SPARK-PL: Multiple Spaces

Alexey Solovyev

Cordelia Ziraldo

Joyeeta Dutta-Moscato

Qi Mi

Abstract

In this tutorial, we'll show how to create models with multiple spaces.

Table of Contents

1. Creating multiple spaces

A usual SPARK model has one space. Moreover, it is required to have a space in each model. But it is also possible to have multiple spaces in one model.

There is always a default space in each SPARK model. This space is used whenever a space is required but no explicit reference to a space is given. For example, when new agents are created and it is not specified in which space they are to appear, then the default space is used. In SPARK-PL the default space is declared with 'space' keyword as usual. Example:

```
space StandardSpace -20 20 -30 30 true false
```

Additional spaces are created in a different way with special commands (not keywords). For each type of space there is a command which creates this type of space and adds it to a model. Now, there are two commands: 'add-standard-space' and 'add-grid-space', which create and add a standard space(continuous space) and a grid space respectively.

These commands have the same arguments, so consider one of them.

```
add-standard-space name x-min x-max y-min y-max wrap-x wrap-y
```

The first argument "name" is a name of a new space (a double quoted string). All spaces in a model should have different names and these names are used to get a reference to a space using 'get-space' command which takes one argument: name of a space. Note that the default space always has name "space" so it is prohibited to use this name for any additional space. Other arguments are standard space initialization arguments.

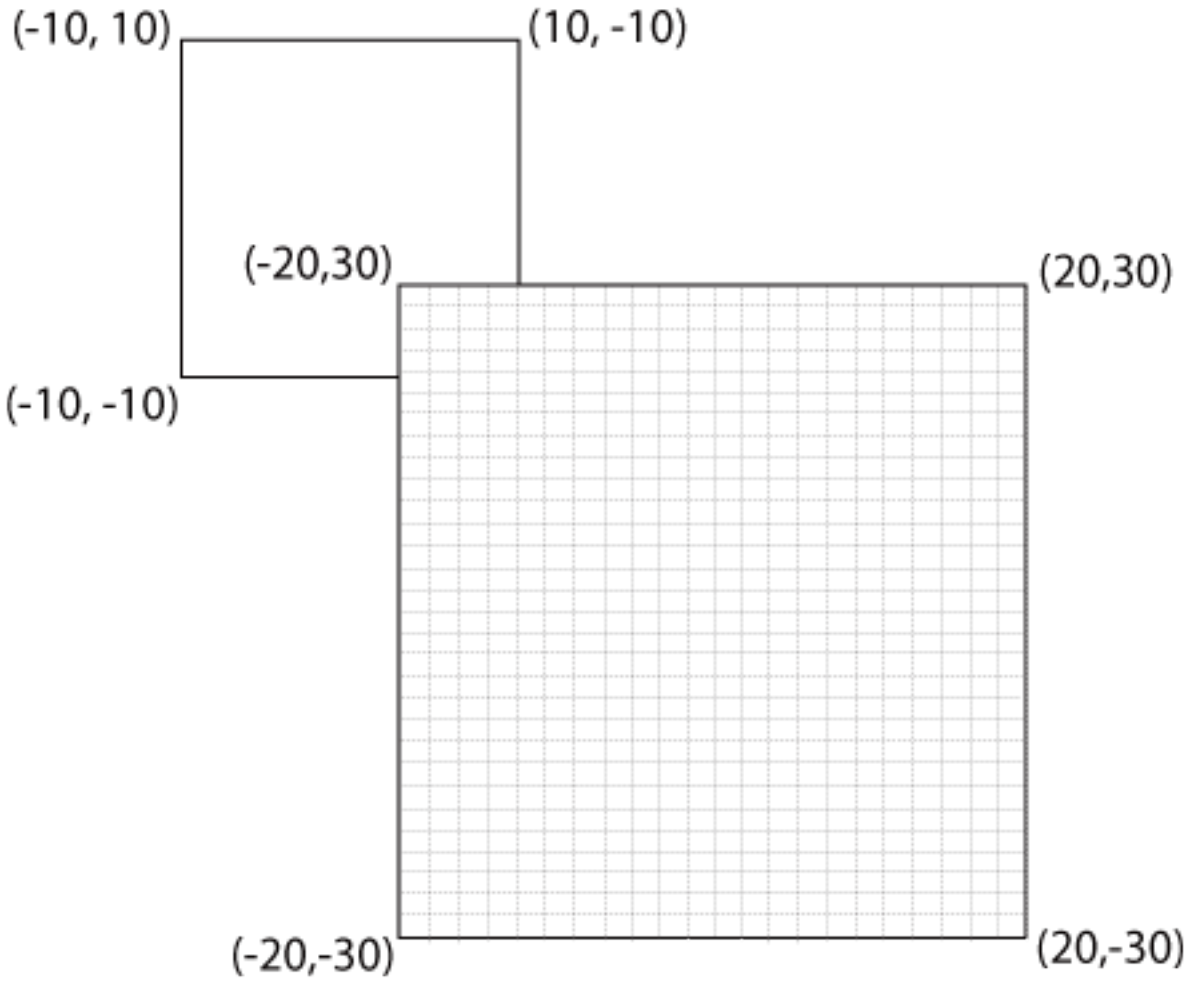
In SPARK-PL, it is possible to add a new space in any part of a model: as a global variable with initialization, inside 'setup' method, and even agents can add new spaces in their step functions. However it is more natural to add all spaces as global variables with initialization. Here is an example

```
model Model

; Default space
space StandardSpace -10 10 -10 10 true true

; Additional space
global space2 = add-grid-space "space2" -20 20 -30 30 false false
```

Figure 1.



In this example an additional grid space is created. Moreover, a reference to this space is stored in a global variable 'space2', so this space can be accessed from any part of a program using this global variable.

All spaces have type 'Space' and there are several methods inside this type. The basic methods are: 'x-min', 'x-max', 'y-min', 'y-max', 'x-size', 'y-size' which return the dimension of a space. Other methods will be described in the next sections.

2. Working with agents

Standard commands 'create' and 'create-one' create agents in the default space. In order to create agents in a specific space, use methods 'create' and 'create-one' of the 'Space' type. Consider an example:

```
to setup
 ; Instead of a local variable a global variable can be used
 var space2 = get-space "space2"

 space2.create Agent1 100

 ask space2.create-one Agent 2
 [
  set-random-position
 ]

 ; And the same commands written with 'ask' keyword
 ask space2
 [
  create Agent1 100
 ]

 ask space2
 [
  ask create-one Agent2
  [
   set-random-position
  ]
 ]
end
```

Figure 2.

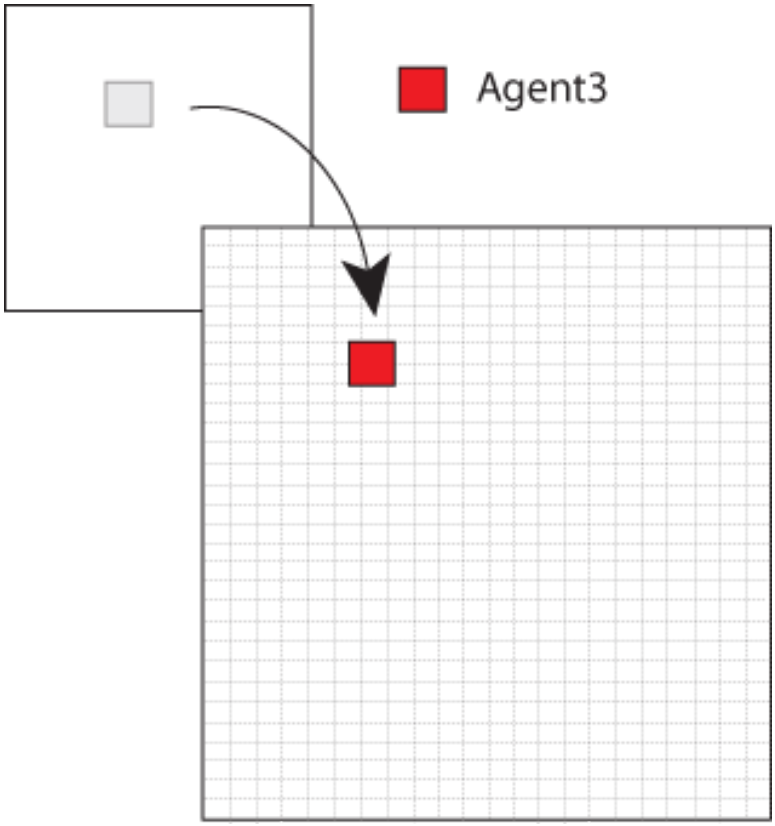
Be careful using the construction 'ask space [create something]'. Only space agents (and derived agents) can be created inside 'ask space' command.

Agents can be moved to other spaces. Use agent's 'move-to-space' method to do so. This method has two arguments: a space and a position in this space. After calling this method, an agent will be moved to a new space and at a given position inside this space. If a new space coincides with the space where an agent is located, then agent will only change its position. Example:

```
agent Agent3 : SpaceAgent

to jump-to-another-space
 ; Assume that 'space2' is a global variable
 ; for destination space
 move-to-space space2 position
end
```

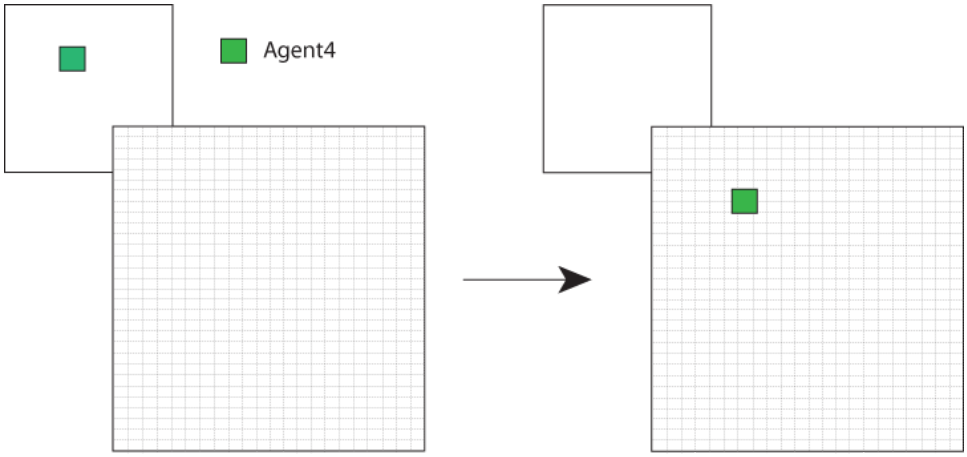
Figure 3.



This method can be also used for creating agents in different spaces: first create an agent in the default space and then move it to the destination space.

```
to setup
 ask create Agent4 100
 [
  move-to-space space2 position
 ]
end
```

Figure 4.



Space agents have 'hatch' and 'hatch-one' methods. These methods create new agents at the same position as a current agent. Note that this methods respect the space information so new agents will be created in the same space as a current agent.

Commands 'agents-here' and 'all-agents-here' also respect the space information. They return only agents located in the same space as a current agent. But the commands 'agents-at' and 'all-agents-at' always return agents from the default space. There are methods 'agents-at' and 'all-agents-at' in the 'Space' type which can be used to get agents at a specific location in a given space.

```
model Model

;Default space
space GridSpace -10 10 -10 10 true true

; Additional space of the same dimension as the default space
global space2 = add-standard-space "Additional Space"
                                   space-xmin space-xmax
                                   space-ymin space-ymax
                                   false false

to setup
 create Agent5 2

 ask space2
 [
  create-one Agent 5
 ]
end


agent Agent5 : SpaceAgent


to step
 ; each step 2 '2' and 1 '1' will be printed
 ; refers to agent's own space
```

```
 print count (agents-here Agent5)


 ; each step 3 '2' will be printed
 ; refers to only default space
 print count (agents-at Agent5 position 1)


 ; each step 3 '1' will be printed
 ;refers to space2
 print count (space2.agents-at Agent5 position 1)


 ; each step 2 '2' and 1 '1' will be printed
 ; refers to agent's own space
 print count (get-agent-space.agents-at Agent5 position 1)
```

end

Figure 5.

In this example a method 'get-agent-space' was used. This method returns a space in which a current agent is located.

3. Working with data layers

Each data layer is associated with one space. A usual 'create-grid' command creates a grid inside the default space. Also, globally defined data layers without explicit declaration are always created in the default space. In order to create a data layer inside a specific space, an explicit declaration with a command 'create-grid-in-space' is required. This command has four arguments: space in which a grid is created, name of a grid (should be the same as grid's variable name), x-dimension, y-dimension. Be aware that the space should be initialized before you use it in 'create-grid-command' (see example).

```
model Model


; The default space
space GridSpace -10 10 -10 10 true true

; Additional space
global space2 = add-standard-space "space2" (-10) 10 (-10) 10 true true

; Space which is not initialized yet
global space3

; Created in the default space
global data1 : grid

; Created in the default space
global data2 = create-grid "data2" 20 20

; Created in the additional space
global data3 = create-grid-in-space space2 "data3" 10 20

; It is prohibited to write
; global data4 = create-grid-in-space space3 "data4" 10 20
; though it will be translated and compiled without errors,
; but space 'space3' is not initialized yet.

to setup
 space3 = add-grid-space "space3"(-10) 10 (-10) 10 true true
end
```

Note that in order to create a data layer inside a specific space that space should be already initialized. Because of that, it is not possible to create data layers inside a space which is initialized inside the 'setup' method.

Methods 'value-here', 'add-value-here', 'set-value-here' work only for agents and data layers existing in the same space.

4. SPARK interface

In the visualization properties dialog you can select a space for visualization. All visible agents inside a selected space will be rendered. In the properties dialog all data layers are shown but only data layers in the selected space will be visualized.