SE Course 2021
Hackathon #2 - Goal-DSL

Description

GoalDSL is a Domain-Specific Language used for the evaluation of IoT applications and system behaviors, based on goal-driven design. The general idea is that we can define goal rules based on messages arrived at specific topics. For example, a goal may define a rule to wait until receiving an event or a message on a topic.

To expand this idea to the context of Cyber-Physical Systems, beyond topic-related goals, it is useful to be able to define environment-related goals for mobile things, such as robots. For example, in robotics it is common to require definition of goals related to the pose of the robot, or to follow a trajectory.

The DSL consists of three core modules/packages: a) the textual language (metamodel + grammar) for the definition of domain models, b) a code generator that performs an M2T transformation to produce the source code, and c) a library, namely Goalee that implements several concepts of GoalDSL in Python 3.5+ and it is used by the code generator to produce the source code from a given Goaldsl model. Goalee can also be used as a standalone Python library for implementing goal-driven targets for applications.

Below are the links to the aforementioned packages, which we will need to develop the tasks:.

- Language (Metamodel + Grammar): https://github.com/robotics-4-all/goal-dsl
- GoalGen: https://github.com/robotics-4-all/goal-gen
- Goalee: https://github.com/robotics-4-all/goalee

In the context of the current hackathon you are expected to develop software that validates the execution of IoT applications by defining a set of rules (Goals). Two environments are given, a smart garden and a smart house. For each environment a number of automations are expected to be evaluated using either GoalDSL or directly in Python using Goalee.

Submission

Upload a zip file on userreq (SE 2021 Project) that will contain one of the following, depending on the selected implementation framework / language (Goalee || GoalDSL):

- GoalDSL Model files
- Python source code (uses Goalee)

The name of the zip file must conform to the templates:

- *goaldsl_{name}-{surname}_{AEM}.zip* in case of using GoalDSL
- *goalee_{name}-{surname}_{AEM}.zip* in case of directly using Goalee for the implementation

Please don't forget to fill the Questionnaire!

Smart Garden

Environment Description

The Smart Garden is split into four (4) regions, namely RegionA, RegionB, RegionC and RegionD. Each region contains a number of sensors for monitoring temperature, humidity and air quality measurements, while they also contain windows that can be controlled (open/close) via a relay.

Furthermore, a robotic arm is installed in the garden that is capable of moving on the rails, as evident in the floor-plan of the smart garden below. The head of the robotic arm is equipped with a watering mechanism that can be controlled via a digital water valve, while it also has a mechanism to mix organics with water on demand.

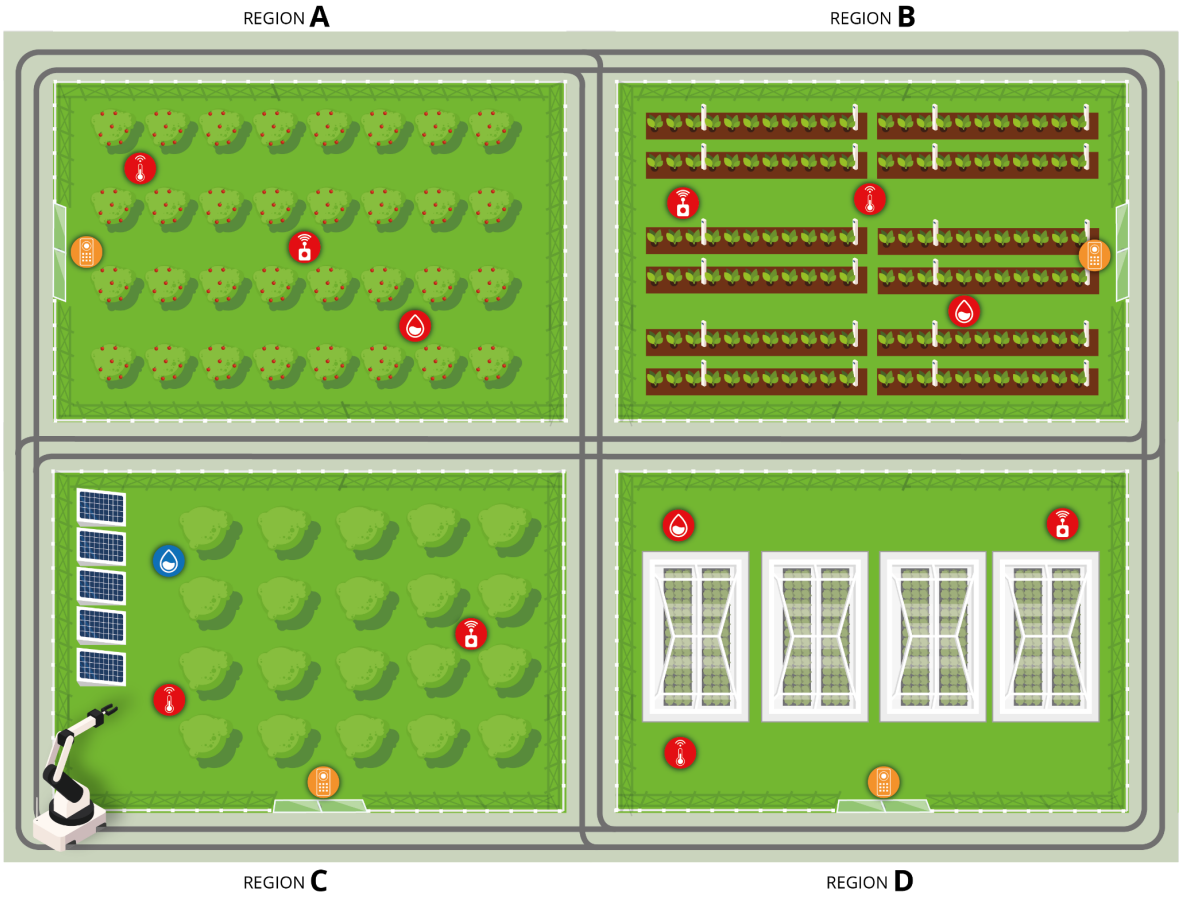

Fig. 1: Floor-Plan of the Smart Garden

Environment API

Topic URI	DataModel	Description
garden.robotic_arm.water_valve.open	{}	Open the water valve in the garden
garden.robotic_arm.water_valve.close	{}	Close the water valve in the garden
garden.robotic_arm.mix_organics	{}	Mix organics with water
garden.robotic_arm.base.pose	{	Publishes the pose of the base of the

	translation: {x: 0, y: 0, z: 0}, orientation: {x: 0, y: 0, z: 0} }	robotic arm with a frequency of 1Hz.

garden.temperature	{
	regionA: 0.0,
	regionB: 0.0,
	regionC: 0.0,
	regionD: 0.0,
	}
		Publishes temperature measurements from all Regions with a frequency of 1Hz

garden.humidity	{
	regionA: 0.0,
	regionB: 0.0,
	regionC: 0.0,
	regionD: 0.0,
	}
		Publishes humidity measurements from all Regions with a frequency of 1Hz

garden.air_quality	{
	regionA: 0.0,
	regionB: 0.0,
	regionC: 0.0,
	regionD: 0.0,
	}
		Publishes air-quality measurements from all Regions with a frequency of 1Hz

garden.regionA.window	{
	state: 0
	}
		Listens for relay state commands to open or to close the window in region A.

garden.regionB.window	{
	state: 0
	}
		Listens for relay state commands to open or to close the window in region B.

garden.regionC.window	{
	state: 0
	}
		Listens for relay state commands to open or to close the window in region C.

garden.regionD.window	{
	state: 0
	}
		Listens for relay state commands to open or to close the window in region D.

general.datetime	{
	Date: '15/06/2021',
	Time: '11:20'
	}
		Publishes datetime messages on a topic for general purpose usage.

Automations

Each automation must be defined using a single Target with GoalDSL.

Automation #1: Control environment quality of the in-door garden

Monitor the state of temperature, humidity and air-quality sensors for each of the 4 regions of the garden and:

- When either temperature is greater than 30 degrees or humidity is above 80% (0.8)
    - Open the relevant window
    - Wait for temperature to be less than 25 degrees (max = 30 minutes)
    - Wait for humidity to be less than 30% (0.3) (max = 30 minutes)
    - Close the window

- When the air-quality of a any region is less than 0.5 (50%)
    - Open all windows
    - Wait for air-quality to raise beyond 80% (0.8)
    - Close all windows

Evaluate the correctness of the automation using GoalDSL. Use the concurrent execution of Target Goals feature to support concurrent validation of both temperature and air-quality goals.

Automation #2: Water the in-door garden once a day

The robot must water the garden once a day. The path is pre-defined by a sequence of positions (in 2D space).

- Open the on-robot water valve
- Mix organics with water
- Move the robotic arm through the positions [(0, 0), (0, 10), (0, 20), (10, 20), (10, 10), (10, 0), (20, 0), (20, 10), (20, 20)]. The maximum time between each point is 10 minutes
- Close the water valve
- Park robotic arm at position (0, 0)

Smart Home

A smart home is composed of several "smart" physical objects, which collect information from the environment (sensors) or perform an action on it (effectors/actuators). Examples of such devices, commonly used in smart home environments, are temperature, humidity and camera sensors, robot assistants, relays, thermostats, lamps, speakers etc.

Environment Description

As evident from the floor-plan (Fig. 2), the house consists of 4 rooms and a wide corridor.Below is the table with the rooms and the installed physical objects on each room.

Room ID	Sensors	Effectors
Bathroom	Humidity, Temperature	Light, Humidifier, Thermostat
Kitchen	Humidity, Temperature, Gas, Camera	Light, Humidifier, Thermostat, Kitchen Relay
Livingroom	Humidity, Temperature, Camera	Light, Humidifier, Thermostat, Speaker, TV Relay
Bedroom	Humidity, Temperature, Camera	Light, Humidifier, Thermostat
Corridor		Alarm, Central Thermostat

Fig. 2: Floor-Plan of the Smart Home

Beyond sensors and actuators/effectors, a robot is installed in the smart home environment, which is capable of cleaning the floor on demand. Furthermore, the central thermostat that controls the heat of the water is located in the corridor. Finally, an alarm exists in the corridor that can be remotely controlled via the Application Interface (API) of the environment.

Environment API

Topic URI	DataModel	Description

{room_id}.temperature
 	
{room_id} = kitchen | livingroom | bedroom | bathroom
	{
	temperature: 0.0
	}
		Publishes temperature measurements with a frequency of 1Hz

{room_id}.humidity
 	
{room_id} = kitchen | livingroom | bedroom | bathroom
	{
	humidity: 0.0
	}
		Publishes humidity measurements with a frequency of 1Hz. Measurements are in range [0, 1] (percentage)

kitchen.gas	{
	gas: 0.0
	}
		Publishes gas measurements with a frequency of 1Hz. Measurements are in range [0, 1] (percentage)

{room_id}.camera	{ image: []	Publishes camera image frames with a frequency of 1Hz.

{room_id} = kitchen | livingroom | bedroom	}	

{room_id}.human_detected
 	
{room_id} = kitchen | livingroom | bedroom
	{
	position: (0, 0)
	}
		Publishes a message when a human is detected using the installed camera sensor. The message contains the position of the human in 2D space.

{room_id}.intruder_detected
 	
{room_id} = kitchen | livingroom | bedroom
	{
	position: (0, 0)
	}
		Publishes a message when a human intruder is detected using the installed camera sensor. The message contains the position of the intruder in 2D space.

{room_id}.light
 	
{room_id} = kitchen | livingroom | bedroom | bathroom
	{
	red: 0,
	green: 0,
	blue: 0,
	intensity: 0.0
	}
		Listens for commands to control smart lamps.
		Parameters {red, green, blue} are in range [0, 255] and intensity has a value in range [0, 1] (percentage)

{room_id}.humidifier
 	
{room_id} = kitchen | livingroom | bedroom | bathroom
	{
	state: 0
	}
		Listens for state commands to enable/disable the humidifier

{room_id}.thermostat
 	
{room_id} = kitchen | livingroom | bedroom | bathroom | corridor
	{
	state: 0
	}
		Listens for state commands to enable/disable the thermostat

kitchen.kitchen_relay	{
	state: 0
	}
		Listens for state commands to power on/off the kitchen

livingroom.tv_relay	{
	state: 0
	}
		Listens for state commands to power on/off the TV

livingroom.audio.speak	{
	text:''
	}
		Listens to speak commands. Input is a text that is internally transformed into speech and played via the installed speakers (internally performs text-to-speech)

livingroom.audio.play_music	{
	bytes: []
	}
		Music playback service. If a new message arrives before the last one completes, then the current is stopped and the new message is forwarded to the speaker.

livingroom.audio.play_from_youtube	{
	track_name: ''
	}
		Music playback service. Plays music directly from YouTube.

robot.pose	{
	translation: {x: 0, y: 0, z: 0},
	orientation: {x: 0, y: 0, z: 0}
	}
		Publishes the pose of the base of the robot with a frequency of 1Hz.

robot.on_enter	{
	room: ''''
	}
		Sends an event when a transition to a new room happens.
		For example, when the robot enters the kitchen, an event to this topic will be send

		with parameters: {room: “kitchen”}

alarm	{
	state: 0
	}
		Listens for control commands to enable or disable the in-house alarm.

Automations

Each automation must be defined using a single Target with GoalDSL.

Automation #1: Gas alert

In this automation the alarm is enabled in case the gas in the kitchen raises above 30%. Also, all thermostats must be closed, as long as all relays, cameras and humidifiers, the TV and the speaker.

Evaluate the correctness of the automation using GoalDSL.

Automation #2: In-house humidity control

In this automation we want to keep the humidity of all rooms in the range of [30-50]%. The algorithm must apply to all rooms independently.

Evaluate the correctness of the automation using GoalDSL. Use the concurrent execution of Target Goals feature to support all four rooms. Check for a duration of 10 minutes.

Automation #3: Clean the house using the robot

The robot must clean the floor of every room on a daily basis. The robot follows a predefined path and the order of the rooms should be the following:
LivingRoom -› Bedroom -› Kitchen -› Bathroom

Furthermore, the maximum time it takes for the robot to all rooms must be less than 30 minutes (t_overall ‹ 1800 seconds) and the time constraints for each room is given below:

Room ID	Minimum time (minutes)	Maximum time (minutes)
LivingRoom	5	10
Bedroom	2	5
Kitchen	6	11
Bathroom	1	4

Evaluate the correctness of the automation using GoalDSL.

Automation #4: Keep stable room temperature

In this automation we want to keep the temperature of rooms with human presence stable at 28 Celcius. The algorithm must apply to all rooms independently. If the temperature is dropped below 5% of the target, then the thermostat must open, while a raise of 5% causes deactivation of the thermostat.

Evaluate the correctness of the automation using GoalDSL. Use the concurrent execution of Target Goals feature to support all four rooms. Check for three consecutive changes (any of: High-High-High, High-High-Low, High-Low-Low, High-Low-High, Low-Low-Low, Low-Low-High, Low-High-High, Low-High-Low).

Automation #5: Intruder Detection

In this automation we want to implement an algorithm that follows the below procedures when an intruder is detected in the house. The intruder is detected using a backend algorithm that uses camera image frames.

1. Enables the alarm that is installed in the corridor
2. Opens all lights in the house
3. Force blinking the light in the room where the intruder was detected
4. Send the robot in the room where the intruder was detected. The robot has an installed camera that can be used to capture images of the intruder and send them directly to the cloud. The maximum time of reaction must be less than 5 seconds, so the robot must enable flash-mode to reach this time constraint.
5. Open the TV in the living room
6. Start playing "Seek & Destroy" in the living room.

Evaluate the correctness of the automation using GoalDSL.

APPENDIX A

Icons shown in the below image are used in this document to represent several sensors and effectors in smart environments.