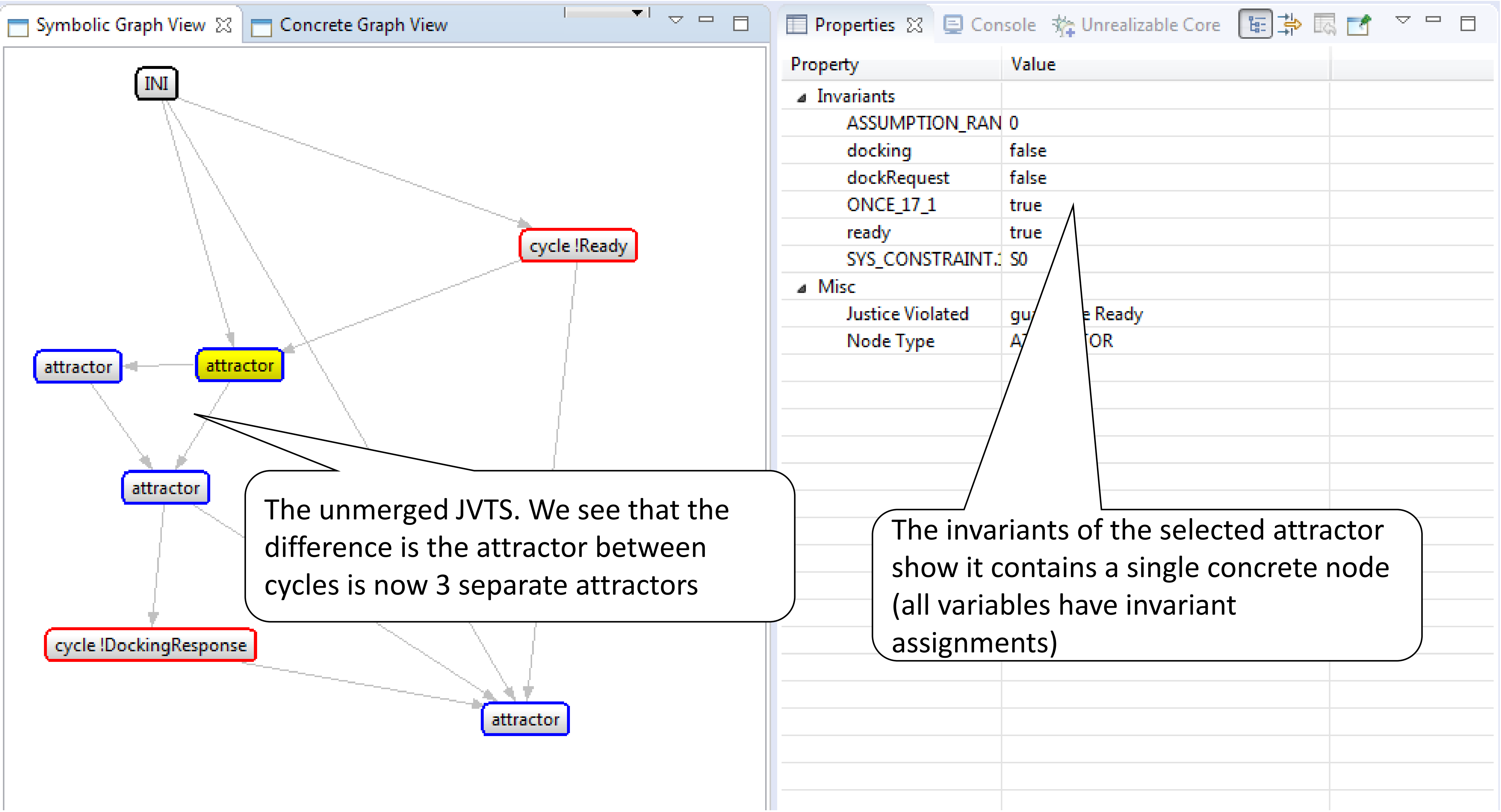
General SYNTECH preferences:

☑ Merge attractors - Attractor nodes in the symbolic graph are combined if their combination does not result in a cycle in the symbolic graph

□ Hide auxiliary variables - Auxiliary variables, which are not part of the specification, will not be displayed as invariants of states in the graph

Concretization Depth - The number of levels to display in the concrete graph view when computing concrete sub graph: 3

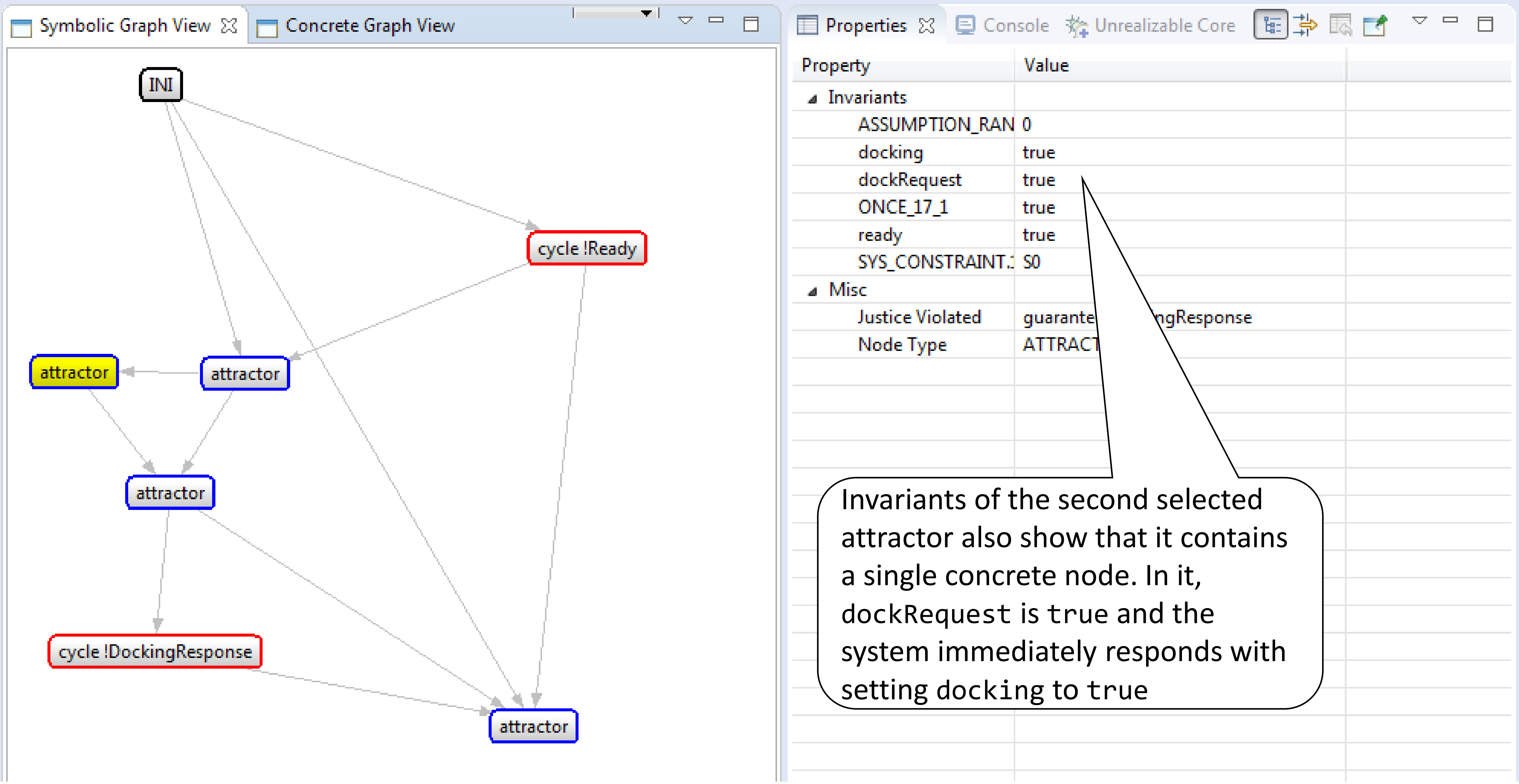
We now look at the JVTS tool preferences page (Window->preferences). We will remove the Merge attractors selection to view the unmerged JVTS



Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_17_1	true
ready	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	gu e Ready
Node Type	A OR

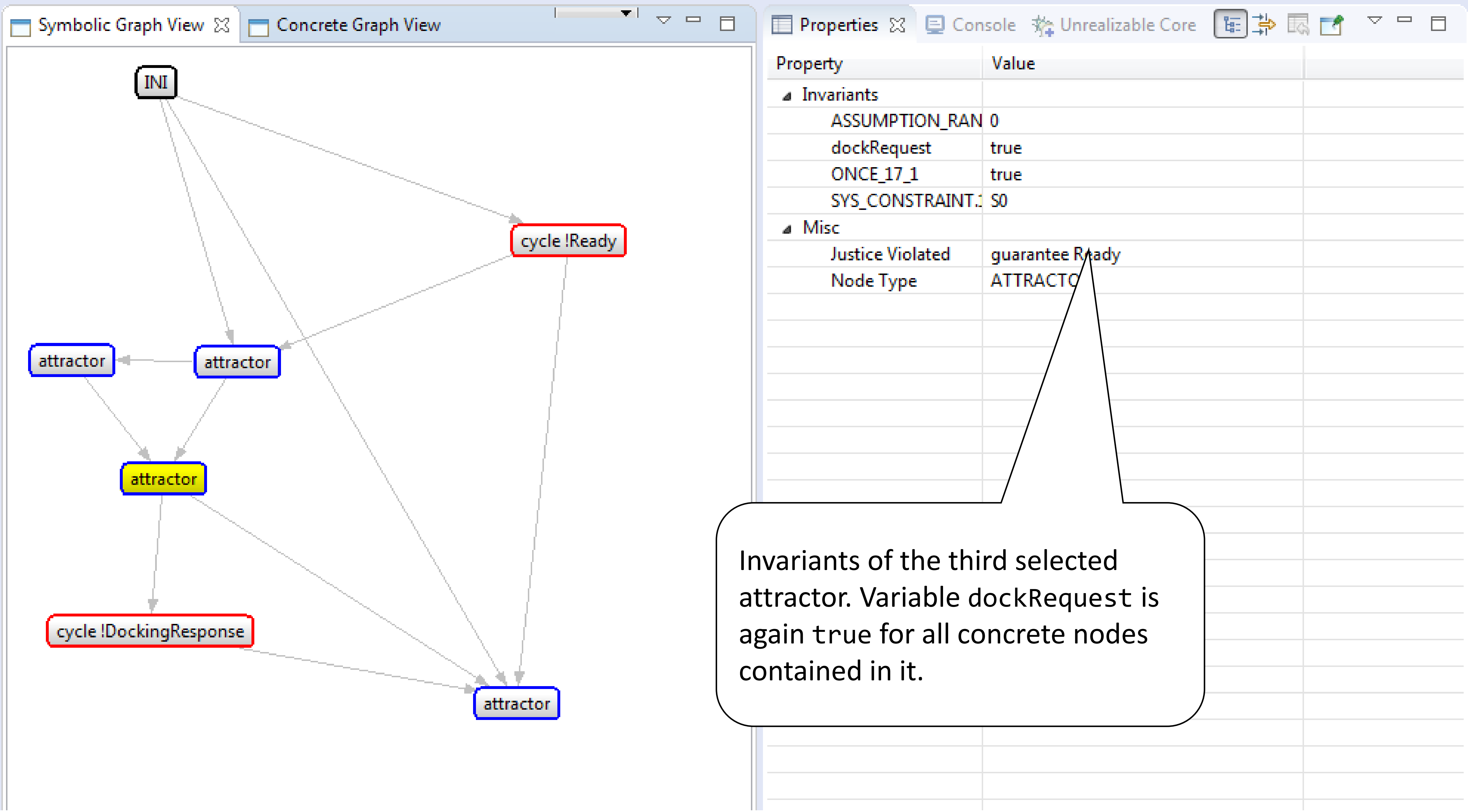
The unmerged JVTS. We see that the difference is the attractor between cycles is now 3 separate attractors

The invariants of the selected attractor show it contains a single concrete node (all variables have invariant assignments)



Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	true
dockRequest	true
ONCE_17_1	true
ready	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarante ngResponse
Node Type	ATTRACT

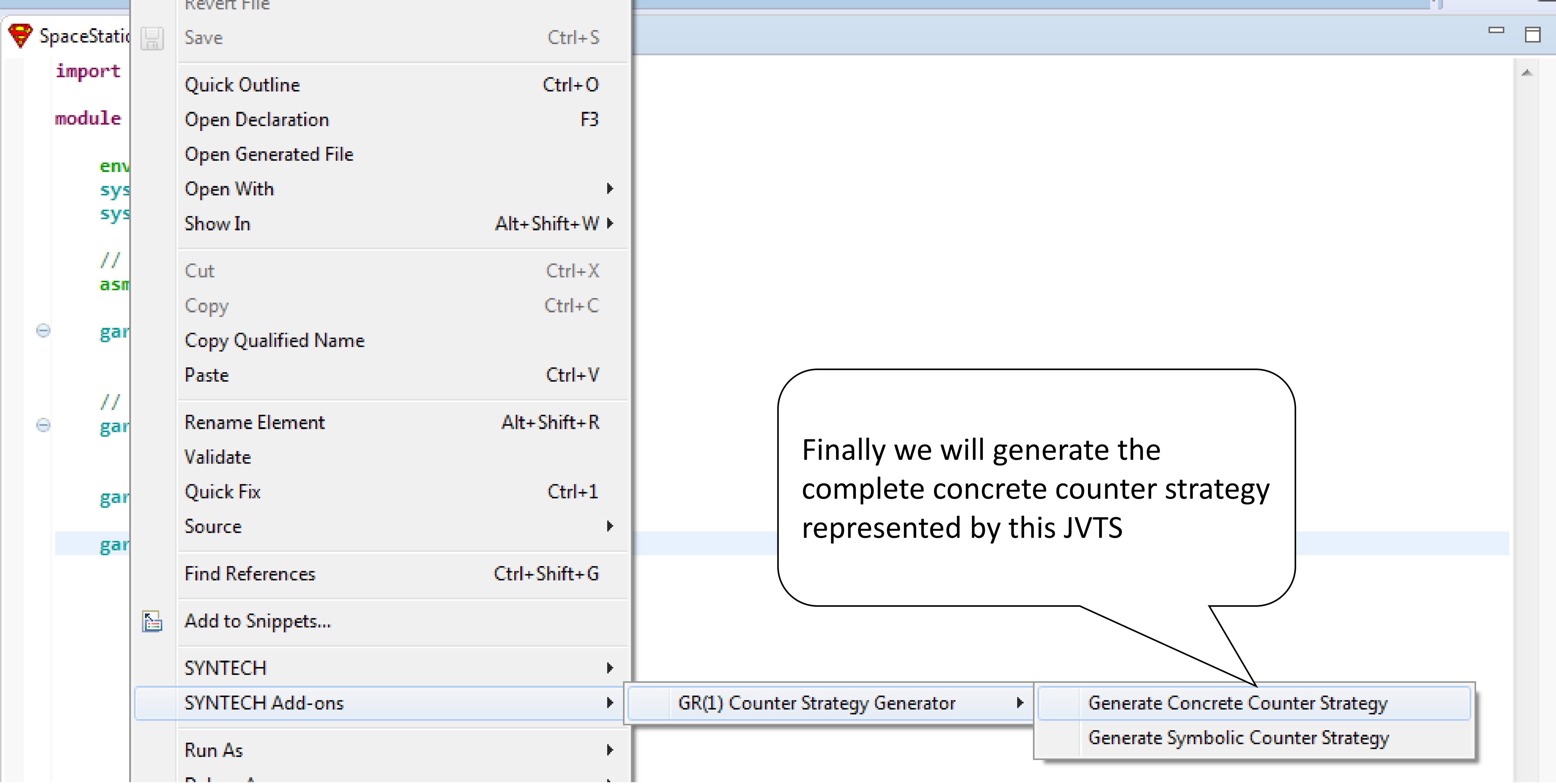
Invariants of the second selected attractor also show that it contains a single concrete node. In it, `dockRequest` is `true` and the system immediately responds with setting `docking` to `true`



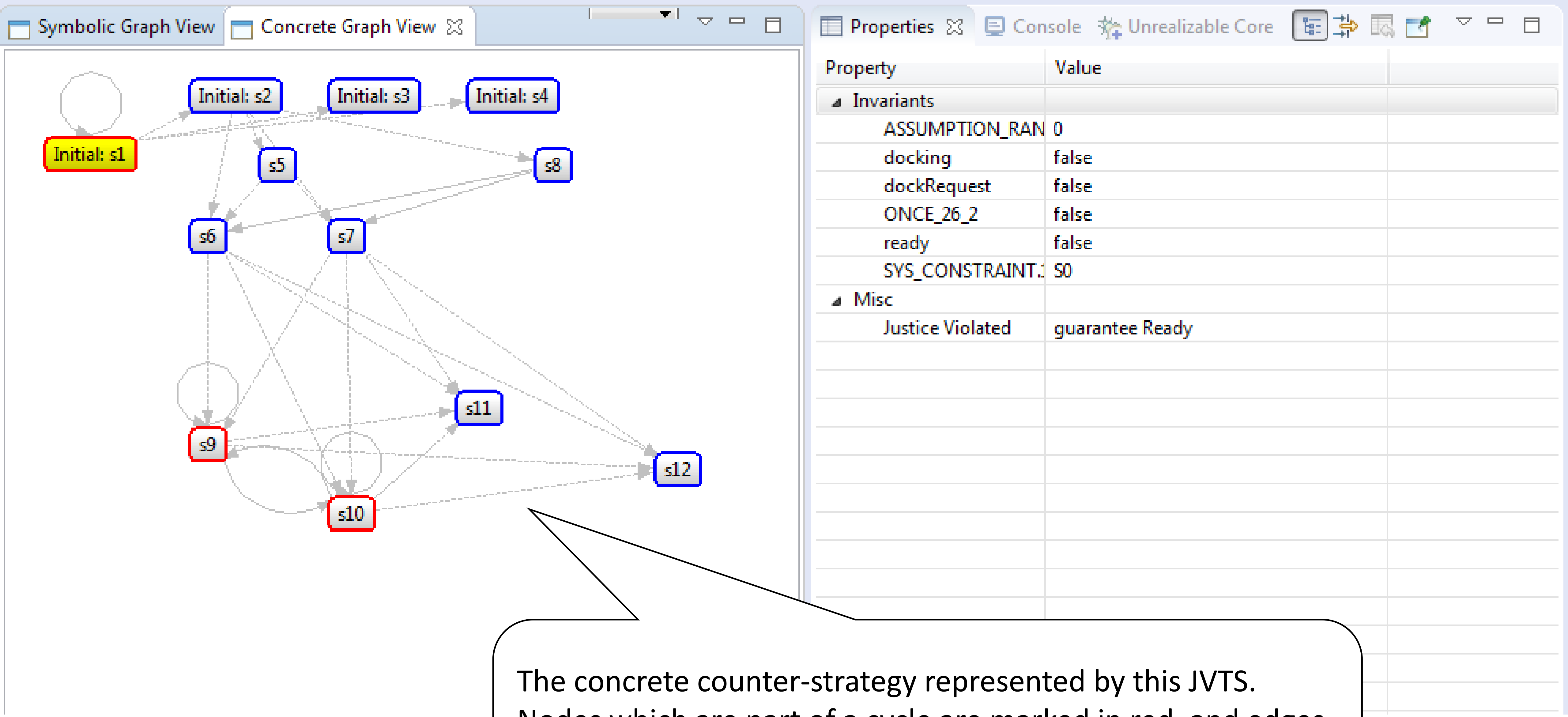
Property	Value
Invariants	
ASSUMPTION_RAN	0
dockRequest	true
ONCE_17_1	true
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready
Node Type	ATTRACTO

Invariants of the third selected attractor. Variable `dockRequest` is again `true` for all concrete nodes contained in it.

Generate Concrete Counter Strategy



Finally we will generate the complete concrete counter strategy represented by this JVTS



Property	Value
Invariants	
ASSUMPTION_RAN	0
docking	false
dockRequest	false
ONCE_26_2	false
ready	false
SYS_CONSTRAINT.:	S0
Misc	
Justice Violated	guarantee Ready

The concrete counter-strategy represented by this JVTS. Nodes which are part of a cycle are marked in red, and edges between nodes in contained in different JVTS nodes are dotted