SE 3XA3: Software Requirements Specification
Finite State Machine Simulator

Team #16, NonDeterministic Key
Anhao Jiao (jiaoa3)
Kehao Huang (huangk53)
Xunzhou Ye (yex33)

18 March 2022

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
March 18	0.1	Module hierarchy MIS for Condition, State, Transition
April 12	1.0	Complete MIS

1 Module Hierarchy

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Machine Module
Software Decision Module	Util Module State Module Condition Module Transition Module EventData Module Event Module

Table 2: Module Hierarchy

2 MIS of Condition Module

2.1 Interface Syntax

2.1.1 Exported Access Programs

Routine name	In	Out	Exceptions
Condition	str, bool	-	-
check	EventData	bool	-

2.2 Interface Semantics

2.2.1 State Variables

func a predicate, function to call for the condition check

target the expected boolean value return by the predicate. i.e., when True, the condition-checking function should return True to pass; when False, the predicate should return False to pass.

2.2.2 Environment Variables

N/A

2.2.3 Assumptions

The expected value of the predicated (target) is set to True if not specified.

2.2.4 Access Program Semantics

Condition(func, target):
  Input: name of the condition-checking callable, the target state
  Transition: assign values to the appropriate state variables
  Output: None
  Exception: None
check():
  Input: None
  Transition: check whether the condition passes
  Output: whether the condition passes
  Exception: None

3 MIS of State Module

3.1 Interface Syntax

3.1.1 Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
State	str | Enum, str | List[str] | None, str | List[str] | None, bool	-	-
enter	EventData	-	-
exit	EventData	-	-
add_callback	str, str	-	-

3.2 Interface Semantics

3.2.1 State Variables

name the name of the state

on_enter optional callables to trigger when a state is entered

on_exit optional callables to trigger when a state is exit

ignore_invalid_triggers optional flag to indicated if unhandled/invalid triggers should raise an exception

3.2.2 Environment Variables

N/A

3.2.3 Assumptions

N/A

3.2.4 Access Program Semantics

State(name, on_enter, on_exit, ignore_invalid_triggers):
  Input: as described in State Variables
  Transition: assign values to the appropriate state variables
  Output: None
  Exception: None
enter(event_data):
  Input: collection of relevant data related to the ongoing transition attempt

Transition: fire triggers in the context of event_data

Output: None

Exception: None

exit(event_data):

Input: collection of relevant data related to the ongoing transition attempt

Transition: fire triggers in the context of event_data

Output: None

Exception: None

add_callback(trigger, func):

Input: trigger—the type of triggering event. Must be one of 'enter' or 'exit'. func—the name of the callback function.

Transition: append the given function to the according trigger

Output: None

Exception: None

4 MIS of Transition Module

4.1 Interface Syntax

4.1.1 Exported Access Programs

Routine name	**In**	**Out**	**Exceptions**
Transition	str, str, [str | Callable | List[str | Callable] | None] * 4	-	-
add_callback	str, str	-	-
execute	EventData	bool	-

4.2 Interface Semantics

4.2.1 State Variables

source source state of the transition

dest destination state of the transition

prepare callbacks executed before conditions checks

conditions callbacks evaluated to determine if the transition should be executed

before callbacks executed before the transition is executed but only if condition checks have been successful

after callbacks executed after the transition is executed but only if condition checks have been successful

4.2.2 Environment Variables

N/A

4.2.3 Assumptions

N/A

4.2.4 Access Program Semantics

Transition(source, dest, conditions, unless, before, after, prepare):
  Input: as described in State Variables
  Transition: assign values to the appropriate state variables
  Output: None

Exception: None

add_callback(trigger, func):

Input: trigger—the type of triggering event. Must be one of ’before’, ’after’ or ’prepare’. func—the name of the callback function

Transition: append the given function to the according trigger

Output: None

Exception: None

execute(event_data):

Input: collection of relevant data related to the ongoing transition attempt

Transition: execute the transition

Output: whether or not the transition was successfully executed

Exception: None

5 MIS of Event Module

5.1 Interface Syntax

5.1.1 Exported Access Programs

Routine name	In	Out	Exceptions
add_transition	Transition	-	-
add_callback	str, str	-	-
trigger	Any	bool	-

5.2 Interface Semantics

5.2.1 State Variables

name name of the event, which is also the name of the triggering callable

machine the current Machine instance

transitions the collection of transitions

5.2.2 Environment Variables

N/A

5.2.3 Assumptions

N/A

5.2.4 Access Program Semantics

Event(name, machine):
Input: as described in the State Variables
Transition: assign values to the appropriate state variables
Output: None
Exception: None
add_transition(transition):
Input: the Transition instance to add to the list. In another words, the transition to be associated to this Event trigger.
Transition: add the transition to the list of potential transitions.
Output: None

Exception: None

trigger(model):

Input: the currently processed model

Transition: execute all transitions that match the current state.

Output: whether or not a transition was successfully executed.

Exception: None

add_callback(trigger, func):

Input: trigger—the type of triggering event. Must be one of 'before', 'after' or 'prepare'. func—the name of the callback function

Transition: append the given function to the according trigger

Output: None

Exception: None

6 MIS of EventData Module

6.1 Interface Syntax

6.1.1 Exported Access Programs

Routine name	In	Out	Exceptions
-	-	-	-

6.2 Interface Semantics

6.2.1 State Variables

state The State from which the Event was triggered

event The triggering Event

machine The current Machine instance

model The model/object the machine is bound to

args optional positional arguments from trigger method to store internally for possible later callback invocations

kwargs optional keyword arguments from trigger method to store internally for possible later callback invocations

6.2.2 Environment Variables

N/A

6.2.3 Assumptions

N/A

6.2.4 Access Program Semantics

EventData(state, event, machine, model, args, kwargs):
Input: as described in the State Variables
Transition: assign values to the appropriate state variables
Output: None
Exception: None

7 MIS of Machine Module

7.1 Interface Syntax

7.1.1 Exported Access Programs

Routine name	In	Out	Exceptions
add_model	Any, State | str	-	-
remove_model	Any	-	-
get_state	str	State	-
get_model_state	Any	str	-
is_state	str	bool	-
set_state	str | Enum | State	-	-
add_states	str | list | dict | Enum | State	-	-
add_transition	str, str | list[str], str, str | list[str] * 5	-	-
get_triggers	tuple	list[str]	-
get_transitions	str, str | Enum | State, str | Enum | State	list[Transition]	-
remove_transition	str, str, str	-	-
dispatch	str	-	-
callback	str, EventData	-	-

7.2 Interface Semantics

7.2.1 State Variables

model The object(s) whose states we want to manage. If set to 'SELF_LITERAL', the current Machine instance will be used as the model.

states A list or enumeration of valid states. If str or Enum, a new State instance will be created that is named according to the string or enum member's name.

initial The initial state of the passed model.

transitions An optional list of transition.

send_event When true, any arguments passed to trigger methods will be wrapped in an EventData object, allowing indirect and encapsulated access to data. When False, all positional and keyword arguments will be passed directly to all callback methods.

auto_transitions When True (default), every state will automatically have an associated to_<state name>() trigger in the model.

ignore_invalid_triggers When True, any calls to trigger methods that are not valid for the present state (e.g., calling an a_to_b() trigger when the current state is c) will be silently ignored rather than raising a MachineError.

before_state_change A callable called on every state change before the transition happened. It receives the same args as normal callbacks.

after_state_change A callable called on every state change after the transition happened. It receives the same args as normal callbacks.

prepare_event A callable called before possible transitions will be processed. It receives the same args as normal callbacks.

finalize_event A callable called on for each triggered event after transitions have been processed. This is also called when a transition raises an exception.

on_exception A callable called when an event raises an exception.

7.2.2 Environment Variables

SEPARATOR the separator for callback triggers

WILDCARD_ALL the wildcard character to represent all states

WILDCARD_SAME the wildcard character to represent this states

SELF_LITERAL the flag to indicate the machine instance itself is used as the model

7.2.3 Assumptions

N/A

7.2.4 Access Program Semantics

Machine(<state variables>):
  Input: as described in the State Variables
  Transition: assign values to the appropriate state variables
  Output: None
  Exception: None
add_model(model, initial):
  Input: model—to be registered, initial—the initial state for this particular model
  Transition: register the model with the state machine, initialize triggers and callbacks.
  Output: None

Exception: None

remove_model(model):

Input: model to be removed from the machine

Transition: remove a model from the state machine. The model will still contain all previously added triggers and callbacks.

Output: None

Exception: None

get_state(state):

Input: the name of the state in string formate

Transition: find the State instance with the passed name.

Output: the State instance matching the passed name.

Exception: None

get_model_state(model):

Input: the model of interest

Transition: find the current state of the passed model.

Output: the name of the state of the current state of the passed model.

Exception: None

is_state(model, state):

Input: model—the model of interest, state—the name of the checked state

Transition: check whether the current state of the model matches the named state.

Output: boolean indicating whether the model's current state is state

Exception: None

set_state(model, state):

Input: model—the targeted model, state—the value of state to be set

Transition: set the current state of the model

Output: None

Exception: None

add_states(states, on_enter, on_exit, ignore_invalid_triggers):

Input: states—a list, dict, the name of the state, or a State instance to be added into the state machine, on_enter/exit—callbacks to trigger when the state is entered(exited), ignore_invalid_triggers—indicates whether or not triggers fired from invalid states should be ignored.

Transition: parse input and adds new states to the machine.

Output: None

Exception: None

add_transition(trigger, source, dest, conditions, unless, before, after, prepare):

Input: trigger—the name of the method that will trigger the transition, source—the name of the source state, dest—the name of the destination state, conditions—

Conditions(s) that must pass in order for the transition to take place, unless—Condition(s) that must return False for the transition to take place, before/after—callbacks to call before(after) the transition, prepare—callbacks to call before evaluating the Condition(s)

Transition: create a new Transition instance and add it to the internal list.

Output: None

Exception:

get_triggers(*states):

Input: tuple of source states

Transition: collects all triggers from the passed states

Output: a list of all triggers sourced from the passed states

Exception: None

get_transitions(trigger, source, dest):

Input: trigger—the trigger name of the transition, source/dest—limits list to transition from(to) a certain state

Output: list of transitions given the passed limits

Exception: None

remove_transition(trigger, source, dest):

Input: trigger—the trigger name of the transition, source/dest—limits removal to transition from(to) a certain state

Output: None

Exception: None

dispatch(trigger):

Input: the name of the event to be triggered

Output: None

Exception: None

callback(func, event_data):

Input: func—the callbacks function to be invoked, event_data—an EventData instance to pass to the callback or to extract arguments from

Output: None

Exception: None