DOCUMENTATION FOR jSimEnvironment

The jSimEnvironment is a microscopic traffic simulation environment based on the open source MOUTS traffic simulation package (http://homepage.tudelft.nl/05a3n/).

JsimEnvironment is a heavily modified version of MOTUS however, it shares many of the original features of MOTUS. For documentation on how to create KML files , the formatting of KML files, detector input files, etc. please refer to the original MOTUS documentation.

This environment has been connected with the GOAL programming language using EIS-0.3. This document provides the specifications for the actions and percepts that GOAL agents can use and also provides some information regarding the various settings with which the environment can be launched.

ACTION SPECIFICATIONS

IMPORTANT NOTE:
All agents have to preform an action in each step of the simulation because the environment expects it. If the agent really has nothing to perform it should perform the SKIP action.

Accelerate

- *Predicate*: accelerate(<DOUBLE>)
- *Description:* This action sets the acceleration value to be used in the next simulation step for the vehicle that the agent is controlling.
- *Parameters:* Inputs to this action are supposed to be real valued numbers. Positive numbers are interpreted as acceleration values and negative numbers as deceleration values. The value should be in m/s^2
- *Preconditions:* None
- *Post-conditions:* None
- *Additional Explanation:* The input value will automatically be bounded by the maximum deceleration of the agent's vehicle and its maximum acceleration.

Decelerate

- *Predicate:* decelerate(<DOUBLE>)
- *Description:* This action sets the deceleration value to be used in the next simulation step for the vehicle that the agent is controlling.
- *Parameters:* Inputs to this action are supposed to be Real valued numbers. Positive numbers are interpreted as deceleration values and negative numbers as acceleration values. The value should be in m/s^2
- *Preconditions*: None
- *Post-conditions*: None
- *Additional Explanation:* As with the acceleration action the values of the input will automatically be bounded by the maximum acceleration/deceleration of the agent's vehicle.

Change Lane

- *Predicate:* change_lane (<left | right>)
- *Description:* This actions causes a lane change maneuver to be started from the next simulation step. Currently this a lane change is set up to take 5 simulation steps to complete.
- *Parameters:* The only input parameter for this action is the direction in which the lane change has to be performed.
- *Input Values:* The possible values for the input parameter are: **left**, **right**. These should work if written in any case unless it clashes with Prolog standards (i.e. lEFT, rIGHT should also work).
- *Preconditions*: None. Could be set to checking if lane change is allowed but this is not explicitly checked for in the source code.
- *Post-conditions*: None
- *Additional Explanation*: This action is currently set up to be a durative action. It will take some time for an agent to perform the lane change operation. This durative action is a BETA feature that I was working on and probably requires further testing. The durative lane change action can be used in combination with the “Abort Lane Change” action. It should be noted that the standard in most micro-simulation software is to do an instantaneous lane change instead of a durative lane change. The durative lane change also affects the perceptual information provided to the agents (see the “Gap” percept for example).

Abort Lane Change

- *Predicate: abort_lane_change( )*
- *Description:* This cation causes the vehicle to instantaneously abort a lane change maneuver. This will reset the agent's vehicle lane to its original lane in the next simulation step.
- *Parameters:* None
- *Preconditions:* None. This could be set to checking if the vehicle is currently in the process of changing lane.
- *Post-conditions:* None
- *Additional Explanation:* This action is only useful if the lane changing action is set up to be durative which is currently the case.

Skip

- *Predicate:* skip( )
- *Description:* This action causes the agent to perform no new action. This is symbolic for repeating the same action as before. It can also be used to model delays or slow reaction times.
- *Parameters:* None
- *Preconditions*: None
- *Post-conditions*: None
- *Additional Explanation:* It is important to note that in every time step the simulation environments expects all agents to perform an action otherwise the simulation will not proceed to the next step. Therefore if an agents really has nothing to do it can perform the skip action in order to allow the simulation to proceed.

PERCEPT SPECIFICATIONS

Time

- *Predicate: sim_time (<DOUBLE>)*
- *Description:* This percepts gives the current simulation time in *seconds* as a Real valued number.
- *Type:* This percepts is always updated and provided to the agent.
- *Values:* The value returned by the percepts is the number of seconds that have passed since the beginning of the simulation. A value like 1.25 indicates that 1.25 seconds have passed.
- *Additional Explanation:* This percepts can be used in combination with the “Time Step” percept in order to calculate how many steps have been executed.

Time Step

- *Predicate:* sim_time_step (<DOUBLE>)
- *Description:* This percept provides the size of every simulation step in seconds. This can be set as a parameter.
- *Type:* This percept is only provided once because it will not change once the environment has been launched.
- *Values:* The value returned by the percept is the duration of each simulation step in *seconds*. The built-in value for this is currently set to 0.5 second. This can be changed by setting the appropriate environment setting parameter.
- *Additional Explanation:* Using this in combination with the simulation time, one can determine how many time steps have passed in the simulation.

Speed

- *Predicate:* speed (<DOUBLE>)
- *Description:* This percepts provides the current speed of the vehicle that the agent is controlling. The value is in m/s
- *Type:* This value is provided every time that it has changed.
- *Values:* Speed values are formatted according to the decimal format 'n.nn'
- *Additional Explanation:*

Gap

- *Predicate:* gap (<DOUBLE>)
- *Description:* This percept provides the distance in m to the closest obstruction.
- *Type:* The gap is provided every time that it has changed
- *Values:* Gap values are positive real numbers formatted according to the decimal format 'n.nn'
- *Additional Explanation: Obstructions* include vehicles moving in front of the agent's vehicle, other vehicles changing their lane in front of the agent's vehicle, end of a merge lane, vehicles moving in front of the vehicle on target adjacent lane when the agent it self is changing lane, etc. The gap calculation will always return the minimum most relevant gap information on which the agent should react.

Time Gap

- *Predicate:* time_gap (<DOUBLE>)
- *Description:* This percept provides the amount of time left in seconds to reach the closest obstruction as calculated by the gap percept.
- *Type:* The time gap is provided every time that it has changed
- *Values:* Time gap values are postivie real numbers formatted according to the decimal format 'n.nn'
- *Additional Explanation:* For the definition of relevant obstructions see the additional explanations for the gap percept.

Acceleration

- *Predicate:* acceleration (<DOUBLE>)
- *Description:* Current acceleration value of the vehicle in m/s^2
- *Type:* This percept is every time that it has changed in the environment.
- *Values:* Acceleration values are Real values numbers bounded by the agent's vehicle maximum deceleration and maximum acceleration.
- *Additional Explanation:*

Max Acceleration

- *Predicate:* max_acceleration (<DOUBLE>)
- *Description:* This percepts returns the maximum acceleration value in m/s^2 supported by the vehicle that the agent is controlling.
- *Type:* This percept is provided every time that it has changed
- *Values:* A positive real valued number.
- *Additional Explanation:* This value is currently hard coded to the value 2.

Max Deceleration

- *Predicate:* max_deceleration (<DOUBLE>)
- *Description:* This percept return the maximum deceleration value in m/s^2 supported by the vehicle that the agent is controlling.
- *Type:* This percept is provided every time that it has changed
- *Values:* A positive real values number which is the absolute value of the maximum deceleration.
- *Additional Explanation:* In general deceleration values are presented with negative numbers in the simulator, However, the value returned by this percept is the absolute value of the maximum deceleration.

Speed Delta

- *Predicate:* speed_delta (<DOUBLE>)
- *Description:* This percept returns the speed difference in m/s with a vehicle moving in front of the vehicle the agent is controlling.
- *Type:* This percept is provided every time that it has changed
- *Values:* Values for this percepts are Real valued numbers.

- *Additional Explanation:* Note that this percept only considers a vehicle moving in front of the agent's vehicle on the same lane. It does not consider other moving objects on adjacent lanes as some of the other percepts do.

Block

- *Predicate:* block (<INTEGER: 0 | 1 >)
- *Description:* This percept returns a 1 if a vehicle is somehow blocked by an obstruction and 0 otherwise.
- *Type:* This percept is provided every time that it has changed
- *Values:* The possible values are 0 as in the vehicle is not blocked and 1 as in the vehicle is blocked by an obstruction.
- *Additional Explanation:* Obstructions include another moving vehicle downstream of the agent's vehicle or reaching the end of a merge or subtracted lane. In addition if an agent is in the process of changing lane, the downstream vehicle on the target lane is also considered an obstruction.

Lane

- *Predicate:* lane (<STRING: ID>)
- *Description:* This percept provides the string id of the lane the agent's vehicle is currently located on
- *Type:* This percept is provided every time that it has changed
- *Values:* String representing the id of a lane.
- *Additional Explanation:* In the simulator lane IDs are generally numbers that are assigned in increasing order to the lane information that is parsed from the input KML map files.

Lane Speed Limit

- *Predicate:* lane_speed_limit (<DOUBLE>)
- *Description:* This percepts provides the speed limit for the current lane the agen'ts vehicle is positioned on. This speed limit is provided in m/s
- *Type:* This percept is provided every time that it has changed.
- *Values:* Values are positive Real numbers.
- *Additional Explanation:* Speed limit values are specified in km/h in the input KML map files.

Left Lane Change Allowed

- *Predicate:* left_lane_change_allowed (<Boolean>)
- *Description:* This percepts provides information about whether a lane change to the left is allowed at the current position of the agent's vehicle.
- *Type:* This percept is provided every time that it has changed.
- *Values:* True for when a lane change is allowed and false for when not.
- *Additional Explanation:*

Right Lane Change Allowed

- *Predicate:* right_lane_change_allowed (<Boolean>)
- *Description:* This percepts provides information about whether a lane change to the right is allowed at the current position of the agent's vehicle.
- *Type:* This percept is provided every time that it has changed.

- *Values:* True for when a lane change is allowed and false for when not.
- Additional Explanation:

Left Lane

- *Predicate:* left_lane (<STRING: ID >)
- *Description:* This percept provides the string id of the lane to left of the current vehicle's lane.
- *Type:* This percept is provided every time that it has changed
- *Values:* String representing the id of a lane.
- *Additional Explanation:* In the simulator lane IDs are generally numbers that are assigned in increasing order to the lane information that is parsed from the input KML map files.

Right Lane

- *Predicate:* right_lane (<STRING: ID>)
- *Description:* This percept provides the string id of the lane to right of the current vehicle's lane.
- *Type:* This percept is provided every time that it has changed
- *Values:* String representing the id of a lane.
- *Additional Explanation:* In the simulator lane IDs are generally numbers that are assigned in increasing order to the lane information that is parsed from the input KML map files.

Downstream Lane

- *Predicate:* downstream_lane (<STRING: ID>)
- *Description:* This percept provides the string id of the lane downstream of the current vehicle's lane.
- *Type:* This percept is provided every time that it has changed
- *Values:* String representing the id of a lane.
- *Additional Explanation:* In the simulator lane IDs are generally numbers that are assigned in increasing order to the lane information that is parsed from the input KML map files.

Lane Change In Progress

- *Predicate:* lane_change_in_progress (<Boolean>)
- *Description:* This percept provides information on whether a lane change action is currently in progress.
- *Type:* This percept is provided every time that it has changed.
- *Values:* True for the lane change is in progress and false otherwise
- *Additional Explanation:* Keep in mind that is is only useful in combination with the lane change action being a durative action. This is a beta feature and it is not common for micro-simulation to have durative lane changing.

On Route

- *Predicate:* on_route (<Boolean>)
- *Description:* This percepts provides information about whether the vehicle's route can be followed from its current lane position.

- *Type:* This percept is provided every time that it has changed.
- *Values:* True when the vehicle's destination is reachable from the current position and false otherwise.
- *Additional Explanation:*

Lane Change Required

- *Predicate:* require_lane_change (<Integer>)
- *Description:* This percept provides information about whether the agent needs to change a lane in order to be able to reach its desired destination.
- *Type:* This percept is provided every time that it has changed.
- *Values:* Integer numbers indicating the number of lane changes required to be able to reach destination.
- *Additional Explanation:* Positive numbers here indicate the number of lane changes to the left that are required to stay on the correct path to the destination, negative numbers indicate the number of lane changes to the right.

Left Lane Lead Gap

- *Predicate:* left_lane_lead_gap (<DOUBLE>)
- *Description:* This percepts provides information on the gap in left lane from the current position of the vehicle to a leading vehicle on the left lane. The units are in meters.
- *Type:* This percept is provided every time that it has changed.
- *Values:* Values are positive real numbers.
- *Additional Explanation:* This can be used in combination with the follow gap to determine the total gap to the left side of the vehicle.

Left Lane Follow Gap

- *Predicate:* left_lane_follow_gap (<DOUBLE>)
- *Description:* This percepts provides information on the gap in left lane from the current position of the vehicle to a following vehicle on the left lane. The units are in meters.
- *Type:* This percept is provided every time that it has changed.
- *Values:* Values are positive real numbers.
- Additional Explanation: This can be used in combination with the lead gap to determine the total gap to the left side of the vehicle.

Right Lane Lead Gap

- *Predicate:* right_lane_lead_gap (<DOUBLE>)
- *Description:* This percepts provides information on the gap in right lane from the current position of the vehicle to a leading vehicle on the right lane. The units are in meters.
- *Type:* This percept is provided every time that it has changed.
- *Values:* Values are positive real numbers.
- *Additional Explanation:* This can be used in combination with the follow gap to determine the total gap to the right side of the vehicle.

Right Lane Follow Gap

- *Predicate:* right_lane_follow_gap (<DOUBLE>)
- *Description:* This percepts provides information on the gap in right lane from the current

position of the vehicle to a following vehicle on the right lane. The units are in meters.

- *Type:* This percept is provided every time that it has changed.
- *Values:* Values are positive real numbers.
- Additional Explanation: This can be used in combination with the lead gap to determine the total gap to the right side of the vehicle.

Left Lane Change Imposed Acceleration

- *Predicate:* left_lc_imposed_acc (<DOUBLE>)
- *Description:* Returns the acceleration imposed on a potential follower if the agent made a left lane change. Units are in m/s^2 .
- *Type:* This percept is provided every time that it has changed.
- *Values:* Real numbers indicating the acceleration / deceleration that is imposed.
- *Additional Explanation:* This is an experimental feature and not tested. It is based on the MOBIL lane changing model.

Left Gap Acceptable

- *Predicate:* left_gap_acceptable (<Boolean>)
- *Description:* This percept provides information on whether the left gap is acceptable for a lane change according to the MOBIL lane change model.
- *Type:* This percept is provided every time that it has changed.
- *Values:* True if acceptable and false if not.
- *Additional Explanation:* This is an experimental feature based on the MOBIL model and not tested.

Left Lane Change Beneficial

- *Predicate:* left_lc_beneficial (<Boolean>)
- *Description:* This percept provides information on whether a lane change to the left is considered beneficial according to the MOBIL lane changin model incentive structure.
- *Type:* This percept is provided every time that it has changed.
- *Values:* True if considered beneficial and false otherwise
- *Additional Explanation:* This is an experimental feature based on the MOBIL model and not tested.

SIMULATION SETTINGS

Simulation settings can be used in the environment init parameters to change the behavior of the jSim simulation environment.

Basic Settings

Simulation Scenario

- *name:* scenario

- *Values:* Values can be one of the following: A16, MERGING, DENSITY_FLOW, DENSITY_FLOW_BLOCKED, MIXED_TRAFFIC, MIXED_TRAFFIC_BLOCKED, ACCELERATION, SCALABILITY.
- *Description:* This setting starts one of the predefined scenarios of the simulation environment.

Simulation Step Size

- *name:* stepSize
- *Values:* Positive real valued numbers
- *Description:* This setting detemines the size of each simulation step in seconds.

Simulation Duration

- *name:* duration
- *Values:* Positive Real values numbers
- *Description:* This setting detemines the total duration of the simulation.

Simulation Log Output Directory

- *name:* outputDir
- *Values:* *String representing a File System Path*
- *Description:* This value detemines the folder in which the simulation output files such as logs as written in.

Advanced Settings

Frame Recoding Format

- *name:* recordFormat
- *Values:* A String value “PDF” will cause the simulation frames to be recorded in PDF format, anything else will result in frames to be recorded in “PNG” format.
- *Description:* This an experimental feature for recoding the simulation frame by frame.

Debug

- *name:* debug
- *Values:* Boolean values
- *Description:* Causes the simulation to run in DEBUG mode. This will cause a lot of information to be printed to the console.

Debug Model

- *name:* debugModel
- *Values:* Boolean Values
- *Description:* This will cause the simulation to keep track of internal car-following and agent

actions and output them as logs. This is an experimental feature.

Debug Trajectory

- *name:* debugTrajectory
- *Values:* Boolean Values
- *Description:* This will cause the simulation environment to keep track of trajectory data for analysis and output trajectory data to the log output folder. This is an experimental feature.

Trajectory Period

- *name:* trajectoryPeriod
- *Values:* Real valued number in seconds
- *Description:* This setting determines the trajectory sampling rate in seconds. WARNING: Setting this to a low value will result in large memory usage!

Debug Detector

- *name:* debugDetector
- *Values:* Boolean Value
- *Description:* This setting will cause the simulation environment to keep track of detector data for analysis. The data will be outputted to the log folder.

Detector Period

- *name:* detectorPeriod
- *Values:* Real values number
- *Description:* This setting determines the detector data aggregation period in seconds.

Import Detector from KML

- *name:* importDetectorFromKML
- *Values:* Boolean value
- *Description:* This setting will cause the simulation to import detector info from an input file.

Detector Info File Path

- *name:* detectorInfoFilePath
- *Values:* Path to file
- *Description:* This setting determines the path to the detector info file.

Import Origin From KML

- *name:* importOriginFromKML
- *Values:* Boolean Value
- *Description:* This setting causes the simulation to import origin data (e.g. generated demand

values) from KML and additional files. Used in combination with detector info files.

More advanced settings are available in the source code. All of these settings are in the *BuiltinSettings.java* source file and can be used in the source code.

KNOWN ISSUES

- The simulator has to be relaunched every time that it is used. The reason is probably related to the use of static fields in combination with the singleton pattern to instantiate the simulator. Resetting the initial value of some static field every time the main method is executed should fix the issue.