ASLEV Data Flow Diagrams

Leon Starr
2025-7-27/ v1.0.0

State name: NOT REQUESTED

Anum: A13

Signum: 9

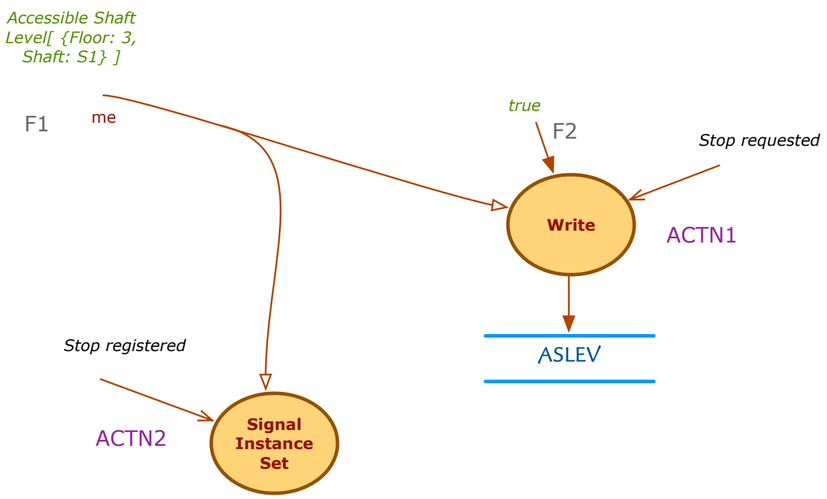
No actions

```
state NOT REQUESTED
activity
    // Not a desired dest for the Cabin
transitions
    Stop request > Registering stop
    Floor calling > Registering floor call
--
```

State name: Registering stop

Anum: A15

Signum: 9




```
state Registering stop
activity
    Stop requested = TRUE
    Stop registered -> me
transitions
    Stop registered > Requesting service
```

The problem with Stop requested.set is that we are implicitly writing the value of the Stop Requested attribute instead of merely applying a type operator.

We would have to know somehow that the .set operator does not require a scalar input, it always sets the value to true, regardless, and that an attribute write is required.

Instead, we need to clearly distinguish, in the Scrall grammar, between the output of a type operator, which is always a single scalar flow and a write or an assignment

So, instead of Stop requested.set, we could do one of these:

```
Stop requested = true // keyword
Stop requested = Boolean[true] // Selector op
Stop requested = _yes  // if Symbol type
```

In the first case, we specify an assignment taking advantage of the true keyword. This approach only works for a boolean type.

We could also use a symbolic selector Boolean[true], but the true keyword is shorthand for this anyway.

Or, we could use a Symbol Set type instead of boolean and define _yes, _no values. But this doesn't add any enlightenment from an analysis perspective.

Se could apply a boolean name pair to get the same effect, but we discard this approach for the same reason.

Instead of saying Direction.toggle() for the purpose of changing the up value to down or vice versa, with the implied write, we do this instead:

```
Direction = Direction.toggle // operator based on current value
```

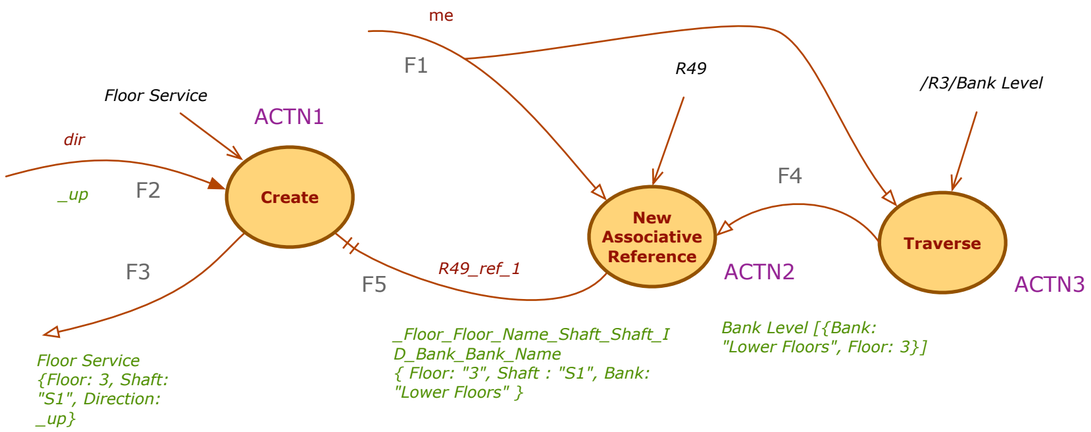
Here we make the assignment explicit, and the toggle operator simply outputs _up or _down depending on the current value of Direction.

Here we do use the boolean name pair _up/_down mapped to true/false on the Boolean base type.

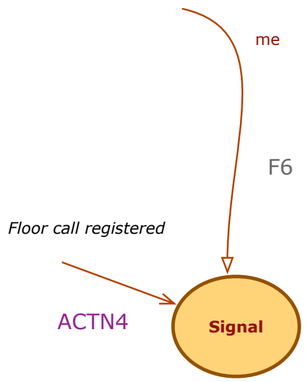
State name: Registering floor call

Anum: A18

Signum: 19



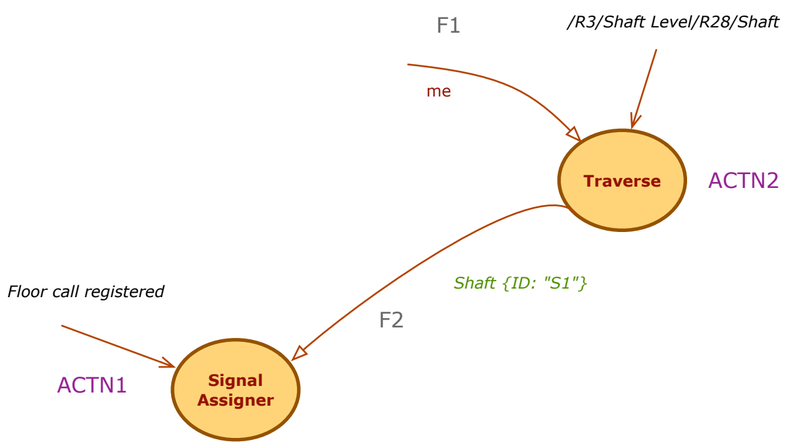

```
state Registering floor call( dir: Direction )
activity
    *Floor Service( Direction: ^dir ) &R49 me, /R3/Bank Level
    Floor call registered -> me
transitions
    Floor call registered > Requesting service
--
```



State name: Requesting service

Anum: A18

Signum: 9

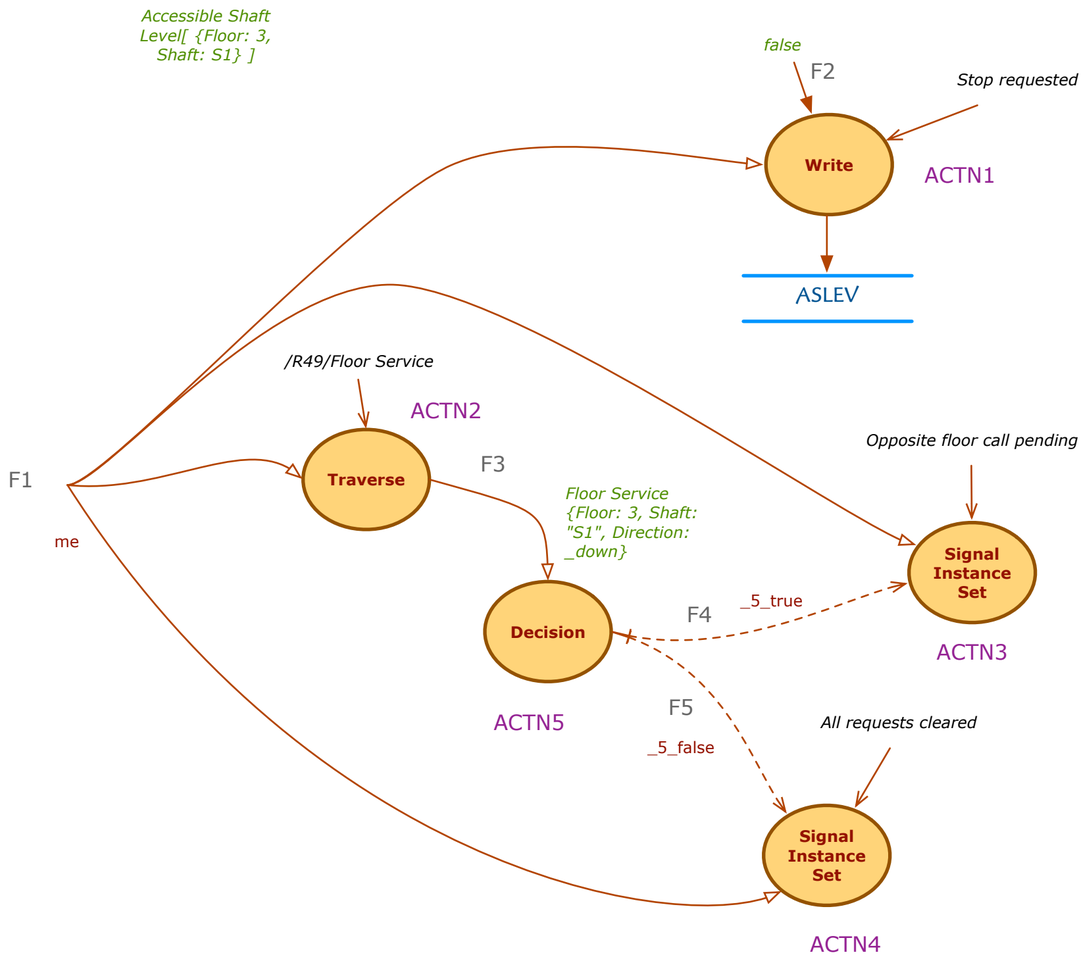



```
state Requesting service
activity
    Service requested -> R53( /R3/R28/Shaft )
    Wait for service -> me
transitions
    Wait for service > REQUESTED
--
```

State name: Clear stop request

Anum: A20

Signum: 14



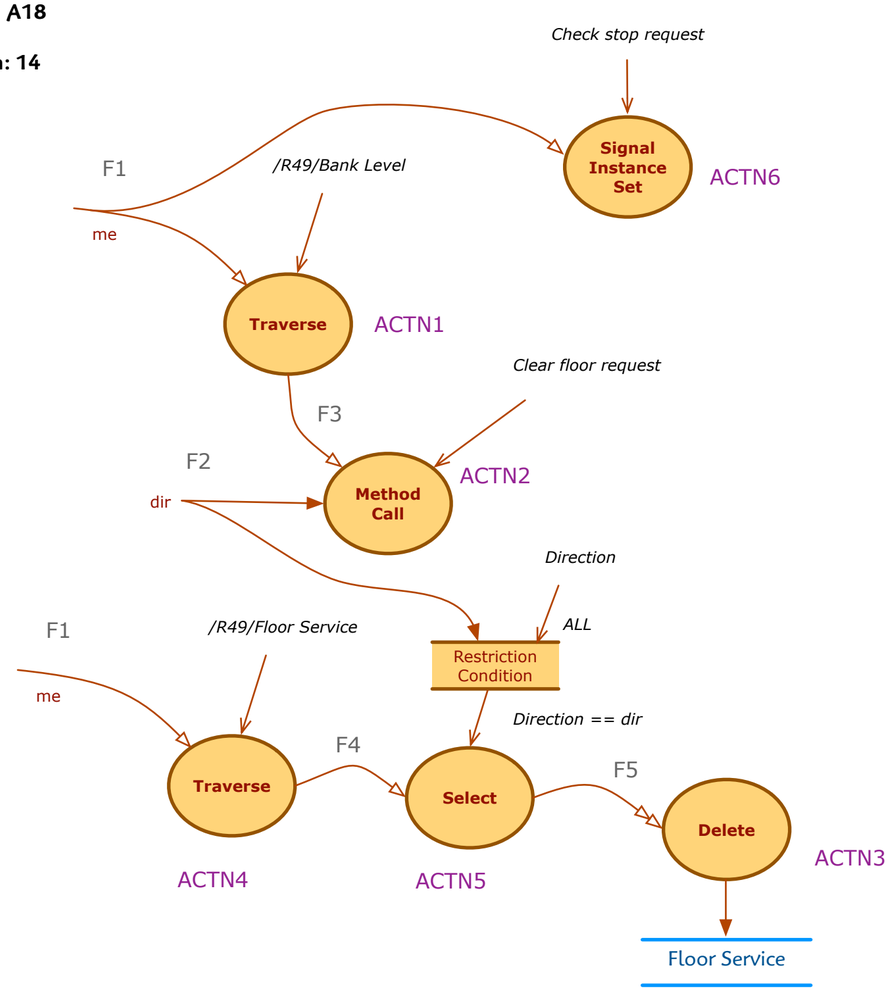

```
state Clear stop request
activity
    Stop requested = FALSE

    /R49/Floor Service?
        Opposite floor call pending -> :
        All requests cleared -> me
transitions
    Opposite floor call pending > REQUESTED
    All requests cleared > NOT REQUESTED
--
```

State name: Clear floor request

Anum: A18

Signum: 14




```
state Clear floor request( dir: Direction )
activity
    /R49/Bank Level.Clear floor request( ^dir )
    !* /R49/Floor Service( Direction: ^dir )
    Check stop request -> me
transitions
    Check stop request > Clear stop request
```