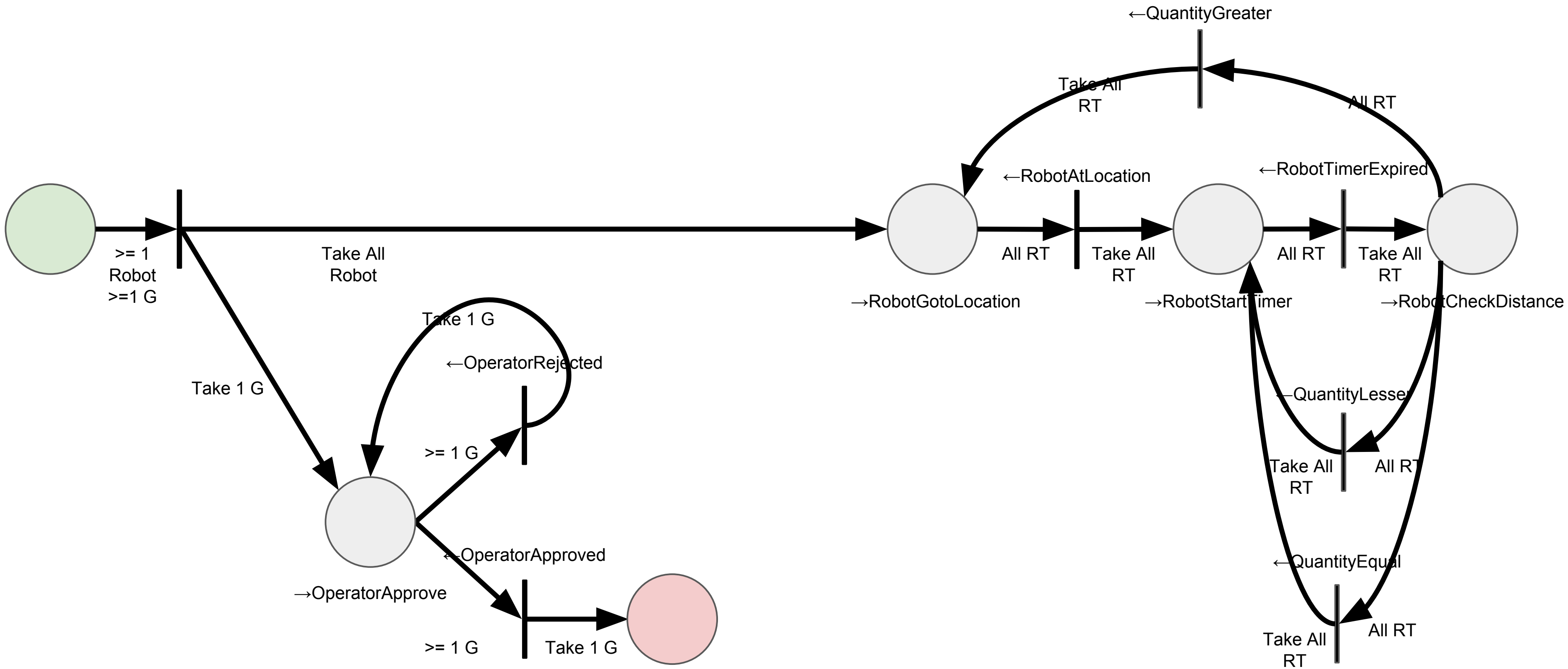
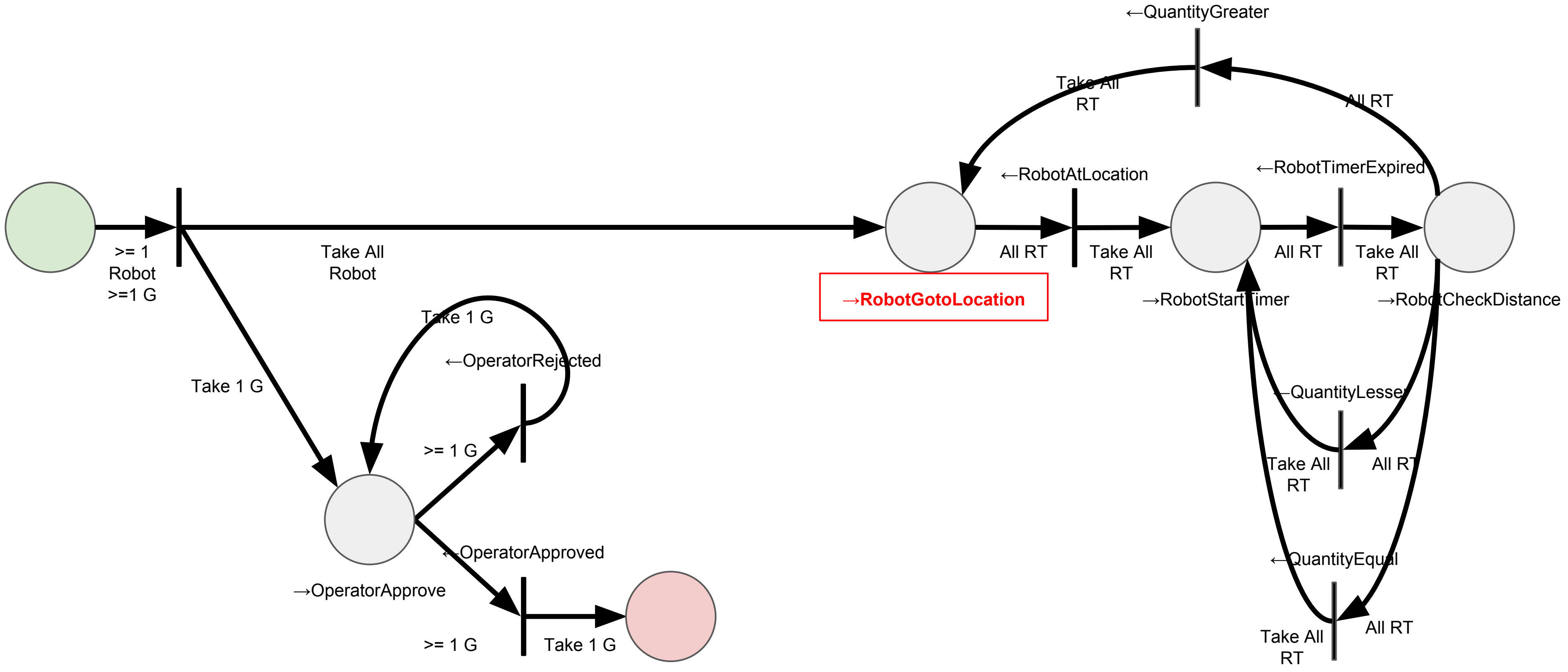
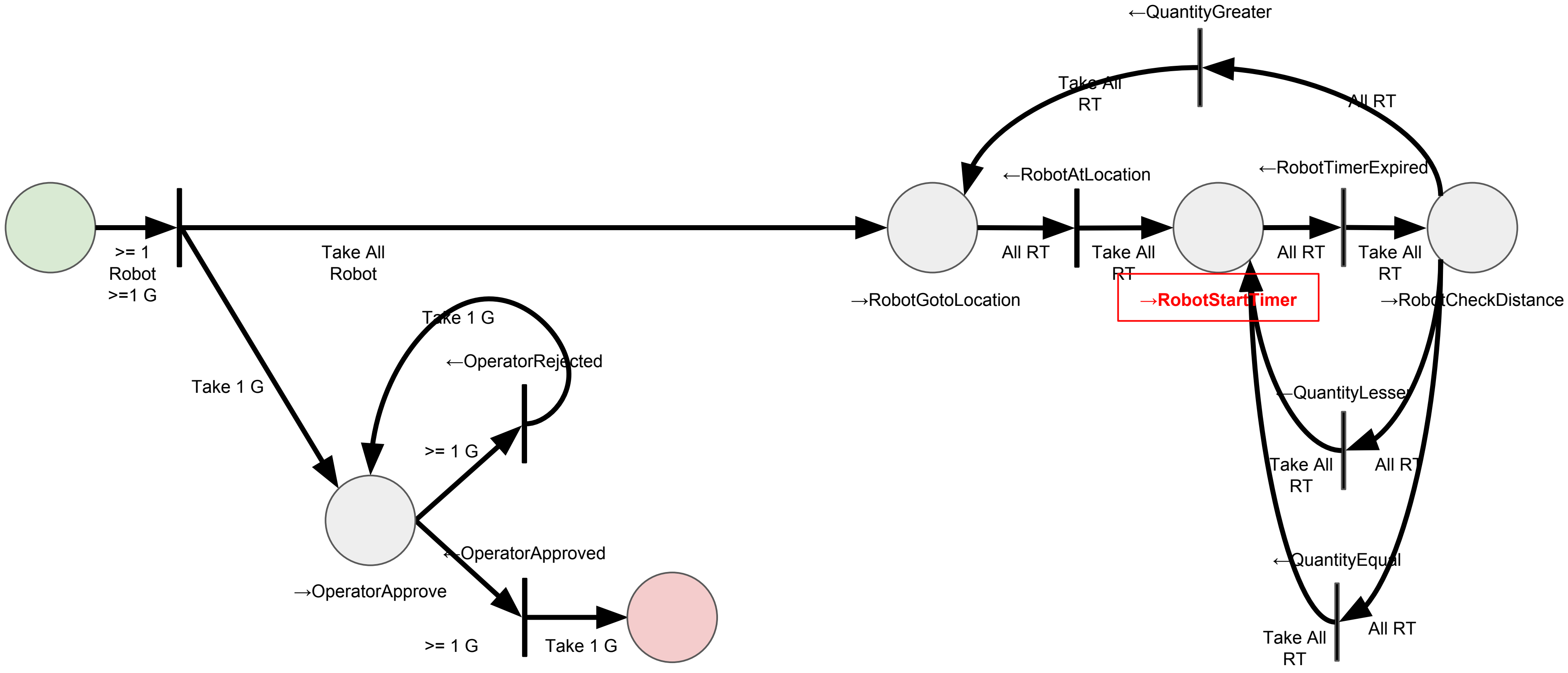
Variables

Let’s consider the pieces of information needed to run the station keep SPN:

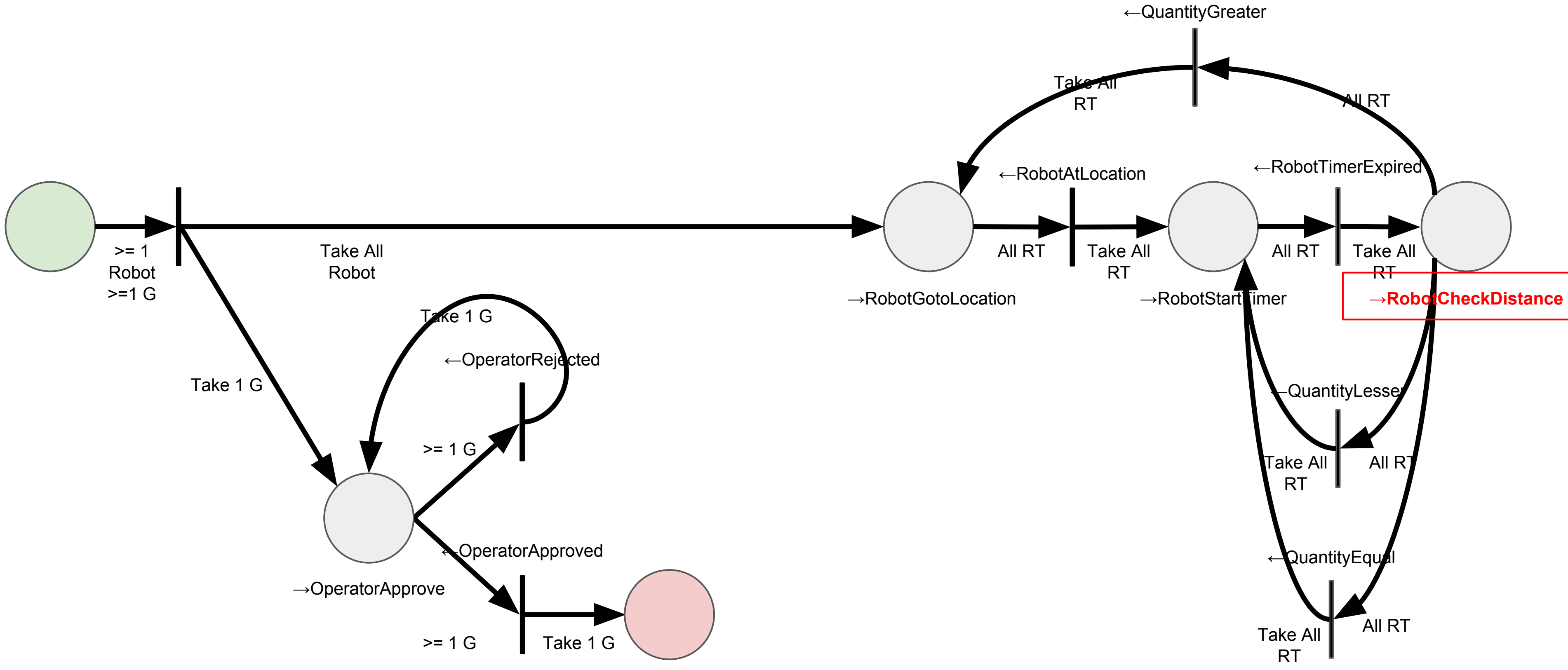




Where should the boats go?



How long should the timer last?



What location should the boat calculate its distance to?

How far is “too far”?

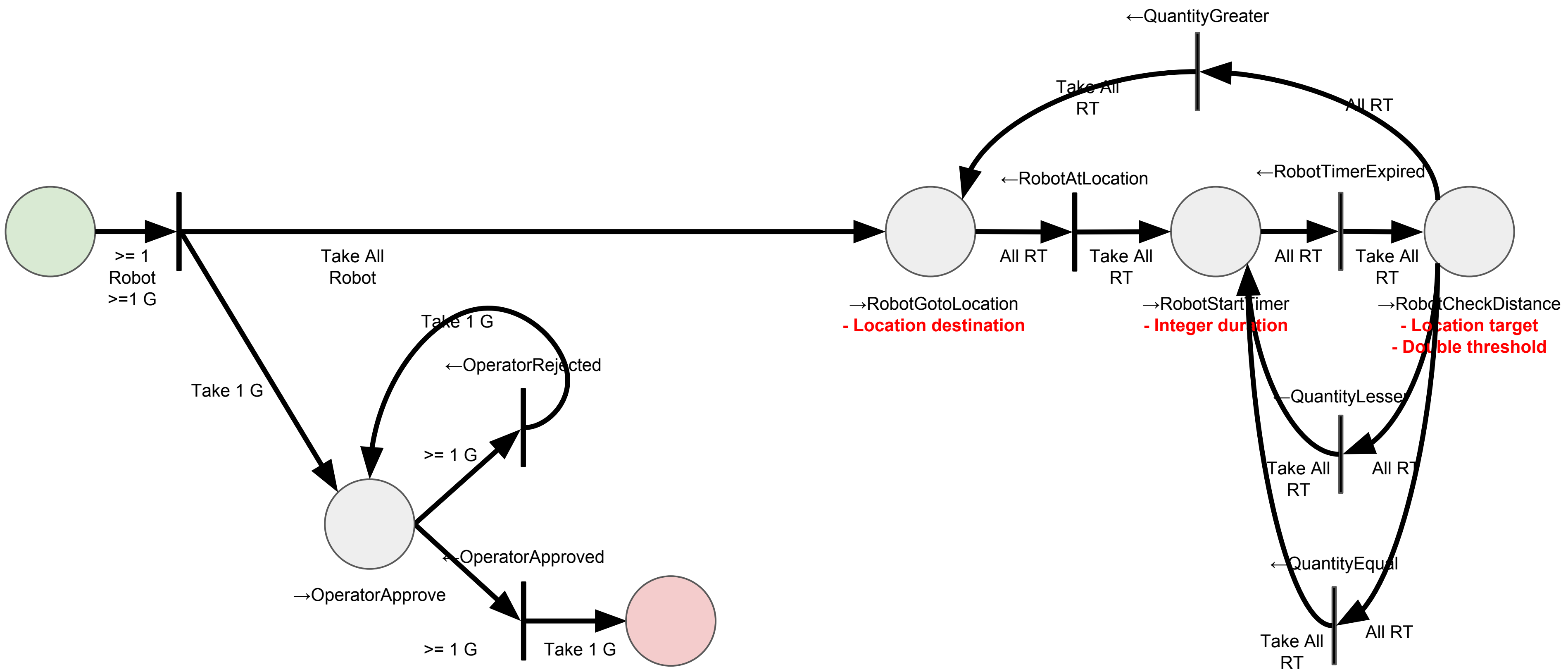
Each of these pieces of information is called a “field.” Each output and input event can have any number of fields. Each field has a type, name, and assignable value.

For output events, the values of their fields are used to make the request to the operator, robot, or AI.

For input events, the values of their fields are assigned by the operator, robot, or AI when the input event is generated.

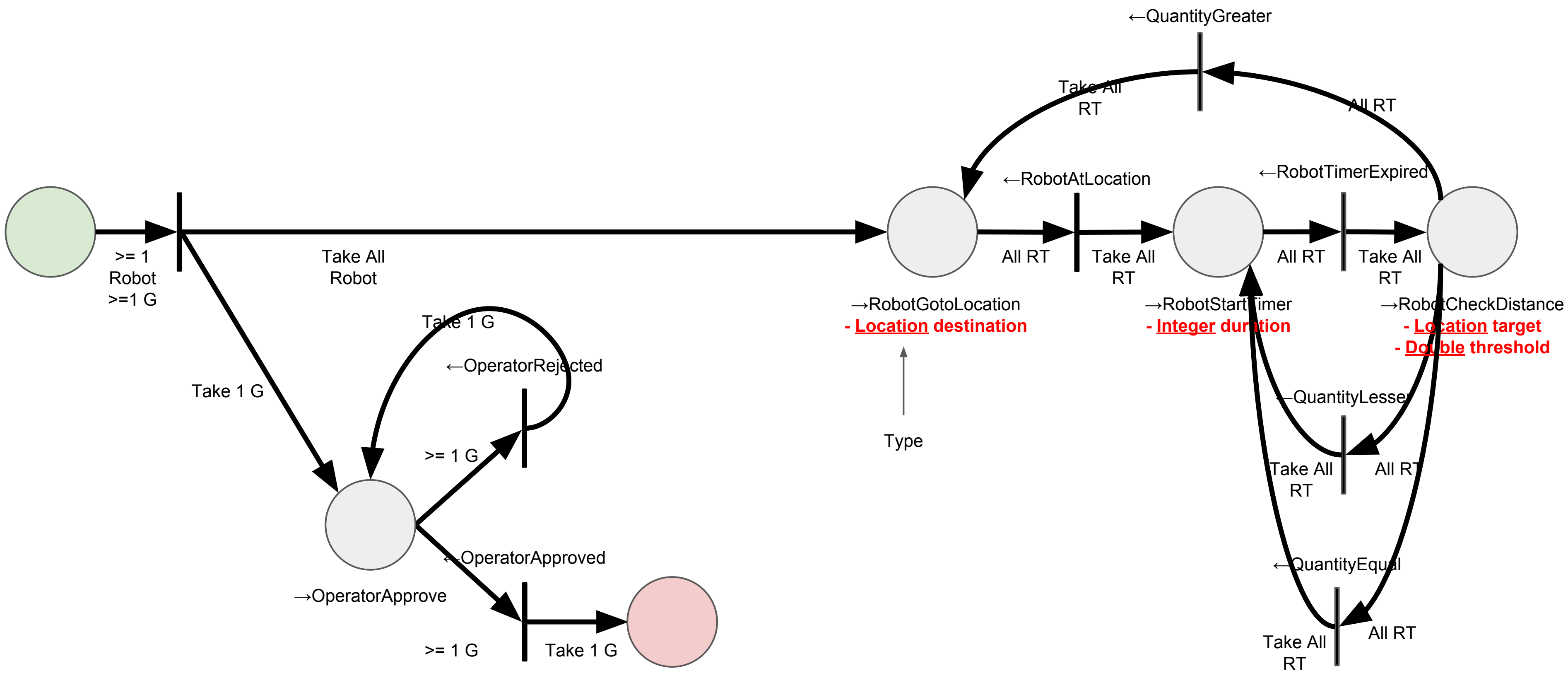
First let’s look at each output event’s fields.

Note that →OperatorApprove has 0 and →RobotCheckDistance has 2



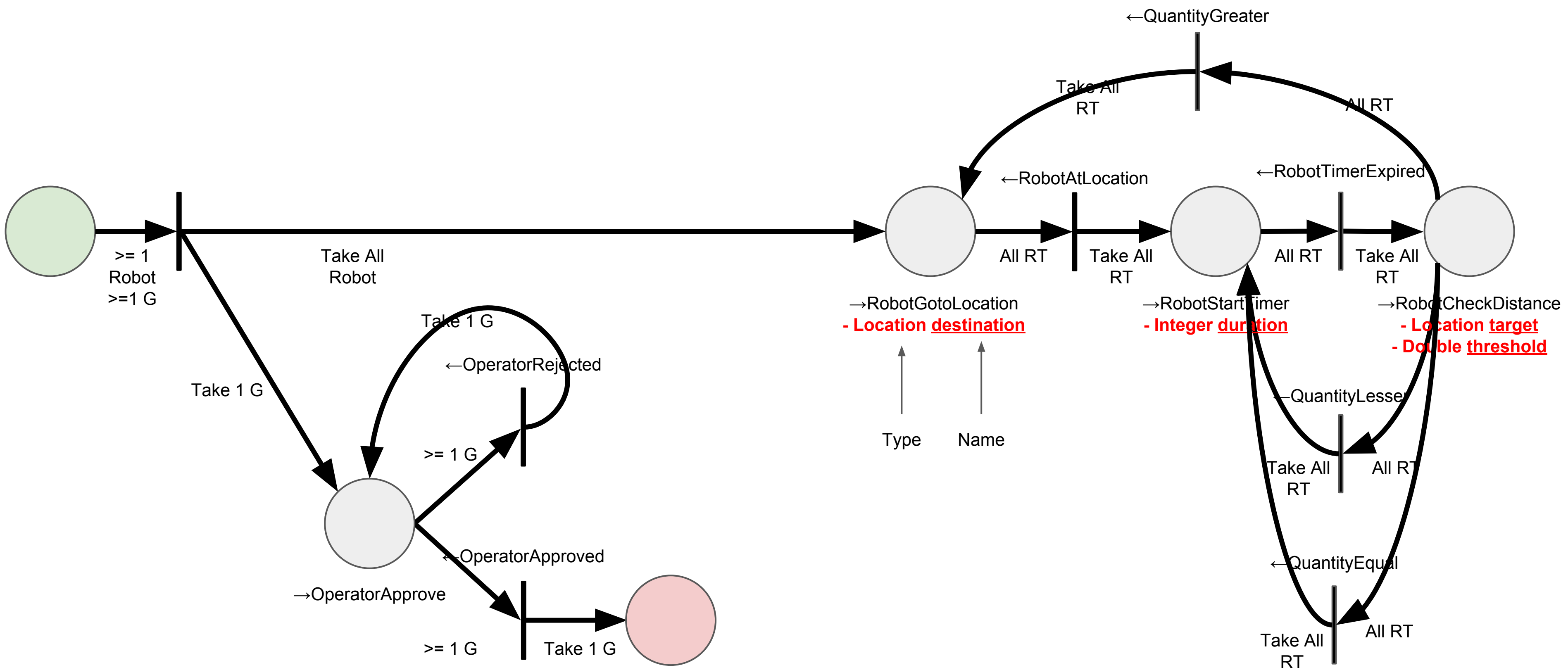
First let's look at each output event's fields.

Underlined is each field's type.



First let’s look at each output event’s fields.

Underlined is each field’s name.

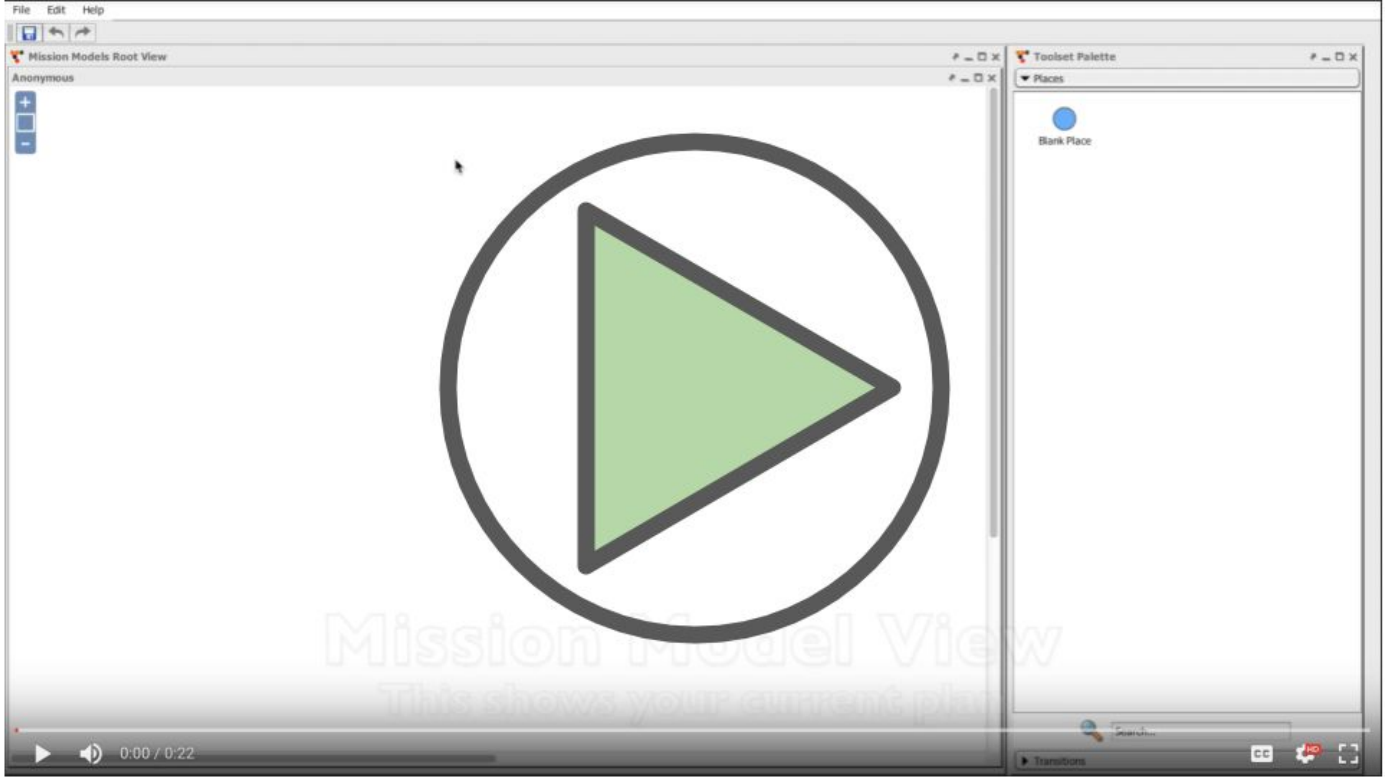


For output event fields, there are 3 ways to assign it a value (also called a definition).

1- Assign it a value when developing the SPN

If we know the value we want to use when developing the SPN in DREAMM, we can assign it then.

For example, let’s say we always want to wait 10 seconds between distance checks.



Watch “Setting a Static Value”: This video will show you how to define the value of an output event’s field in DREAMM.

Job 8-1: Define the →RobotStartTimer output event’s Integer duration to 10 seconds

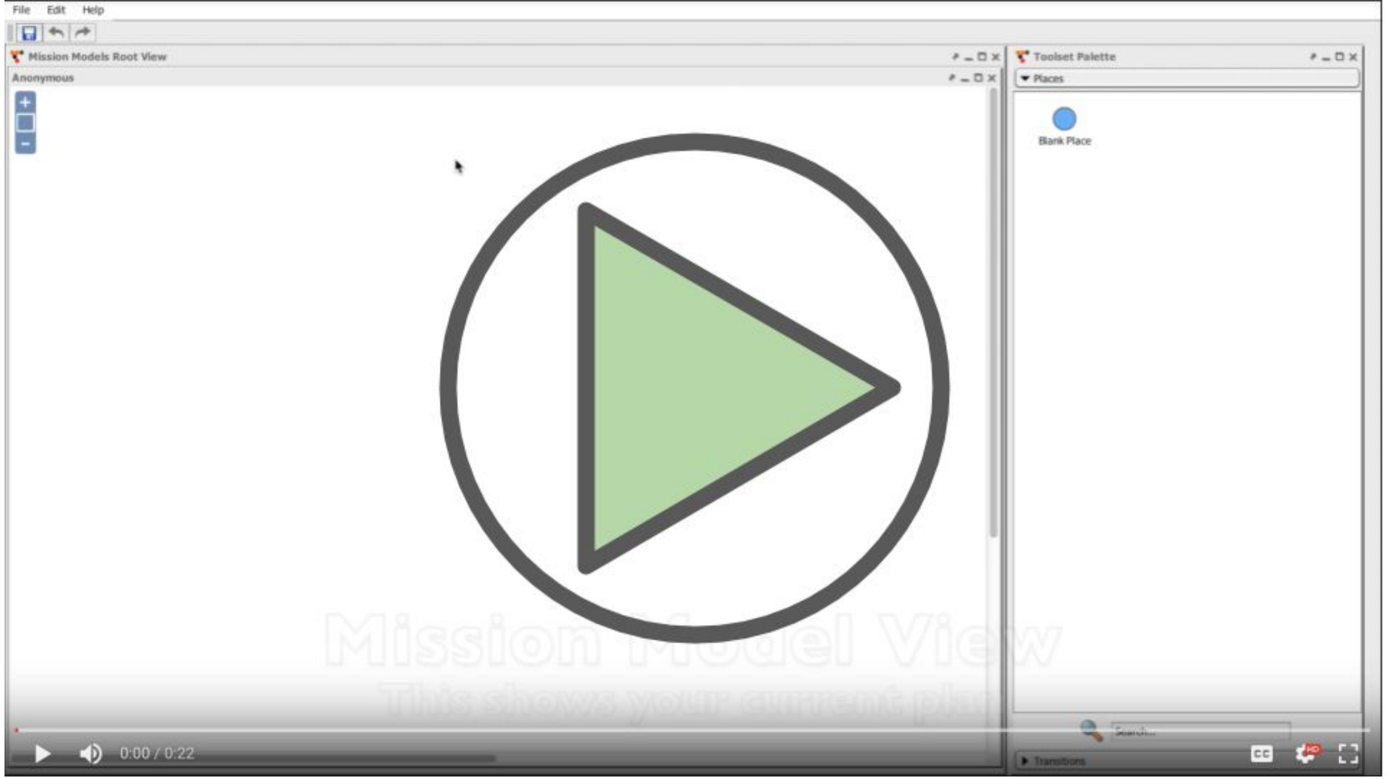
For output events, there are 3 ways to assign a value (also called a definition) to a field.

1- Assign it a value when developing the SPN

2- Assign it a value when starting the SPN

If we won’t know the value of a field until the the operator starts the plan, we can have the operator define the value at that point.

For example, in most cases the station keep location will not be known ahead of time and the operator will need to specify the location when the SPN is started.



Watch “Using a Runtime Value”: This video will show you how to have the value of an output event’s field be filled out at runtime.

Job 8-2: Verify →RobotGotoLocation output event’s location field is not defined

For output events, there are 3 ways to assign a value (also called a definition) to a field.

1- Assign it a value when developing the SPN

2- Assign it a value when starting the SPN

3- Use the value in another input or output event’s field

If we want an output event’s field to use the same value another event’s field is using, we use a variable.

A variable consists of a type, name, and a value, similar to an event’s field. Initially a variable has no value.

Variables can either be written to or read from:

- Write: Set the value of the variable
- Read: Get the value of the variable

For each field whose value we want to use elsewhere in the plan, we tell it to write to a variable name.

For each output event field we want to use that written value, we tell it to read from that variable name.

There are 2 situations where we would tell an output event’s field to write to a variable:

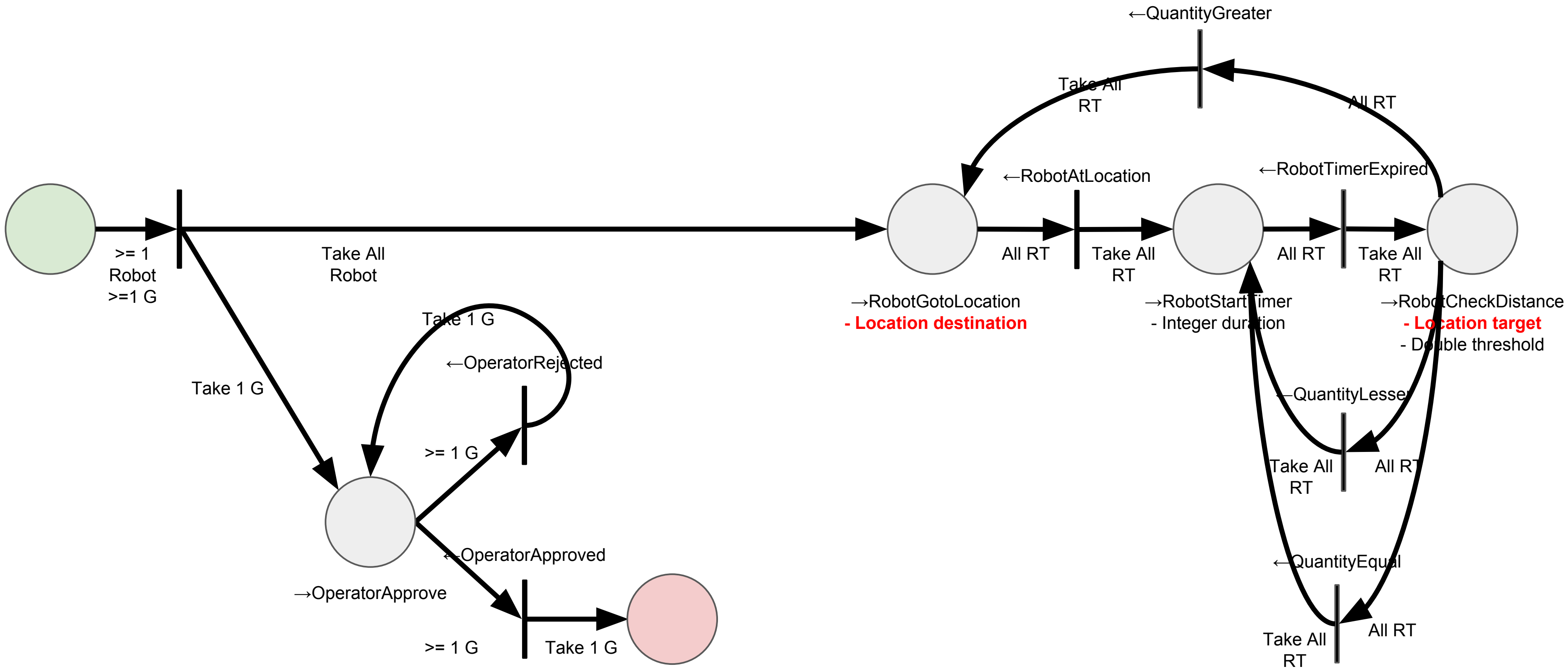
1- A field’s value is assigned when the SPN is developed and we want to use that same value in other fields

We could assign the desired value to each of these fields, but if we decide to change the value later, we have to change the value for each field.

Instead, we can assign the value to just one of these fields and also tell it to write to a variable. Then we tell each of the other fields to read from that variable name.

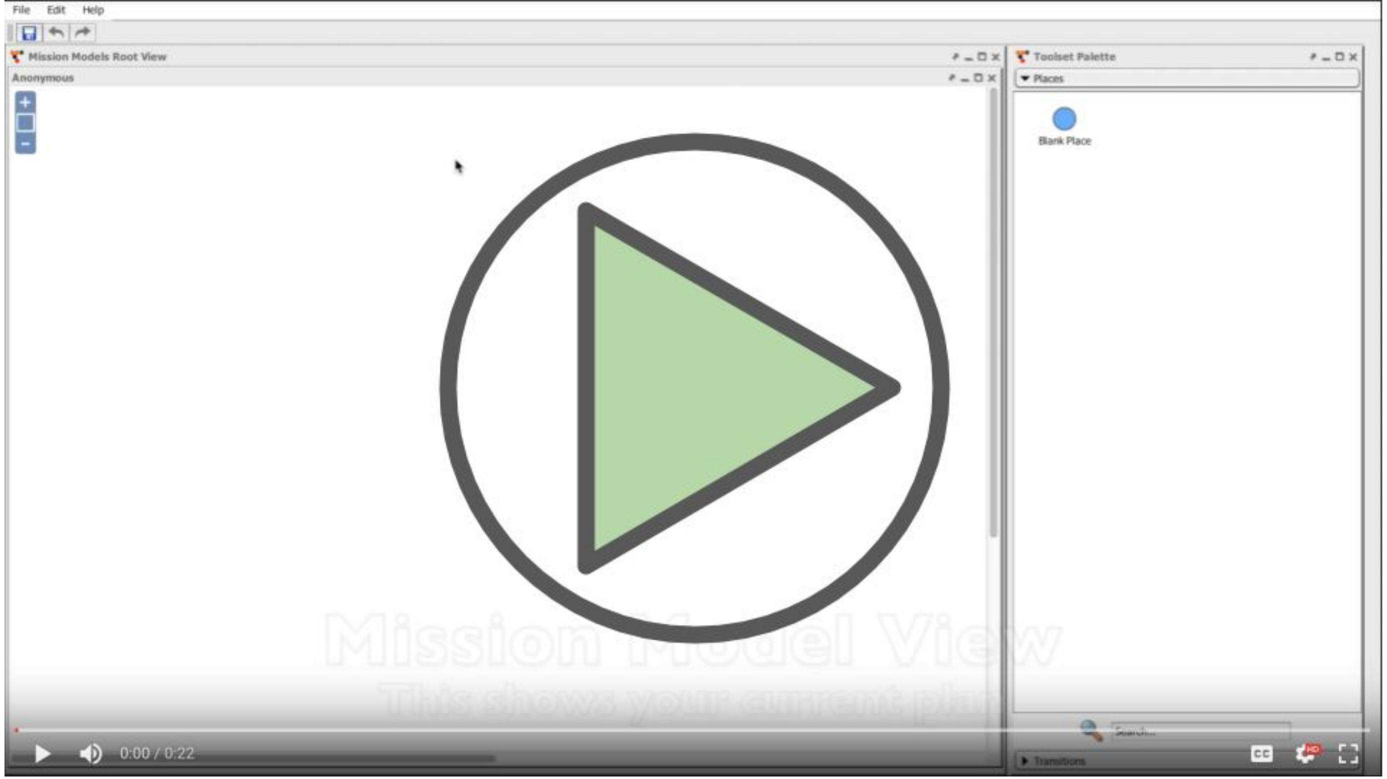
This way, if we want to change the value in the future, we just have to change it in one place.

In our station keeping SPN, the operator has to specify the station keeping location twice: once for →RobotGotoPoint, and a second time for →RobotCompareDistance.



Consider a scenario where the station keep location is known when the plan is developed. To have the two use the same location, we would do the following

1- Define the value in the →RobotGotoLocation output event’s Location destination field
2- Specify a variable name for that value to be written to
3- Have the →RobotCheckDistance output event’s Location target read from that variable



Watch “Saving a Static Value”: This video will show you how to define the value of an output event’s field and save it to a variable in DREAMM (steps 1 and 2, but not 3).

Job 8-3: Define the →RobotGotoLocation output event’s Location destination field and have its value be saved to the variable “destination”

There are 2 situations where we would tell an output event’s field to write to a variable:

1- The field’s value is assigned when the SPN is developed and we want to also use that value elsewhere

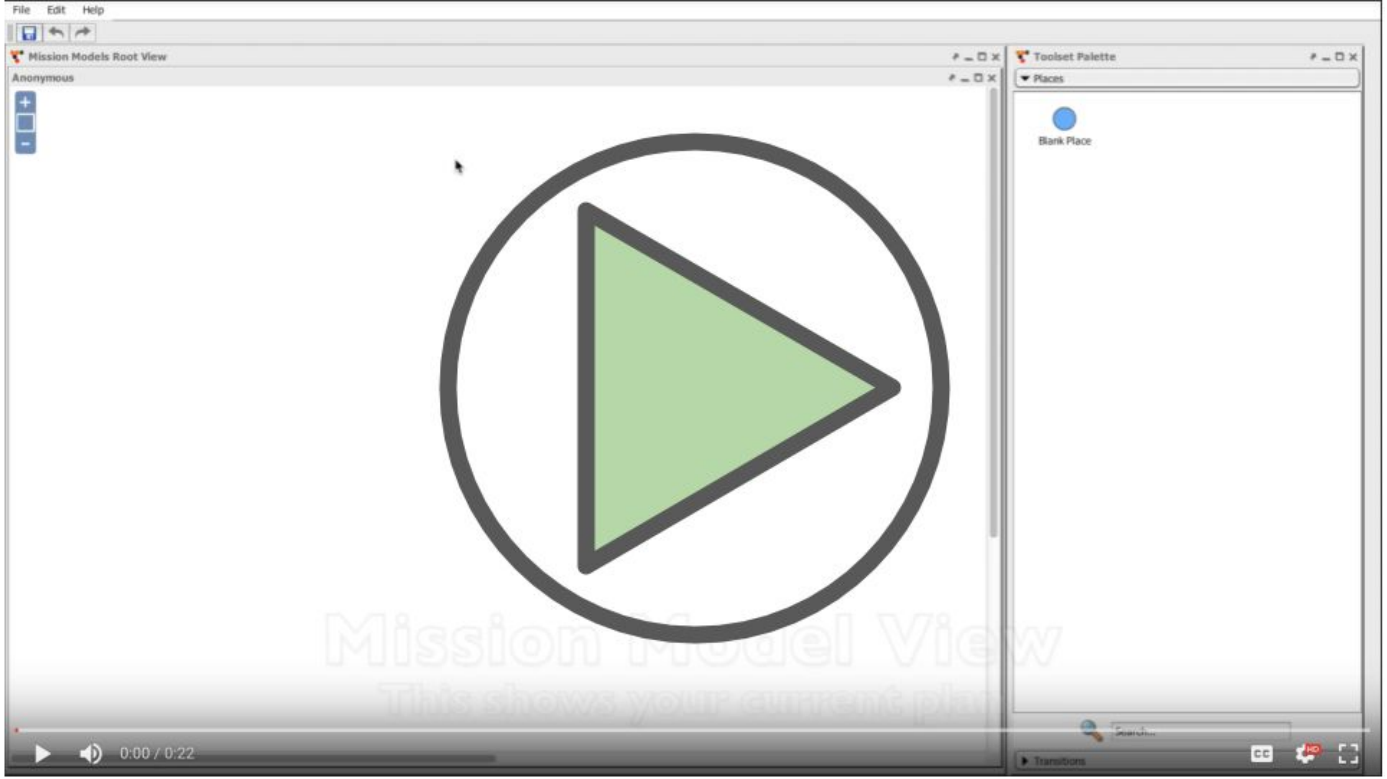
2- The field’s value is assigned when the SPN is started and we want to also use that value elsewhere

We could have the operator assign the value for each of these fields when the SPN is started, but they may accidentally assign two different values.

Instead, we can tell one of these fields to write the value it will receive to a variable and tell the other fields to read from that variable.

Consider a scenario where the station keep location is known when the plan is developed. To have the two use the same location, we would do the following

1- Leave the value in the →RobotGotoLocation output event's Location destination field blank so it is defined at startup
2- Specify a variable name for the value to be written to when it is received
3- Have the →RobotCheckDistance output event's Location target read from that variable



Watch “Saving a Runtime Value”: This video will show you how to save an output event field which will be defined at startup to a variable in DREAMM (steps 1 and 2, but not 3).

Job 8-4: Make the →RobotGotoLocation output event’s Location destination field be defined at startup and have its value be saved to the variable “destination”

Currently we tell each robot to move to the same location, so they will constantly hit each other trying to move to the same spot. We can use an AI service to compute a unique spot near a provided location for each robot, then tell each robot to move to their unique location to avoid collisions.

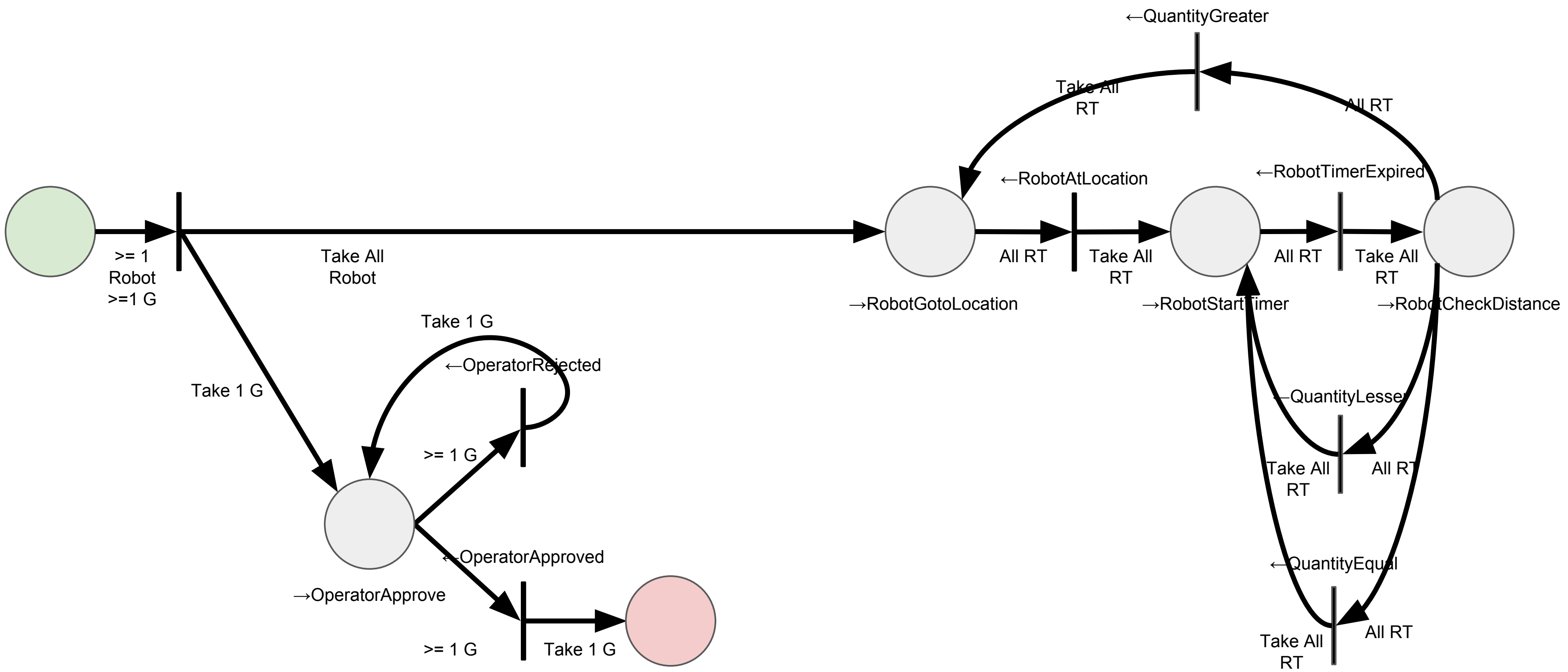
AI services use a library of algorithms to perform computations that would be difficult and/or time consuming for a human operator to perform at run time.

To have an AI to compute a unique spot, we use the following events:

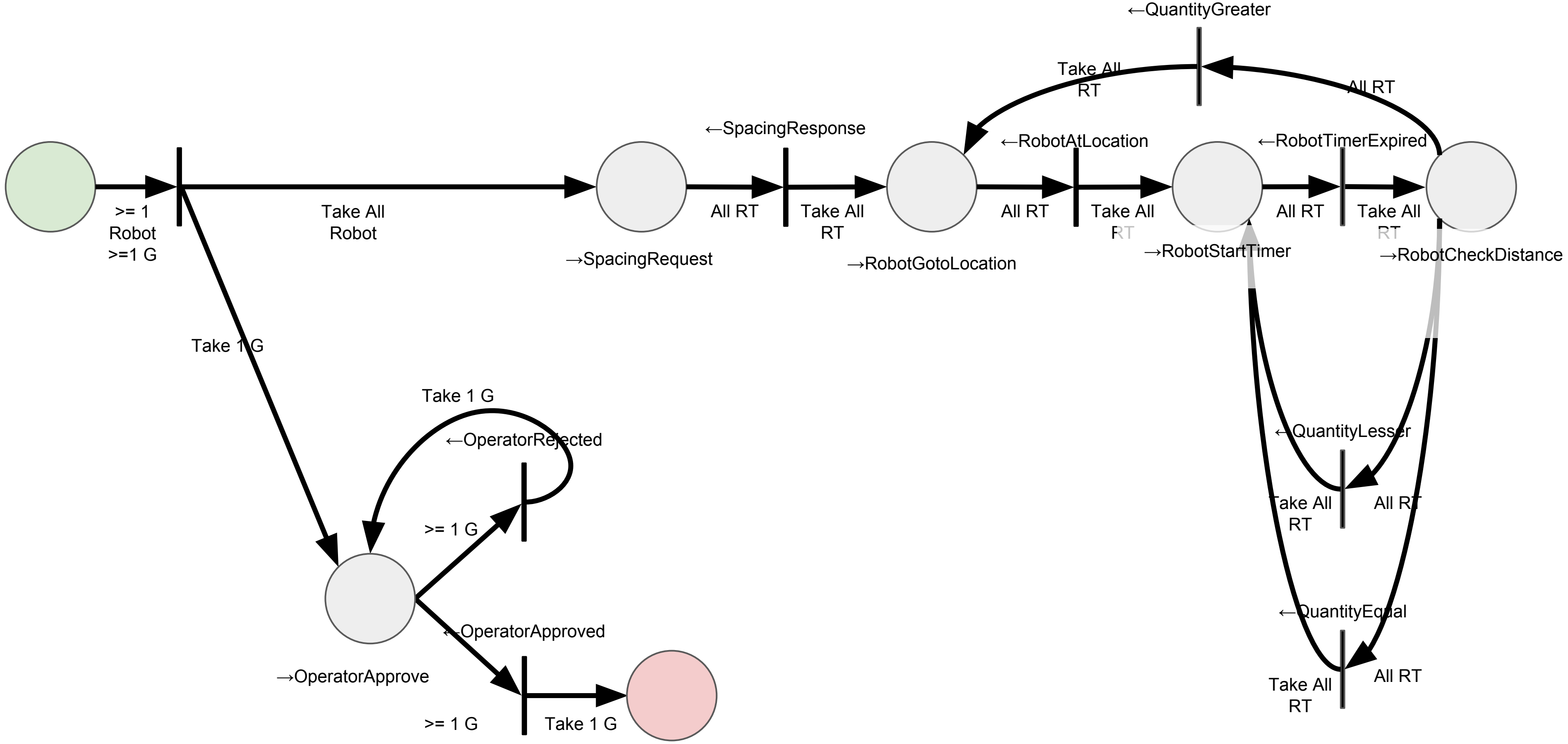
→SpacingRequest (Output event): Tells the AI to compute a unique location near a specified location for each robot token used to activate the event

←SpacingResponse (Input event): Returns the unique locations for the robots

QUIZ 8-1: Create and/or select the place and transition for →SpacingRequest and ←SpacingResponse



Quiz 8-1 Solution



Let’s look at the fields for the new events:

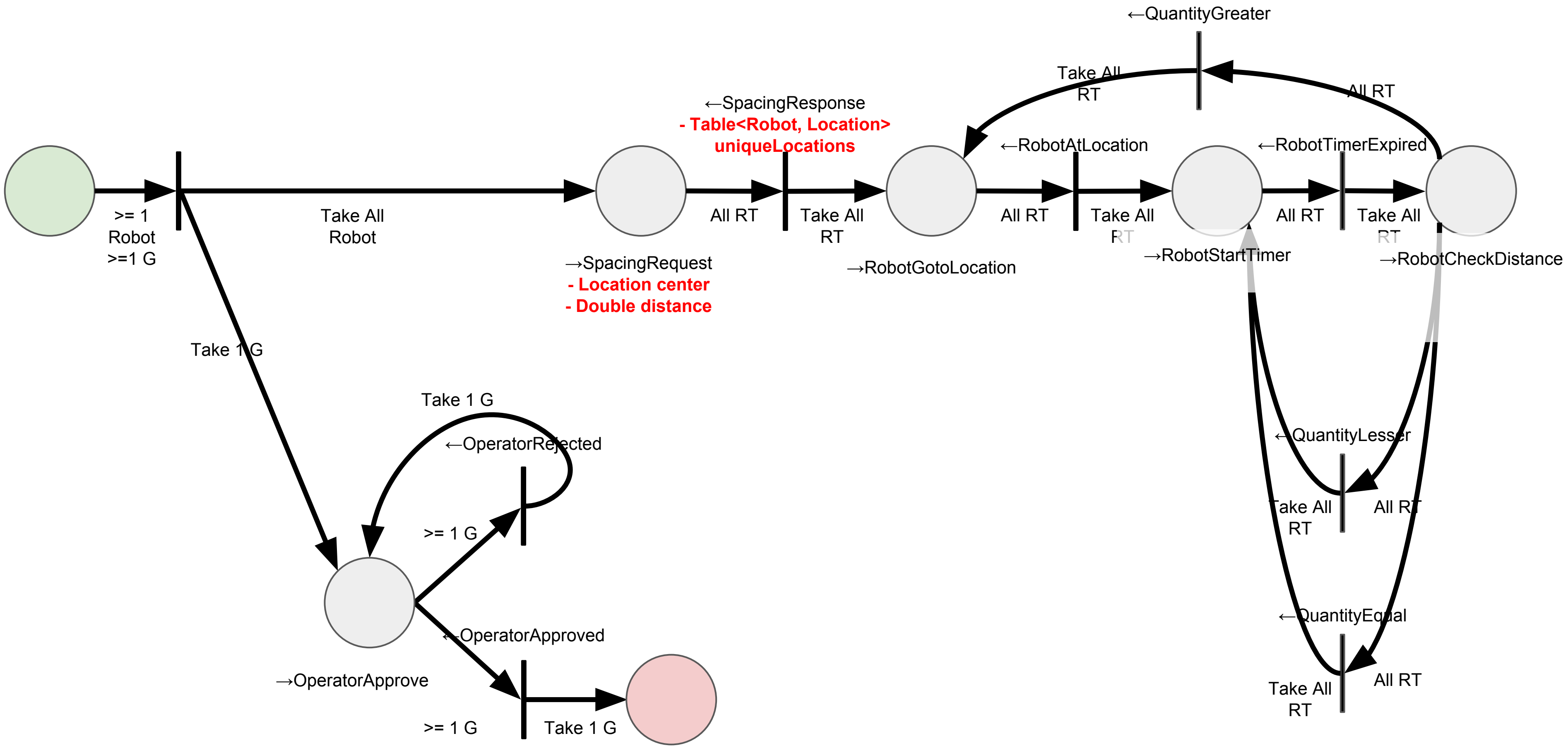
→SpacingRequest (Output event): Tells the AI service to compute a unique location near a specified location for each robot token used to activate the event

- Location center: The location that the robots’ unique locations should be centered around
- Double distance: The minimum distance between any two robots’ unique locations (in meters)

←SpacingResponse (Input event): Returns the unique locations for the robots

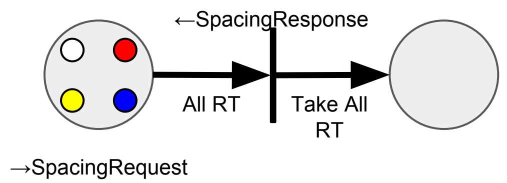
- Table<Robot, Location> uniqueLocations: A table of values: for each robot used to activate the SpacingRequest, the table has a row for that robot and its unique location.

←SpacingResponse lists the robot tokens used to activate the →SpacingRequest as Relevant Tokens.



←QuantityGreater
Take All RT
All RT
←SpacingResponse
- Table<Robot, Location>
uniqueLocations
←RobotAtLocation
←RobotTimerExpired
>= 1 Robot >=1 G
Take All Robot
All RT
Take All RT
All RT
Take All RT
All RT
Take All RT
→SpacingRequest
- Location center
- Double distance
→RobotGotoLocation
→RobotStartTimer
→RobotCheckDistance
Take 1 G
Take 1 G
←OperatorRejected
>= 1 G
←QuantityLesser
Take All RT
All RT
←OperatorApproved
→OperatorApprove
>= 1 G
Take 1 G
←QuantityEqual
Take All RT
All RT

QUIZ 8-2: What are the relevant tokens?



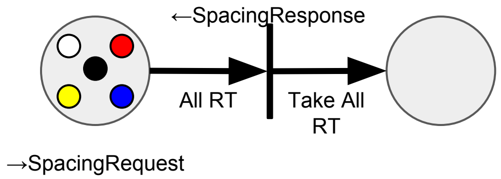
- Red token is Red robot’s token
- Yellow token is Yellow robot’s token
- Blue token is Blue robot’s token
- White token is White robot’s token

Quiz 8-2 Solution

The relevant tokens are

- Red robot’s token
- Yellow robot’s token
- Blue robot’s token
- White robot’s token

QUIZ 8-3: What are the relevant tokens?



- Red token is Red robot’s token
- Yellow token is Yellow robot’s token
- Blue token is Blue robot’s token
- White token is White robot’s token
- Black token is a generic token

Quiz 8-3 Solution

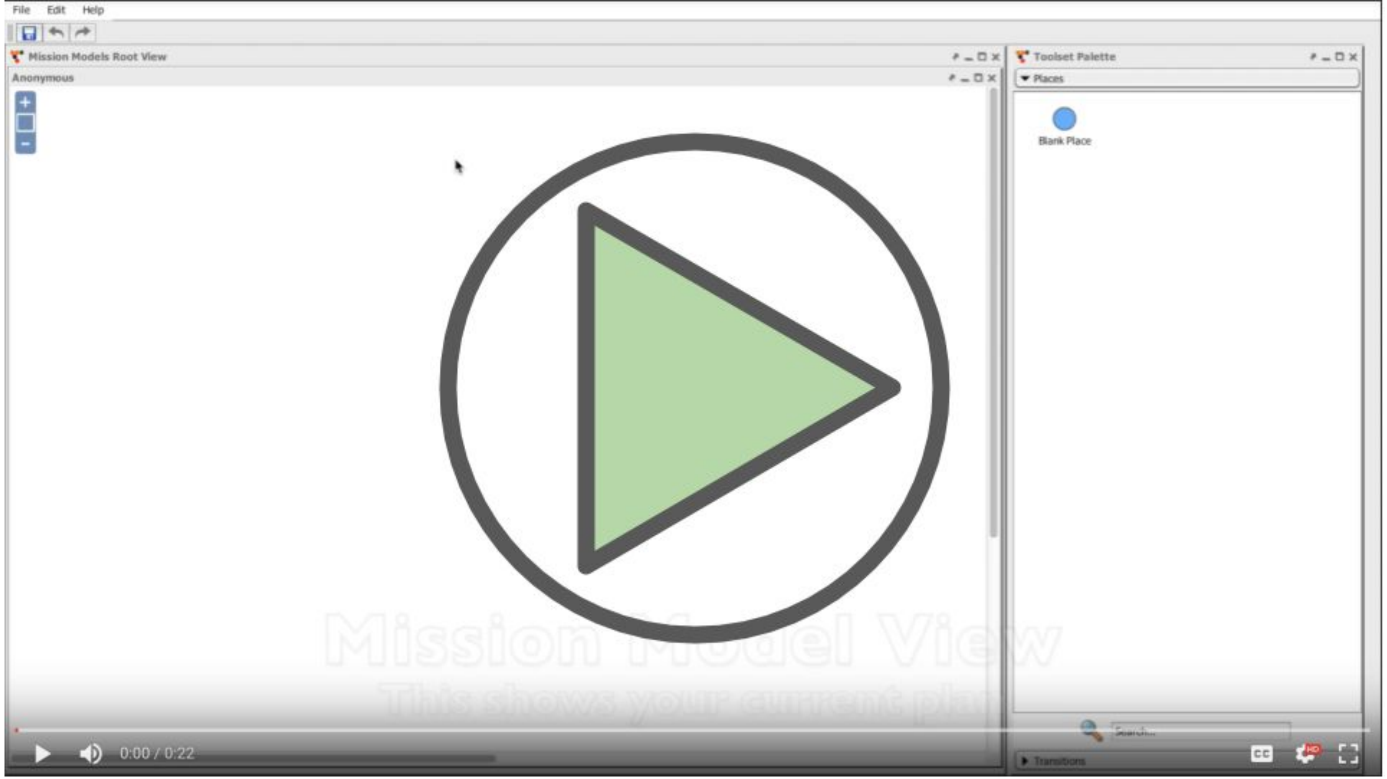
The relevant tokens are the same as before

- Red robot’s token
- Yellow robot’s token
- Blue robot’s token
- White robot’s token

The generic token is ignored by →SpacingRequest

Recall that, for input events, fields are pieces of information whose value will be provided by the operator, robot, or AI when the input event is generated. ←SpacingResponse has a field “Table<Robot, Location” which we can write to a variable.

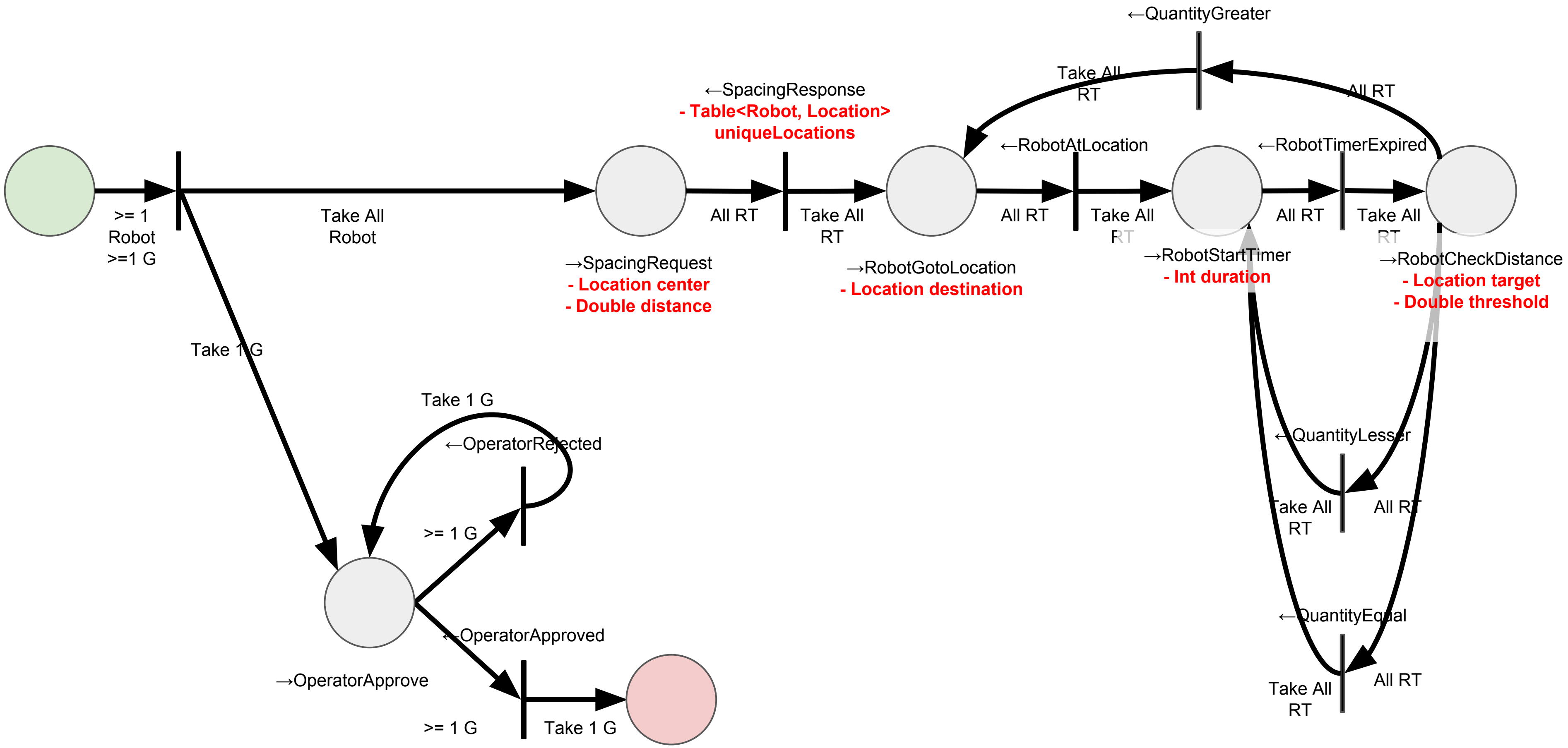
For each field in an input event, we can specify a variable name it should write its value to. Any output event fields which want to use this value can then reference the variable.



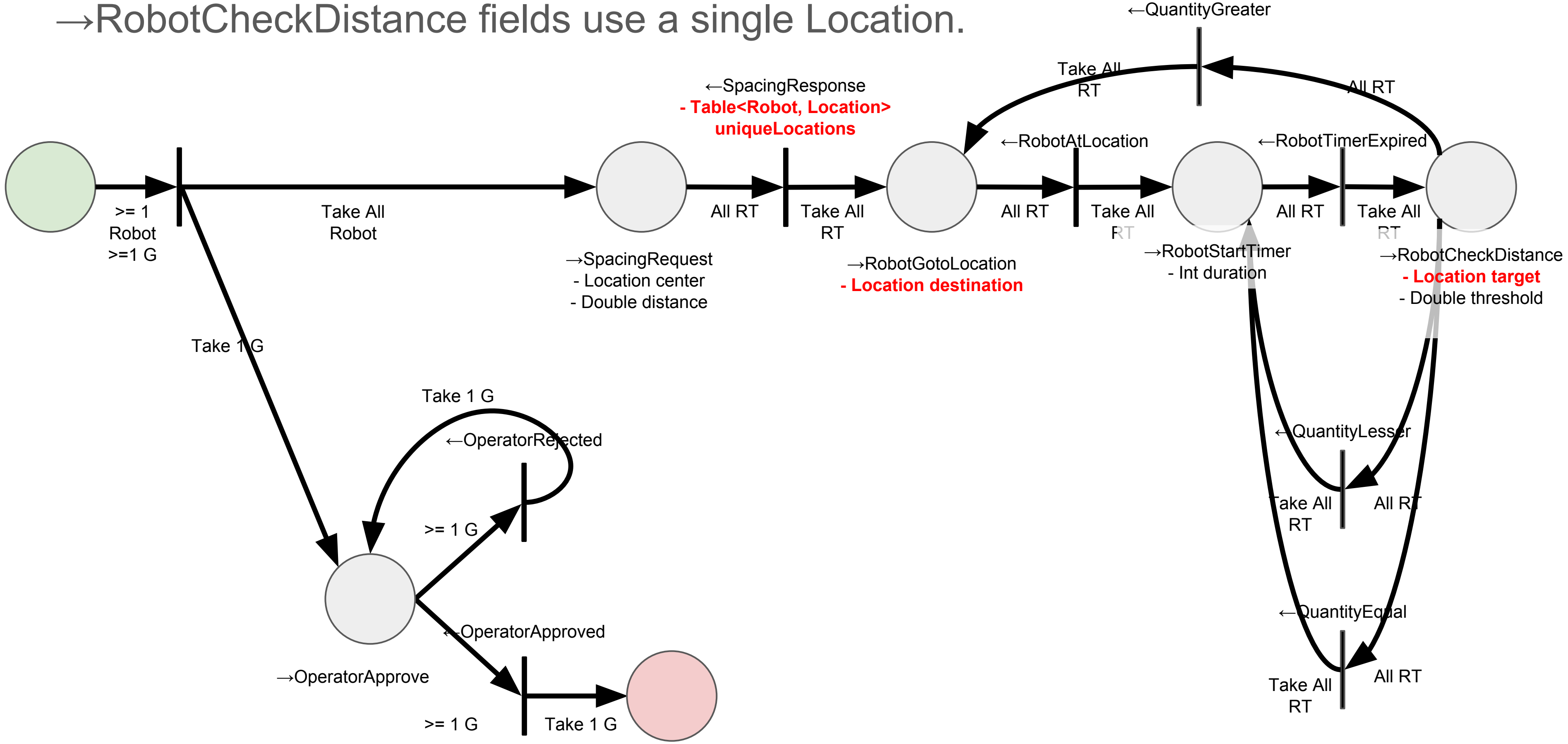
Watch “Saving an Input Event Value”: This video will show you how to specify the variable an input event’s field’s value should be written to.

Job 8-5: Make the ←SpacingResponse input event’s Table<Robot, Location uniqueLocations field write its value to the variable “spacedLocations”

Now we want to have the proxies move to their unique location and compute their distance to their unique location.



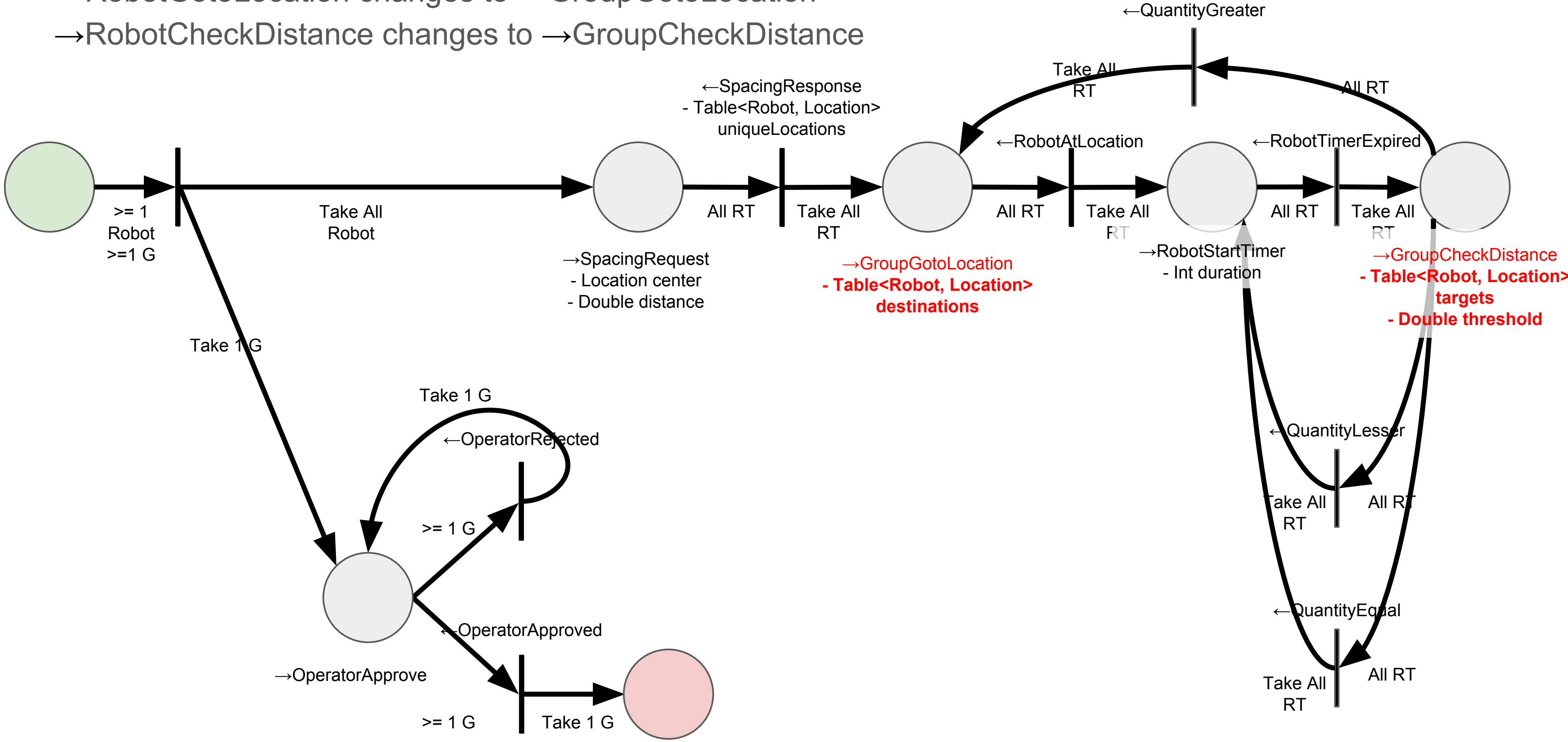
However, these types do not match. The ←SpacingReponse uniqueLocations field returns a Table<Robot, Location>, but the →RobotGotoLocation and →RobotCheckDistance fields use a single Location.



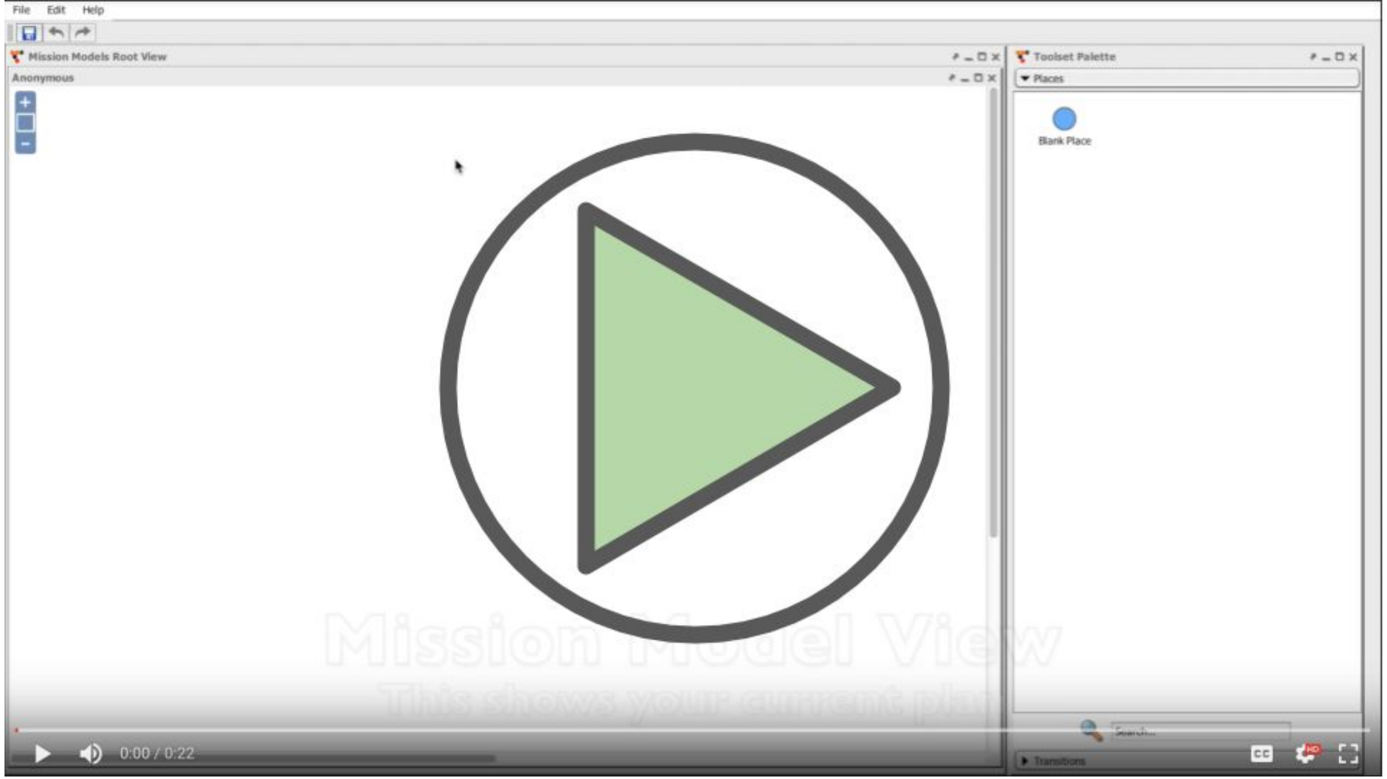
We solve this by changing output events which use a single Location to versions which instead uses a Table<Robot, Location>.

→RobotGotoLocation changes to →GroupGotoLocation

→RobotCheckDistance changes to →GroupCheckDistance



Now we can have the →GroupGotoLocation destinations field and →GroupCompareDistance targets field read from the variable we have ←SpacingResponse uniqueLocations write to.



Watch “Reading a Value from a Variable”: This video will show you how to have an output event field use the value from an variable.

Job 8-6: Have the →GroupGotoLocation destinations field and →GroupCompareDistance targets field read the value from the variable ←SpacingResponse uniqueLocations writes to.