KCL Internship

ROSPlan Esterel Plan Parsing

Oscar Lima

ISR: Institute for Systems and Robotics
LARSyS: Laboratory for Robotics and Engineering Systems
IST: Instituto Superior Tecnico, Lisboa Portugal

December 4, 2018

Content

- This slides explain KCL ROSPlan[1] Esterel plan compilation.
- ROSPlan is a bridge between AI planning and robotics.
- Esterel Plan Parse/Dispatch is the default ROSPlan execution layer.
- In a nutshell: Converts planner output into STN[2].
- The author of ROSPlan Esterel Dispatch is Michael Cashmore[3], my contribution to this work is limited and is covered at the end of this talk (interference edges optimization)

[1] https://github.com/KCL-Planning/ROSPlan

[2] Simple Temporal Network

[3] https://nms.kcl.ac.uk/michael.cashmore/index.html

Examples, use cases

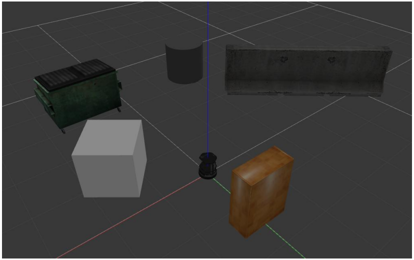
The following PDDL domains are used as examples in this talk:

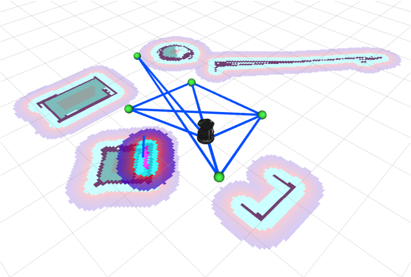
- turtlebot domain
- mapanalyzer domain
- rush to school

domains available under: `https://github.com/oscar-lima/rosplan_debug/tree/kinetic/rpd_pddl_examples/common/pddl`

turtlebot domain

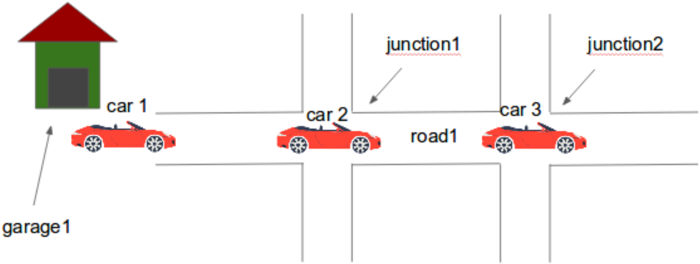
- objective: visit waypoints
- only 1 action: goto_waypoint (source, destination)
- click to open domain , problem





mapanalyzer domain

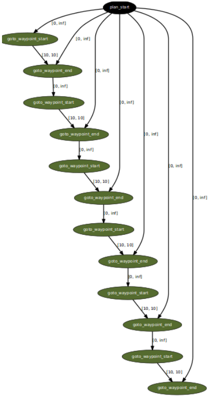
- objective: build roads efficiently to minimize travel time
- relevant actions: move_vehicle_road, build_road
- cars move between junctions
- duration gets computed as distance / speed
- NOTE: this problem is related with issue #140
  which provides with a nice benchmark for large plans



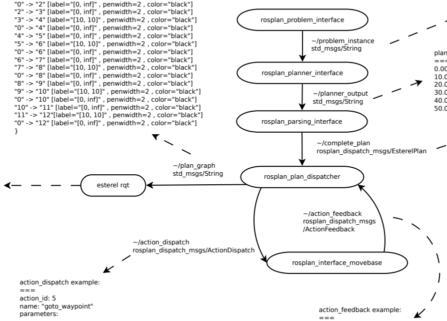
rush_to_school domain

- objective: Batdad takes Ben and Siena to school
- relevant actions: move_vehicle, get_in_car, get_down_from_car
- click to open domain , problem

Input - Output, all ROSPlan pipeline (turtlebot example)



```
plan_graph example:
===
digraph plan {
0[label="plan_start",style=filled,fillcolor=black,fontcolor=white];
1[label="goto_waypoint_start",style=filled,fillcolor=darkolivegreen,fontcolor=white];
2[label="goto_waypoint_end",style=filled,fillcolor=darkolivegreen,fontcolor=white];
3[label="goto_waypoint_start",style=filled,fillcolor=darkolivegreen,fontcolor=white];
4[label="goto_waypoint_end",style=filled,fillcolor=darkolivegreen,fontcolor=white];
5[label="goto_waypoint_start",style=filled,fillcolor=darkolivegreen,fontcolor=white];
6[label="goto_waypoint_end",style=filled,fillcolor=darkolivegreen,fontcolor=white];
7[label="goto_waypoint_start",style=filled,fillcolor=darkolivegreen,fontcolor=white];
8[label="goto_waypoint_end",style=filled,fillcolor=darkolivegreen,fontcolor=white];
9[label="goto_waypoint_start",style=filled,fillcolor=darkolivegreen,fontcolor=white];
10[label="goto_waypoint_end",style=filled,fillcolor=darkolivegreen,fontcolor=white];
11[label="goto_waypoint_start",style=filled,fillcolor=darkolivegreen,fontcolor=white];
12[label="goto_waypoint_end",style=filled,fillcolor=darkolivegreen,fontcolor=white];
"0" -> "1" [label="[0, inf]" , penwidth=2 , color="black"]
"1" -> "2" [label="[10, 10]" , penwidth=2 , color="black"]
"0" -> "2" [label="[0, inf]" , penwidth=2 , color="black"]
"2" -> "3" [label="[0, inf]" , penwidth=2 , color="black"]
"3" -> "4" [label="[10, 10]" , penwidth=2 , color="black"]
"0" -> "4" [label="[0, inf]" , penwidth=2 , color="black"]
"4" -> "5" [label="[0, inf]" , penwidth=2 , color="black"]
"5" -> "6" [label="[10, 10]" , penwidth=2 , color="black"]
"0" -> "6" [label="[0, inf]" , penwidth=2 , color="black"]
"6" -> "7" [label="[0, inf]" , penwidth=2 , color="black"]
"7" -> "8" [label="[10, 10]" , penwidth=2 , color="black"]
"0" -> "8" [label="[0, inf]" , penwidth=2 , color="black"]
"8" -> "9" [label="[0, inf]" , penwidth=2 , color="black"]
"9" -> "10" [label="[10, 10]" , penwidth=2 , color="black"]
"0" -> "10" [label="[0, inf]" , penwidth=2 , color="black"]
"10" -> "11" [label="[0, inf]" , penwidth=2 , color="black"]
"11" -> "12"[label="[10, 10]" , penwidth=2 , color="black"]
"0" -> "12" [label="[0, inf]" , penwidth=2 , color="black"]
}
```



```
planner output example:
===
0.000:  (goto_waypoint kenny wp0 wp0)  [10.000]
10.001: (goto_waypoint kenny wp0 wp1)  [10.000]
20.002: (goto_waypoint kenny wp1 wp2)  [10.000]
30.003: (goto_waypoint kenny wp2 wp3)  [10.000]
40.004: (goto_waypoint kenny wp3 wp4)  [10.000]
50.005: (goto_waypoint kenny wp4 wp5)  [10.000]
```

```
complete_plan example:
===
nodes:
  -
    node_type: 2
    node_id: 0
    name: "plan_start"
    action:
      action_id: 0
      name: ''
      parameters: []
      duration: 0.0
      dispatch_time: 0.0
    edges_out: [0, 2, 5, 8, 11, 14, 17]
    edges_in: []
  -
    node_type: 0
    node_id: 1
    name: "goto_waypoint_start"
    action:
      action_id: 0
      name: "goto_waypoint"
      parameters:
        -
          key: "v"
          value: "kenny"
        -
          key: "from"
          value: "wp0"
        -
          key: "to"
          value: "wp0"
      duration: 10.0
      dispatch_time: 0.0
    edges_out: [1]
    edges_in: [0]
  -
    node_type: 1
    node_id: 2
    name: "goto_waypoint_end"
```

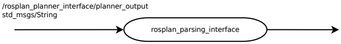
```
edges:
  -
    edge_id: 0
    edge_name: "edge_0"
    signal_type: 0
    source_ids: [0]
    sink_ids: [1]
    duration_lower_bound: 0.0
    duration_upper_bound: 1.797693134
  -
    edge_id: 1
    edge_name: "edge_1"
    signal_type: 0
    source_ids: [1]
    sink_ids: [2]
    duration_lower_bound: 10.0
    duration_upper_bound: 10.0
  -
    edge_id: 2
    edge_name: "edge_2"
    signal_type: 0
    source_ids: [0]
    sink_ids: [2]
    duration_lower_bound: 0.0
    duration_upper_bound: 1.797693134
  -
    edge_id: 3
    edge_name: "edge_3"
    signal_type: 0
    source_ids: [2]
    sink_ids: [3]
    duration_lower_bound: 0.0
    duration_upper_bound: 1.797693134
  -
    edge_id: 4
    edge_name: "edge_4"
    signal_type: 0
    source_ids: [3]
```

```
action_dispatch example:
===
action_id: 5
name: "goto_waypoint"
parameters:
  -
    key: "v"
    value: "kenny"
  -
    key: "from"
    value: "wp4"
  -
    key: "to"
    value: "wp5"
duration: 10.0
dispatch_time: 50.0050010681
```

```
action_feedback example:
===
action_id: 5
status: "action achieved"
information: []
```

Input - Output (esterel parsing only)

```
planner output example:
===
0.000:  (goto_waypoint kenny wp0 wp0)  [10.000]
10.001: (goto_waypoint kenny wp0 wp1)  [10.000]
20.002: (goto_waypoint kenny wp1 wp2)  [10.000]
30.003: (goto_waypoint kenny wp2 wp3)  [10.000]
40.004: (goto_waypoint kenny wp3 wp4)  [10.000]
50.005: (goto_waypoint kenny wp4 wp5)  [10.000]
```



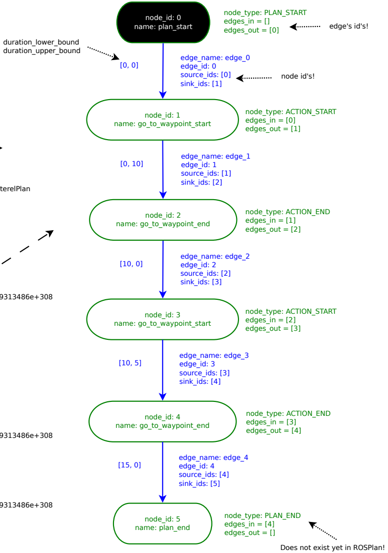
/rosplan_planner_interface/planner_output
std_msgs/String

rosplan_parsing_interface

~/complete_plan
rosplan_dispatch_msgs/EsterelPlan

```
complete_plan example:
===
nodes:
  -
    node_type: 2
    node_id: 0
    name: "plan_start"
    action:
      action_id: 0
      name: ''
      parameters: []
      duration: 0.0
      dispatch_time: 0.0
    edges_out: [0, 2, 5, 8, 11, 14, 17]
    edges_in: []
  -
    node_type: 0
    node_id: 1
    name: "goto_waypoint_start"
    action:
      action_id: 0
      name: "goto_waypoint"
      parameters:
        -
          key: "v"
          value: "kenny"
        -
          key: "from"
          value: "wp0"
        -
          key: "to"
          value: "wp0"
      duration: 10.0
      dispatch_time: 0.0
    edges_out: [1]
    edges_in: [0]
  -
    node_type: 1
    node_id: 2
    name: "goto_waypoint_end"
    action:
```

```
edges:
  -
    edge_id: 0
    edge_name: "edge_0"
    signal_type: 0
    source_ids: [0]
    sink_ids: [1]
    duration_lower_bound: 0.0
    duration_upper_bound: 1.79769313486e+308
  -
    edge_id: 1
    edge_name: "edge_1"
    signal_type: 0
    source_ids: [1]
    sink_ids: [2]
    duration_lower_bound: 10.0
    duration_upper_bound: 10.0
  -
    edge_id: 2
    edge_name: "edge_2"
    signal_type: 0
    source_ids: [0]
    sink_ids: [2]
    duration_lower_bound: 0.0
    duration_upper_bound: 1.79769313486e+308
  -
    edge_id: 3
    edge_name: "edge_3"
    signal_type: 0
    source_ids: [2]
    sink_ids: [3]
    duration_lower_bound: 0.0
    duration_upper_bound: 1.79769313486e+308
  -
    edge_id: 4
    edge_name: "edge_4"
    signal_type: 0
    source_ids: [3]
    sink_ids: [4]
```



duration_lower_bound
duration_upper_bound

node_id: 0
name: plan_start

node_type: PLAN_START
edges_in = []
edges_out = [0]

edge's id's!

[0, 0]

edge_name: edge_0
edge_id: 0
source_ids: [0]
sink_ids: [1]

node id's!

node_id: 1
name: go_to_waypoint_start

node_type: ACTION_START
edges_in = [0]
edges_out = [1]

[0, 10]

edge_name: edge_1
edge_id: 1
source_ids: [1]
sink_ids: [2]

node_id: 2
name: go_to_waypoint_end

node_type: ACTION_END
edges_in = [1]
edges_out = [2]

[10, 0]

edge_name: edge_2
edge_id: 2
source_ids: [2]
sink_ids: [3]

node_id: 3
name: go_to_waypoint_start

node_type: ACTION_START
edges_in = [2]
edges_out = [3]

[10, 5]

edge_name: edge_3
edge_id: 3
source_ids: [3]
sink_ids: [4]

node_id: 4
name: go_to_waypoint_end

node_type: ACTION_END
edges_in = [3]
edges_out = [4]

[15, 0]

edge_name: edge_4
edge_id: 4
source_ids: [4]
sink_ids: [5]

node_id: 5
name: plan_end

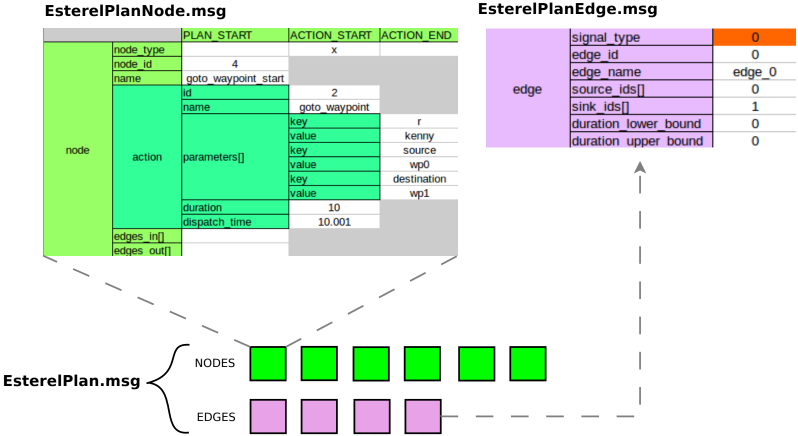
node_type: PLAN_END
edges_in = [4]
edges_out = []

Does not exist yet in ROSPlan!

EsterelPlan msg (graph)

rosplan_dispatch_msgs/EsterelPlan.msg

=============



EsterelPlanNode.msg

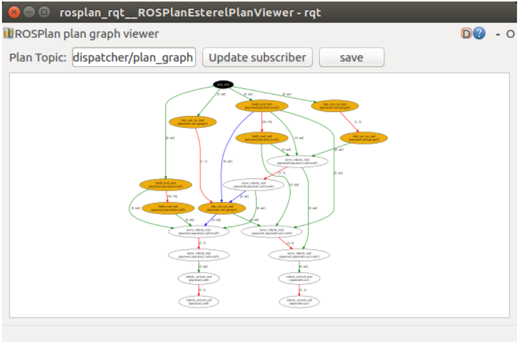
		PLAN_START	ACTION_START	ACTION_END
node	node_type		x	
	node_id	4		
	name	goto_waypoint_start		
	action	id	2	
		name	goto_waypoint	
		parameters[]	key	r
			value	kenny
			key	source
			value	wp0
			key	destination
			value	wp1
		duration	10	
		dispatch_time	10.001	
	edges_in[]			
	edges_out[]			

EsterelPlanEdge.msg

edge	signal_type	0
	edge_id	0
	edge_name	edge_0
	source_ids[]	0
	sink_ids[]	1
	duration_lower_bound	0
	duration_upper_bound	0

Graph visualization

- Based on graphviz
- Uses .dot format
- Can save graph, which can be visualized with xdot
- Updates execution status in real time, based on action feedback


rosplan_rqt__ROSPlanEsterelPlanViewer - rqt
ROSPlan plan graph viewer
Update subscriber
save
plan_start

Plan Topic: dispatcher/plan_graph

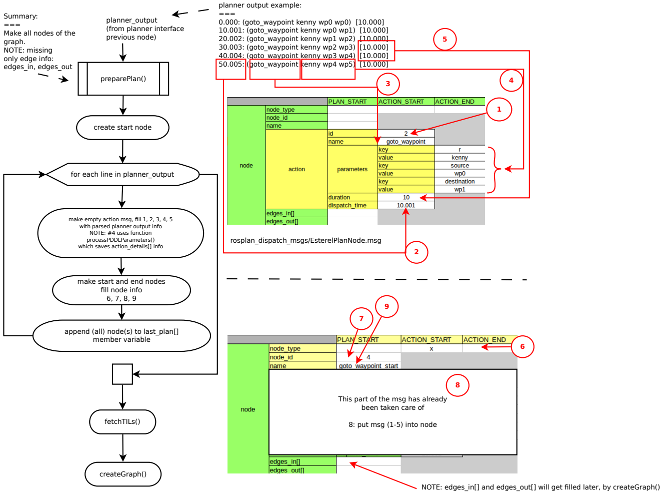
command to show rqt_esterel:

```
rqt --standalone rosplan_rqt.esterel_plan_viewer.ROSPlanEsterelPlanViewer --force-discover
```

Esterel plan generation steps - 1 prepare plan

PreparePlan()

==========



Summary:
===
Make all nodes of the graph.
NOTE: missing only edge info: edges_in, edges_out

planner_output
(from planner interface previous node)

preparePlan()
create start node
for each line in planner_output
make empty action msg, fill 1, 2, 3, 4, 5 with parsed planner output info NOTE: #4 uses function processPDDLParameters() which saves action_details[] info
make start and end nodes fill node info 6, 7, 8, 9
append (all) node(s) to last_plan[] member variable
fetchTILs()
createGraph()
1
2
3
4
5
6
7
8
9

planner output example:

```
===
0.000: (goto_waypoint kenny wp0 wp0)  [10.000]
10.001: (goto_waypoint kenny wp0 wp1)  [10.000]
20.002: (goto_waypoint kenny wp1 wp2)  [10.000]
30.003: (goto_waypoint kenny wp2 wp3)  [10.000]
40.004: (goto_waypoint kenny wp3 wp4)  [10.000]
50.005: (goto_waypoint kenny wp4 wp5)  [10.000]
```

		PLAN_START	ACTION_START	ACTION_END
node	node_type			
	node_id			
	name			
	action	id	2	
		name	goto_waypoint	
		parameters	key	r
			value	kenny
			key	source
			value	wp0
			key	destination
			value	wp1
		duration	10	
		dispatch_time	10.001	
	edges_in[]			
	edges_out[]			

rosplan_dispatch_msgs/EsterelPlanNode.msg

		PLAN_START	ACTION_START	ACTION_END
node	node_type		x	
	node_id	4		
	name	goto_waypoint_start		
	edges_in[]			
	edges_out[]			

This part of the msg has already been taken care of

8: put msg (1-5) into node

NOTE: edges_in[] and edges_out[] will get filled later, by createGraph()

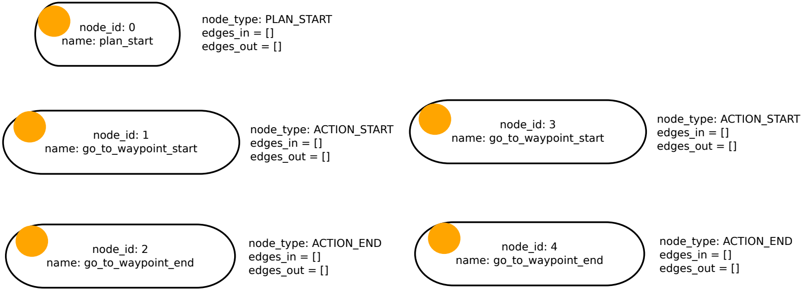
Esterel plan generation steps - 1 prepare plan

step 1 : PreparePlan() output

planner output example:

```
===
10.001: (goto_waypoint kenny wp0 wp1)  [10.000]
20.002: (goto_waypoint kenny wp1 wp2)  [10.000]
```

● Node associated action (not displayed)



node_id: 0
name: plan_start

node_type: PLAN_START
edges_in = []
edges_out = []

node_id: 1
name: go_to_waypoint_start

node_type: ACTION_START
edges_in = []
edges_out = []

node_id: 3
name: go_to_waypoint_start

node_type: ACTION_START
edges_in = []
edges_out = []

node_id: 2
name: go_to_waypoint_end

node_type: ACTION_END
edges_in = []
edges_out = []

node_id: 4
name: go_to_waypoint_end

node_type: ACTION_END
edges_in = []
edges_out = []

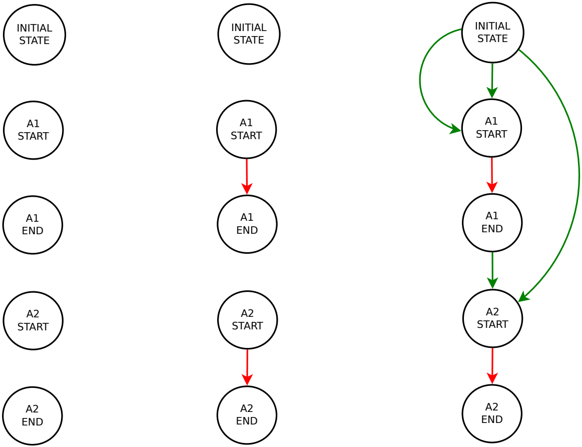
Esterel plan generation steps - 2, 3, 4

Make edges, NOTE: step 5 missing (interference edges)

step 2 : order nodes by dispatch time

step 3 : create start-end action edges

step 4 : create conditional support edges



Interference edge

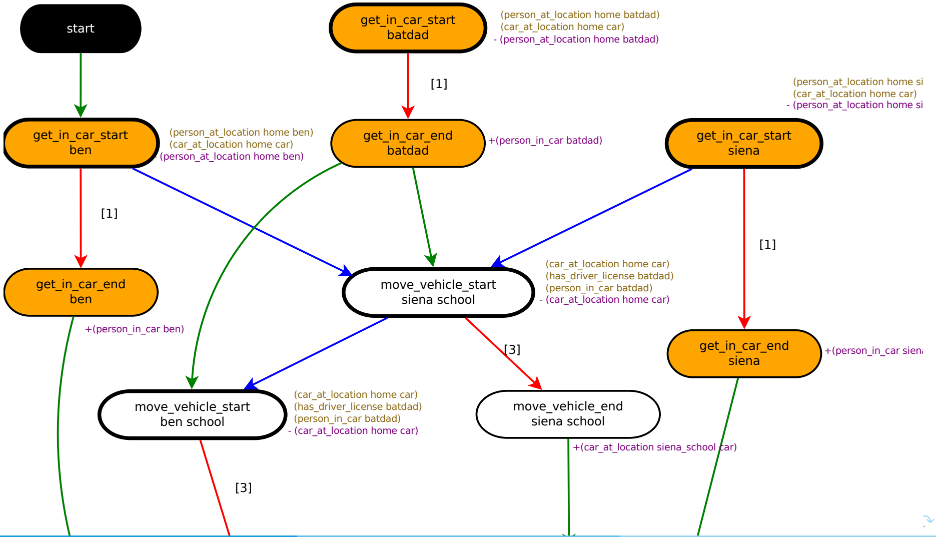
a and b interfere if:

- $eff^{+}a \cap eff^{-}b \neq \emptyset$
- $pre^{+}a \cap eff^{-}b \neq \emptyset$ *
- $pre^{-}a \cap eff^{+}b \neq \emptyset$ *
- $eff^{n}a \cap eff^{n}b \neq \emptyset$

* ROSPlan currently checks the ones marked with *

Interference edge - example

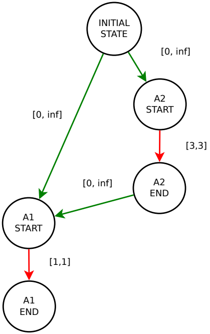
open pdf , domain , problem

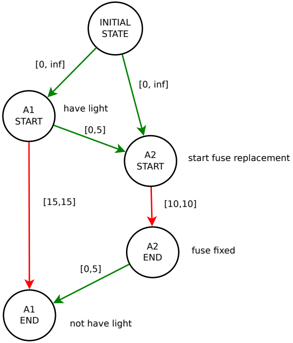

start
get_in_car_start batdad
(person_at_location home batdad)
(car_at_location home car)
- (person_at_location home batdad)
[1]
(person_at_location home si
(car_at_location home car)
- (person_at_location home si
get_in_car_start ben
(person_at_location home ben)
(car_at_location home car)
- (person_at_location home ben)
get_in_car_end batdad
+(person_in_car batdad)
get_in_car_start siena
[1]
[1]
get_in_car_end ben
move_vehicle_start siena school
(car_at_location home car)
(has_driver_license batdad)
(person_in_car batdad)
- (car_at_location home car)
+(person_in_car ben)
[3]
get_in_car_end siena
+(person_in_car siena
move_vehicle_start ben school
(car_at_location home car)
(has_driver_license batdad)
(person_in_car batdad)
- (car_at_location home car)
move_vehicle_end siena school
+(car_at_location siena_school car)
[3]

Semantics behind edges

- Conditional edge encapsulates 1 or more casual links
- All edges specify ordering constraints: source_node effects need to happen before sink_node gets signal
- Node cannot fire unless it has received all incoming edges / signals

Examples:

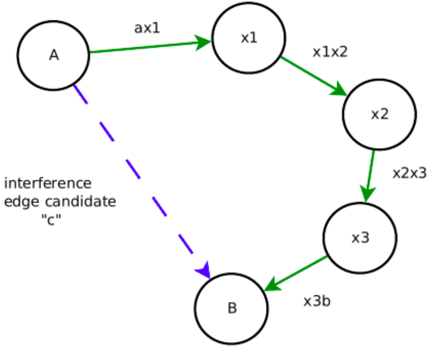




Interference edges optimization

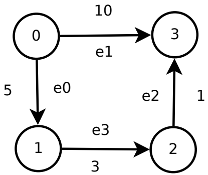
- If interference edge candidate ”c” (in blue) pretends to connect nodes A and B (A being source node and B being sink node) then candidate can be discarded if it exists an alternative path connecting A and B (e.g. ax1, x1x2, x2x3, x3b)

Example:



Proposed solution

- Based on Floyd-Warshall algorithm



graph_as_matrix (not updated)

node sink ids

source ids

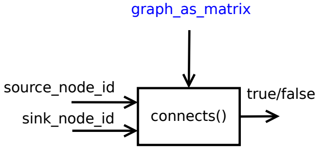
	0	1	2	3
0	0	5	inf	10
1	inf	0	3	inf
2	inf	inf	0	1
3	inf	inf	inf	0

graph_as_matrix (updated)

node sink ids

source ids

	0	1	2	3
0	0	5	**8**	**9**
1	inf	0	3	**4**
2	inf	inf	0	1
3	inf	inf	inf	0



Original solution vs Proposed solution, Results

problem	time w/ complete matrix updates	time w/ single update	time without optimization	time w/ previous method (getBounds)
P15-15-3-1-1	47	35	35	58

Future work

- Esterel plans are currently computed offline.
- Idea is to extend ROSPlan Esterel PlanDispatch to work online by taking input from sensed predicates and updating the execution graph in real time.
- Steps involve scenario creation, state of the art search, formalize question, implement solution in ROSPlan, formal guarantees of safety/correctness + experiments!

Thank you!

Questions? :)