rFSM Coordination Statecharts

Practical session

Markus Klotzbuecher, **Herman Bruyninckx**
Department of Mechanical Engineering, K.U.Leuven

March 6, 2012

KATHOLIEKE UNIVERSITEIT LEUVEN

BRICS

Overview

Goal: construct and validate an `rFSM` Statechart to **coordinate** an **iTaSC laser tracing** application.

1. Run a simple example stand-alone (without RTT)
2. Run a simple example using RTT
3. Extend the simple example to model the iTaSC Coordination.

Basic rFSM Statechart Elements

constructing fsm:

- **states:** `rfsm.state{ entry=<function>, exit=<function>, doo=<function>}`
- **transitions:** `rfsm.transition{src='stateX', tgt='stateY', events={'e_1', 'e_2' }}`
- **connectors:** `rfsm.connector{}`

instantiating:

- **load fsm model from file**: `model=rfsm.load(filename)`
- **initialize model**: `fsm=rfsm.init(model)`

running:

- **execute one step:** `rfsm.step(fsm)`
- **execute** `step` **until nothing left to do:** `rfsm.run(fsm)`

Basic Model

```
return rfsm.state {
 hello = rfsm.state{ entry=function() print("hello") end },
 world = rfsm.state{ entry=function() print("world") end },

 rfsm.trans { src='initial', tgt='hello' },
 rfsm.trans { src='hello', tgt='world', events={ 'e_done' } },
 rfsm.trans { src='world', tgt='hello', events={ 'e_restart' }
}
```

Run it stand-alone ...

```
$ roscd rFSM
$ tools/rfsm-sim examples/hello_world.lua
$ help()
$ > pp()
  root
   hello[LS]
   world[LS]
$ dbg(true)
$ step()
$ pp()
$ step()
$ se("e_restart")
$ run()
```

Running rFSM in RTT Components

```
$ roscd rfsm-rtt-example
```

Important files:

1. `run.sh`: toplevel script to launch the example
2. `deploy.[lua|ops]`: deployment script to launch Lua component, connect, configure ....
3. `launch_fsm`: RTT-Lua Component, creates ports, loads FSM model
4. `fsm.lua`: the actual platform independent FSM.

Build an iTaSC Coordinator...

Requirements:

1. Start safetey ASAP: enable base and joint limit avoidance tasks
2. wait some time for tasks to start up (e.g. 3s), then move to initial position
3. when `e_start_tracing` is received, start tracing by starting `LissajousGenerator`.
4. Required submodes to toggle between different tracing modes:
   - penalize arm vs. penalize base
   - z-constraint active / inactive
   - rotation constraint active / inactive
5. Switching between circle or "infinity" figure tracing. (Note: this requires stopping this `LissajousGenerator`)

Some hints - timeevents ...

Time triggered events

```
require "rfsm_timeevent"
rfsm_timeevent.set_gettime_hook(rtt.getTime)

return state {
   one = state{},
   two = state{},
   three = state{},

   trans{ src='initial', tgt='one' },
   trans{ src='one', tgt='two', events={ 'e_after(0.1)' } },
   trans{ src='two', tgt='three', events={ 'e_after(0.2)' } },
   trans{ src='three', tgt=one, events={ 'e_after(0.3)' } },
}
```

Some hints - sequential AND ...

Lightweight parallel states:

```
require "rfsm_ext"

rfsm_ext.seqand {
  entry=function() dothis() end,
  exit=function() dothat() end,

  s1=rfsm.init(rfsm.load("subfsm1.lua")),
  s2=rfsm.init(
        state {
           entry=...
           s21=state{ },
           s22=state{ }.
           trans { ... } } )
}
```

Summary ...

http://www.orocos.org/wiki/main-page/
european-robotics-forum-2012-workshops/rfsm-session