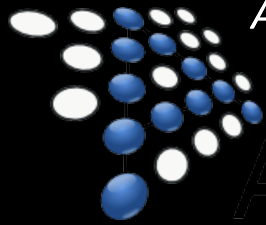
iUML-B Statemachines: New Features

Colin Snook

5th Rodin Workshop, Toulouse, 2nd June 2014



Question

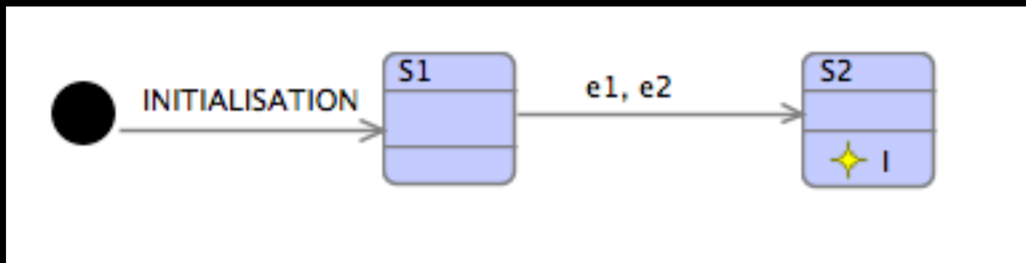
What features would you like?

(in iUML-B Statemachines)

Advance

Recap: iUML-B Statemachines

- Statemachines contained in Event-B Machine.
- Generates data representation of explicit state
- Adds guards and actions to elaborated events



- Transition elaboration : many to many

Label: e1, e2

Extended: false

Elaborates:

Event	Refines
e1	
e2	

Add Event | Remove Event | Create & Add | Remove & Delete

MACHINE basic
SEES basic_implicitContext
VARIABLES

state

INVARIANTS

typeof_state : $state \in state_STATES$

invS2 : $(state = S2) \Rightarrow (I)$

EVENTS

Initialisation

begin

init_state : $state := S1$

end

Event *e1* $\widehat{=}$

when

isin_S1 : $state = S1$

then

enter_S2 : $state := S2$

end

Event *e2* $\widehat{=}$

when

isin_S1 : $state = S1$

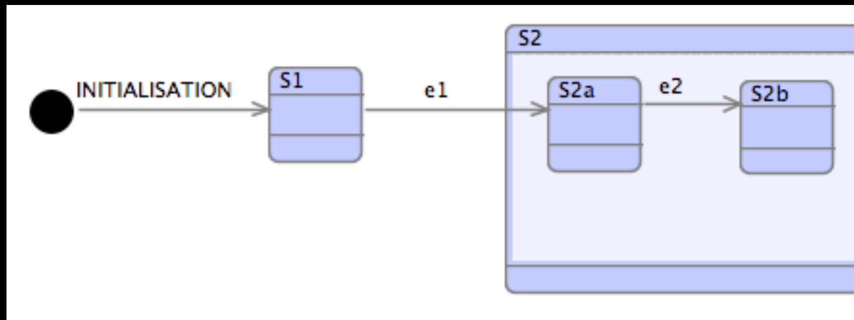
then

enter_S2 : $state := S2$

end

END

Recap: Hierarchical Statemachines



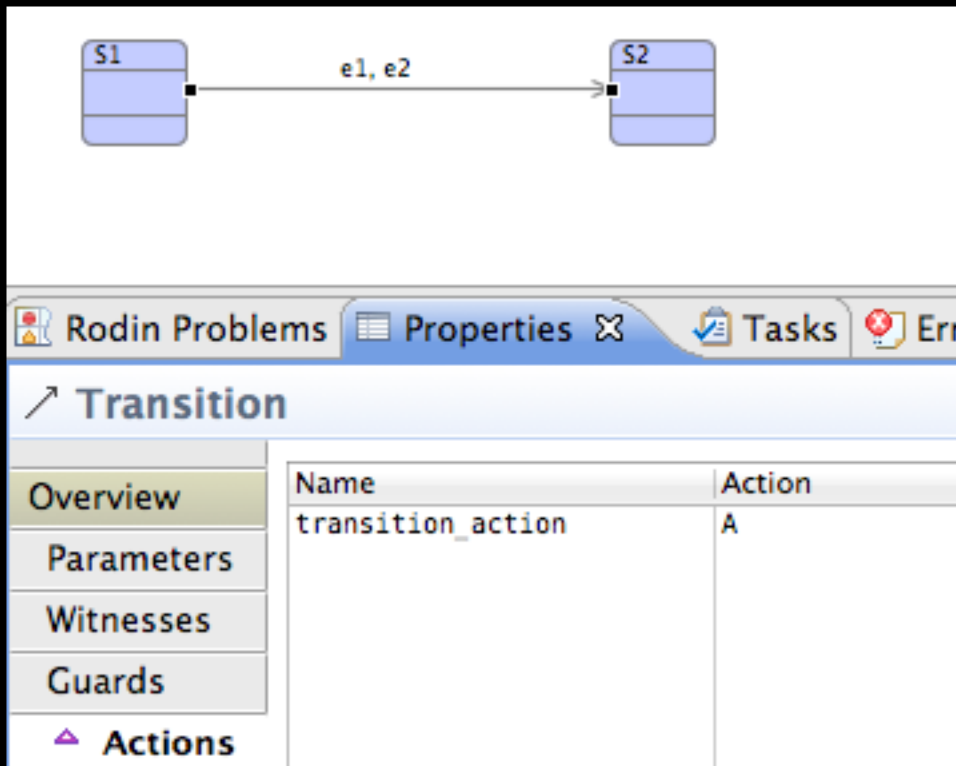
- Multiple Statemachines can be nested inside states
- Often add nesting in refinements

```
MACHINE nesting
SEES nesting_implicitContext
VARIABLES
    state
    S2_substate
INVARIANTS
    typeof_state : state ∈ state_STATES
    typeof_S2_substate : S2_substate ∈ S2_substate_STATES
    superstateof_S2_substate : S2_substate ≠ S2_substate_NULL ⇔ state = S2
EVENTS
Initialisation
    begin
        init_S2_substate : S2_substate := S2_substate_NULL
        init_state : state := S1
    end
Event e1 ≙
    when
        isin_S1 : state = S1
    then
        enter_S2 : state := S2
        enter_S2a : S2_substate := S2a
    end
Event e2 ≙
    when
        isin_S2a : S2_substate = S2a
    then
        enter_S2b : S2_substate := S2b
    end
END
```

Advance

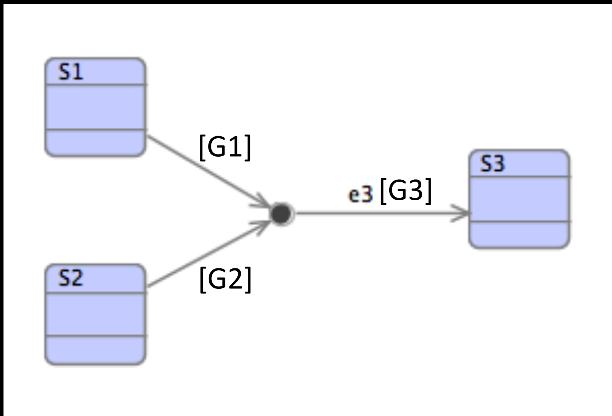
New: Transition Guards/Actions

Event features can be added to a transition. They are replicated in each elaborated event.




```
Event e1 ≙
    any
        trnstn_param
    where
        isin_S1 : state = S1
        trnstn_param_type : trnstn_param ∈ BOOL
        transition_guard : G
    with
        transition_witness : P
    then
        transition_action : A
        enter_S2 : state := S2
    end
Event e2 ≙
    any
        trnstn_param
    where
        trnstn_param_type : trnstn_param ∈ BOOL
        isin_S1 : state = S1
        transition_guard : G
    with
        transition_witness : P
    then
        enter_S2 : state := S2
        transition_action : A
    end
```

New: Junctions



Merging junction :– upstream branches form a disjunctive guard

(Only final transition segments can elaborate events,
Guards may be placed on any transition segment,
Actions etc. may only be placed on final segments)

Event $e3 \widehat{=}$

when

isin_S1_or_isin_S2 : $((state = S1 \land G1) \lor (state = S2 \land G2))$

g3 : $G3$

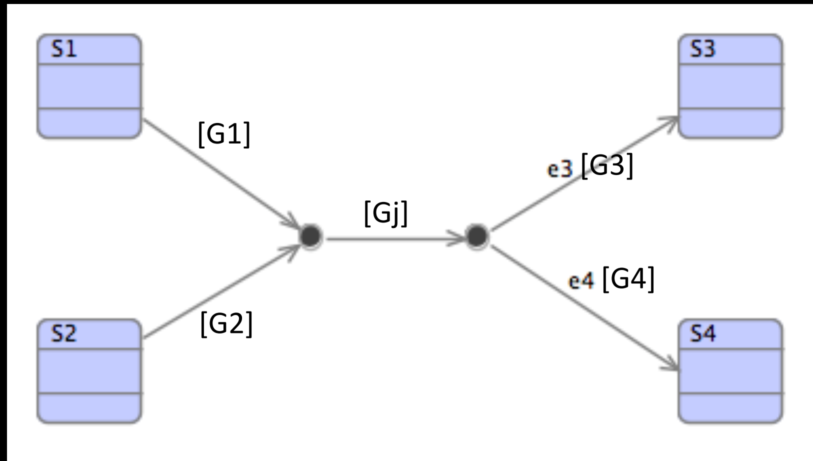
then

enter_S3 : $state := S3$

end

Advance

New: Junctions (cont.)



Splitting junction :– upstream contributes guards to all downstream branches

Event $e3 \widehat{=}$

when

isin_S1_or_isin_S2 : $((((state = S1 \wedge G1) \vee (state = S2 \wedge G2)) \wedge Gj))$

transition_guard : $G3$

then

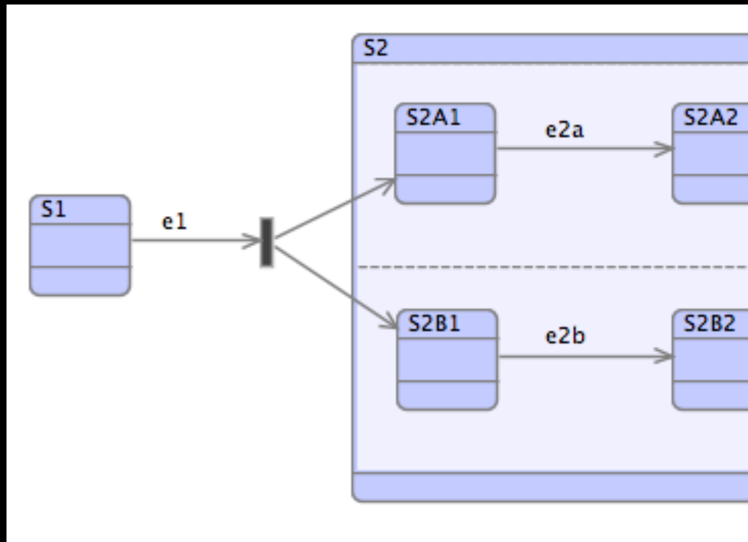
enter_S3 : $state := S3$

end

Advance

ADVANCE

New: Forks



Forks :– allow a transition to enter several parallel nested regions

Event *e1* $\widehat{=}$

when

isin_S1 : $state = S1$

then

enter_S2B1 : $S2B_state := S2B1$

enter_S2 : $state := S2$

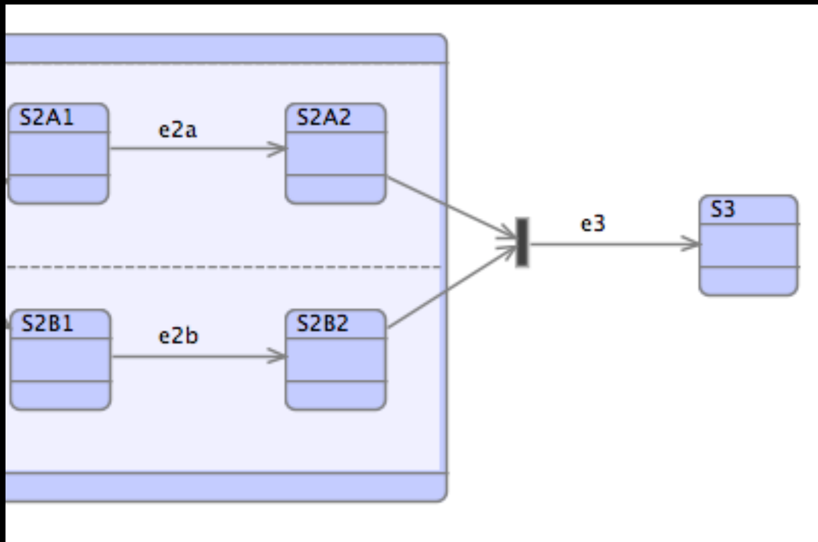
enter_S2A1 : $S2A_state := S2A1$

end

Advance

ADVANCE

New: Joins



Joins :– allow a transition to exit several parallel nested regions.

I.e. the exit transition is not enabled until all the exit states are reached

Event $e3 \widehat{=}$

when

isin_S2A2 : $S2A_state = S2A2$

isin_S2B2 : $S2B_state = S2B2$

then

enter_S3 : $state := S3$

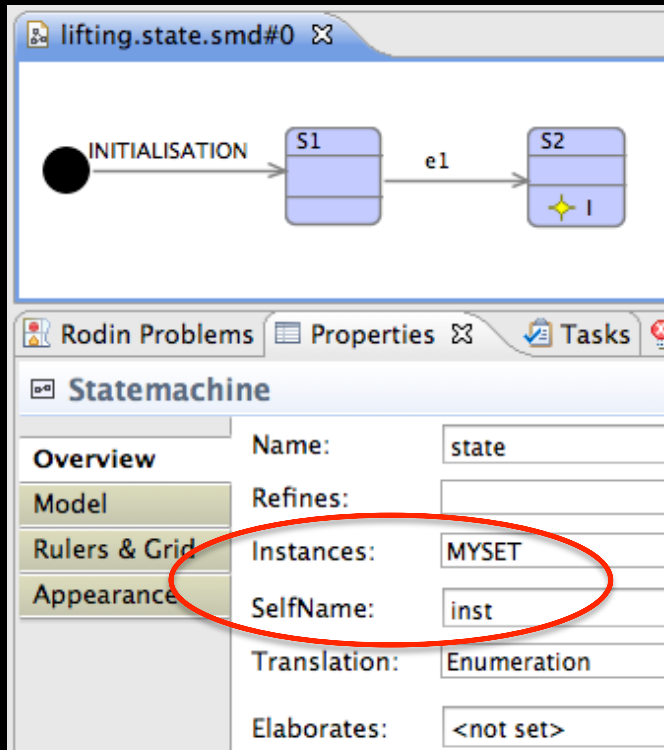
leave_S2A_state : $S2A_state := S2A_state_NULL$

leave_S2B_state : $S2B_state := S2B_state_NULL$

end

Advance

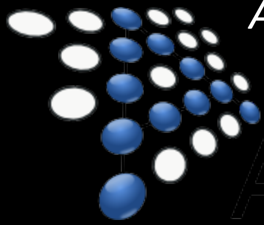
New: Statemachine Instances (lifting)



MACHINE lifting
SEES lifting_implicitContext, liftingInstances
VARIABLES
$state$
INVARIANTS
typeof_state : $state \in MYSET \rightarrow state_STATES$
inv1 : $\forall inst \cdot (state(inst) = S2) \Rightarrow (I)$
EVENTS
Initialisation
begin
init_state : $state := MYSET \times \{S1\}$
end
Event $e1 \;\widehat{=}$
any
$inst$
where
isin_S1 : $state(inst) = S1$
then
enter_S2 : $state(inst) := S2$
end
END

Similar to O-O class lifting.

Class diagrams (coming soon) will utilise this feature.



New: Statemachine Animation Improvements

- Launches BMotionStudio Animation
  - (if one is available for the same machine)
- Updated to support the new statemachine features
- Improved selection of transition firing

Example – Cruise control

ProB – platform:/resource/CruiseControl/m3.sm.smd#_5pFF8MVREeOu4L1HrsQVOQ – Rodin Platform – /Volumes/Data/RodinWorkspaces2.8/Demo

Answers?