

Cabin estimate delay method Data Flow Diagrams

Leon Starr

2025-11-25/0.9

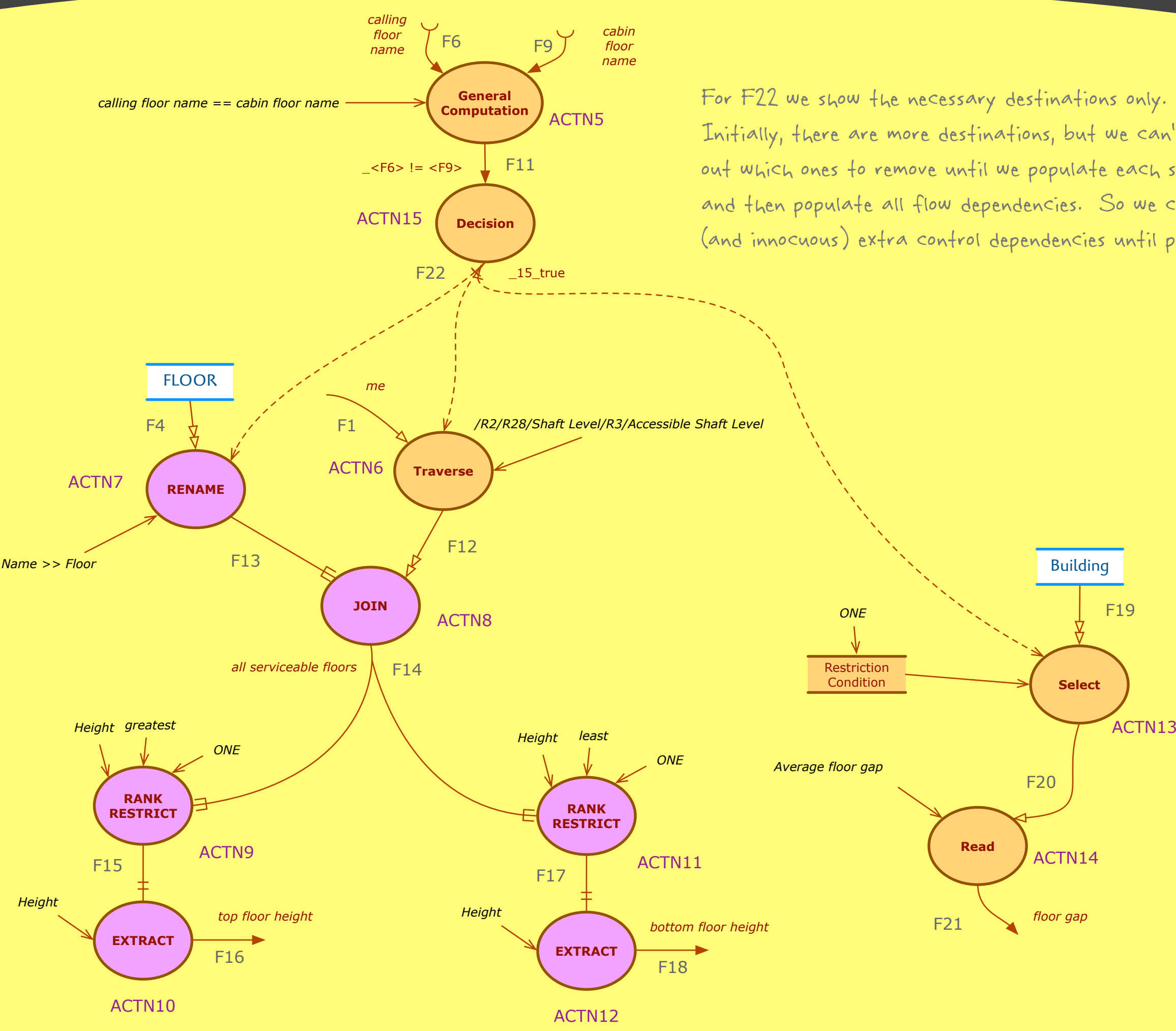


Copyright © 2025, Leon Starr at

MODEL INTEGRATION, LLC

A10

```
calling floor name != cabin floor name?  
{  
  all serviceable floors #= /R2/R28/Shaft Level/R3/Accessible Shaft Level ## Floor[Name >> Floor]  
  top floor height = all serviceable floors(1, ^+Height).Height  
  bottom floor height = all serviceable floors(1, ^-Height).Height  
  floor gap = Building(1).Average floor gap  
}
```

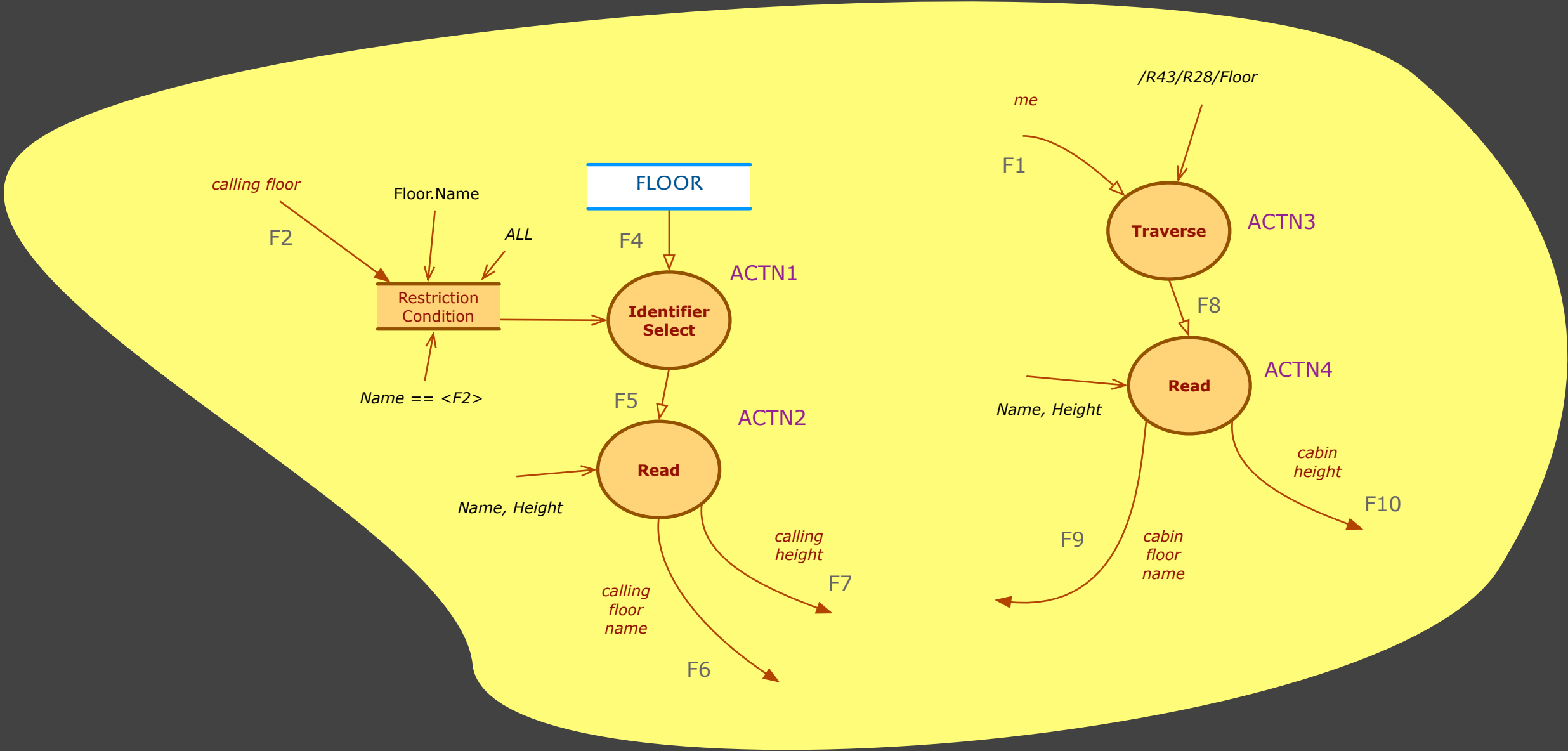


For F22 we show the necessary destinations only. Initially, there are more destinations, but we can't figure out which ones to remove until we populate each statement in the block and then populate all flow dependencies. So we can't remove the superfluous (and innocuous) extra control dependencies until post processing the entire Activity.

A10

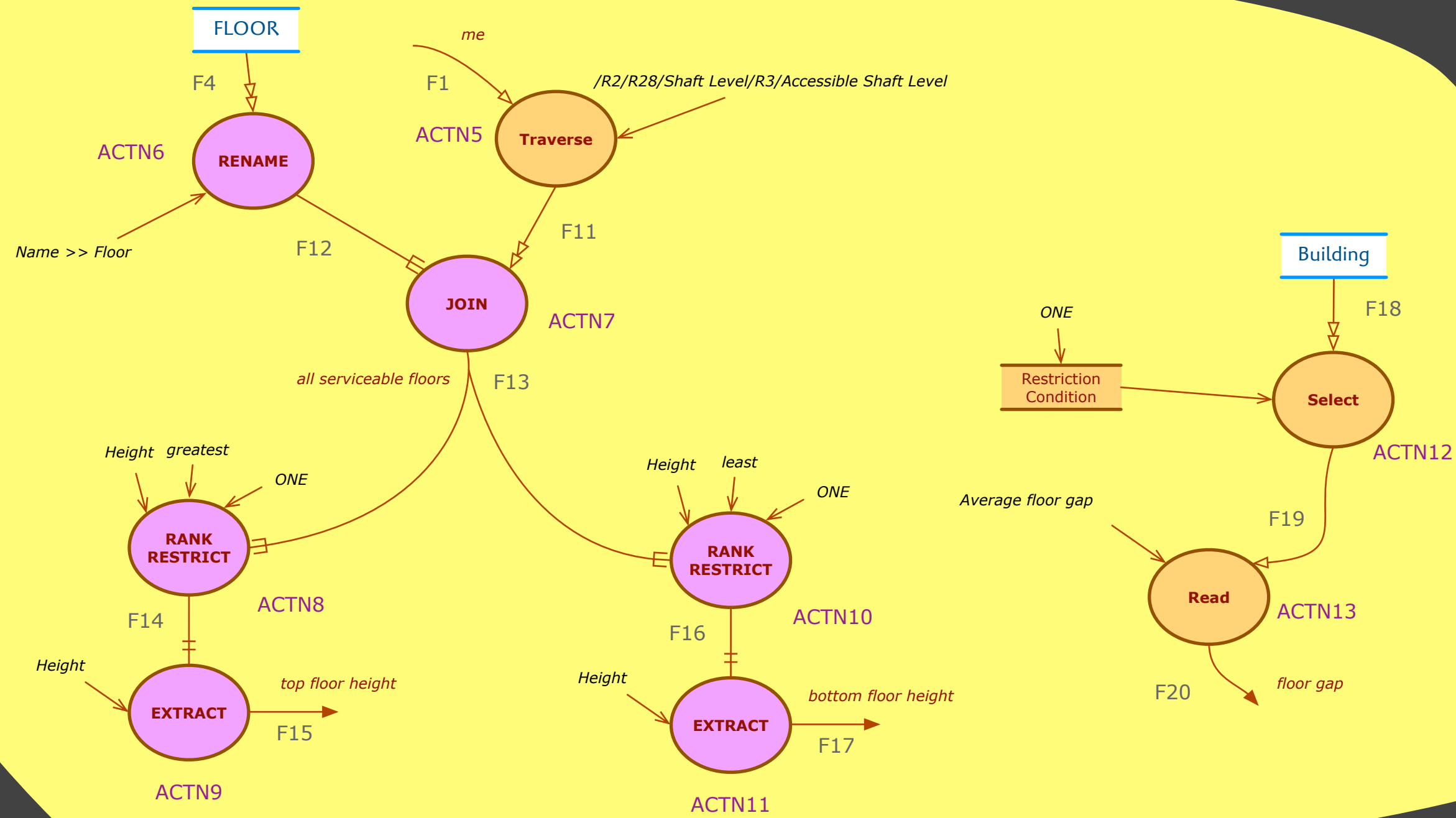
```
--
Cabin.Estimate delay( calling floor : Floor name, call dir : Direction )
--

calling floor name, calling height = Floor( Name: ^calling floor ).(Name, Height)
cabin floor name, cabin height = /R43/R28/Floor.(Name, Height)
```



A10

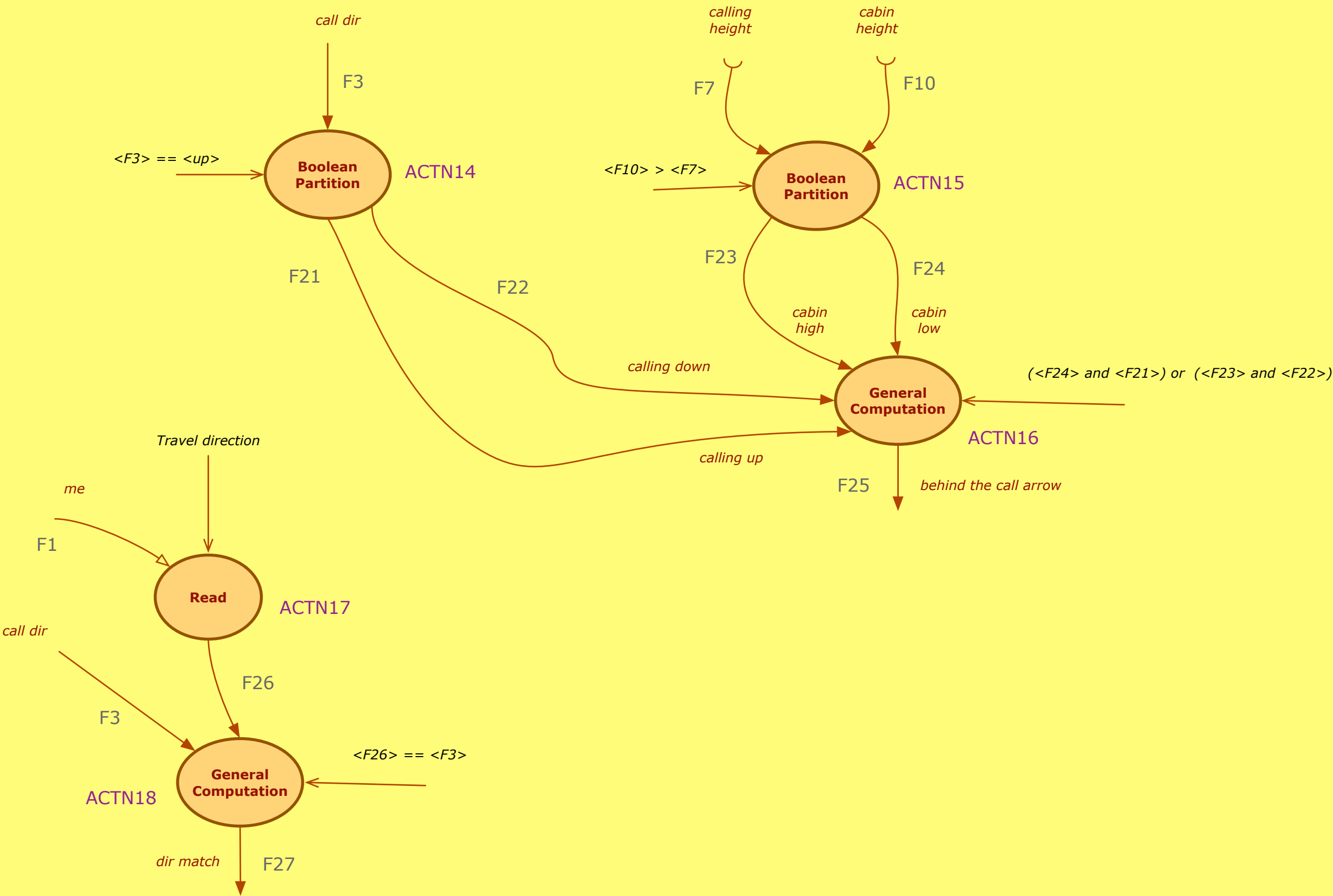
```
all serviceable floors #= /R2/R28/Shaft Level/R3/Accessible Shaft Level ## Floor[Name >> Floor]
top floor height = all serviceable floors(1, ^+Height).Height
bottom floor height = all serviceable floors(1, ^-Height).Height
floor gap = Building(1).Average floor gap
```



A10

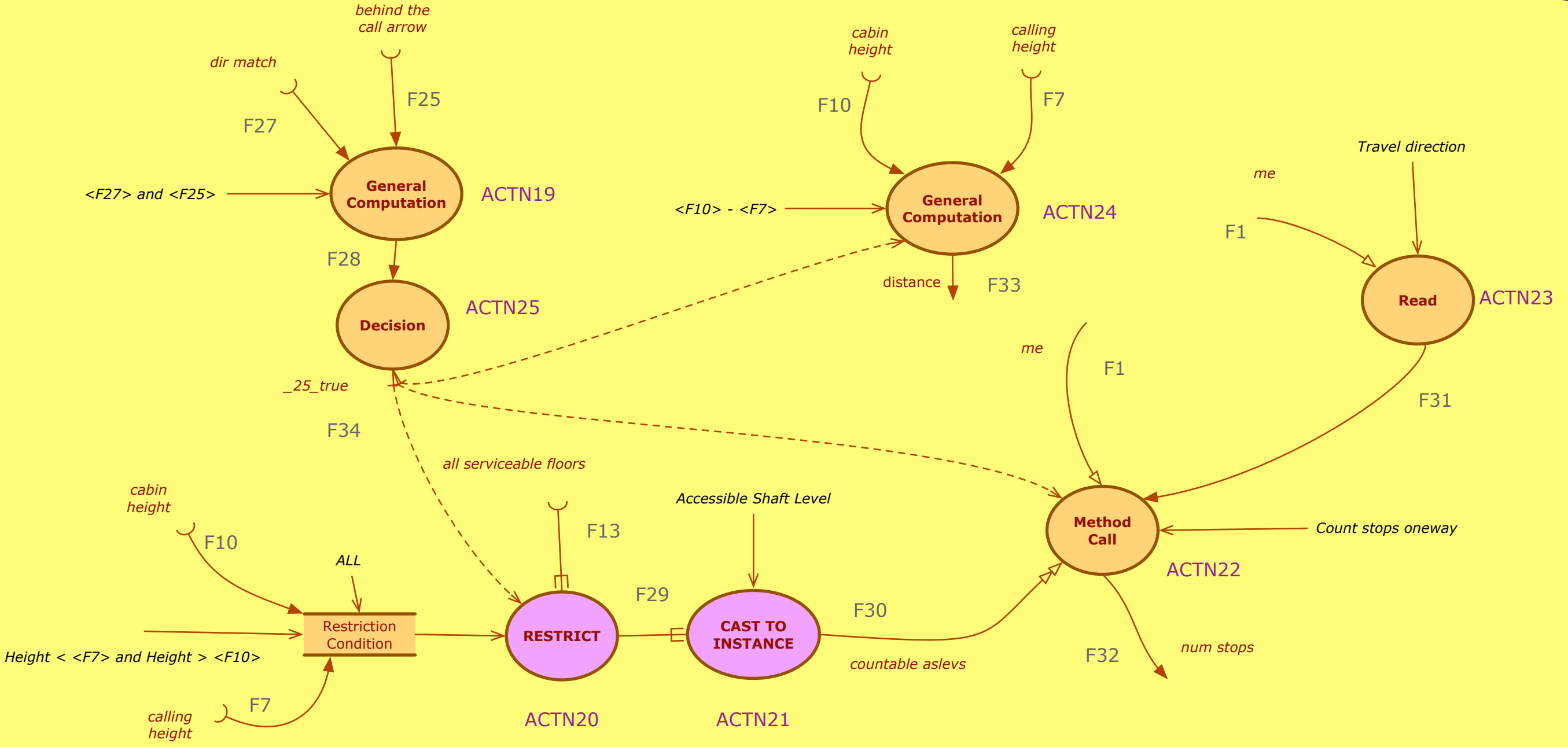
```
calling up, calling down = ^call dir == _up
cabin high, cabin low = cabin height > calling height

behind the call arrow = ( cabin low AND calling up ) OR ( cabin high AND calling down )
dir match = Travel direction == ^call dir
```



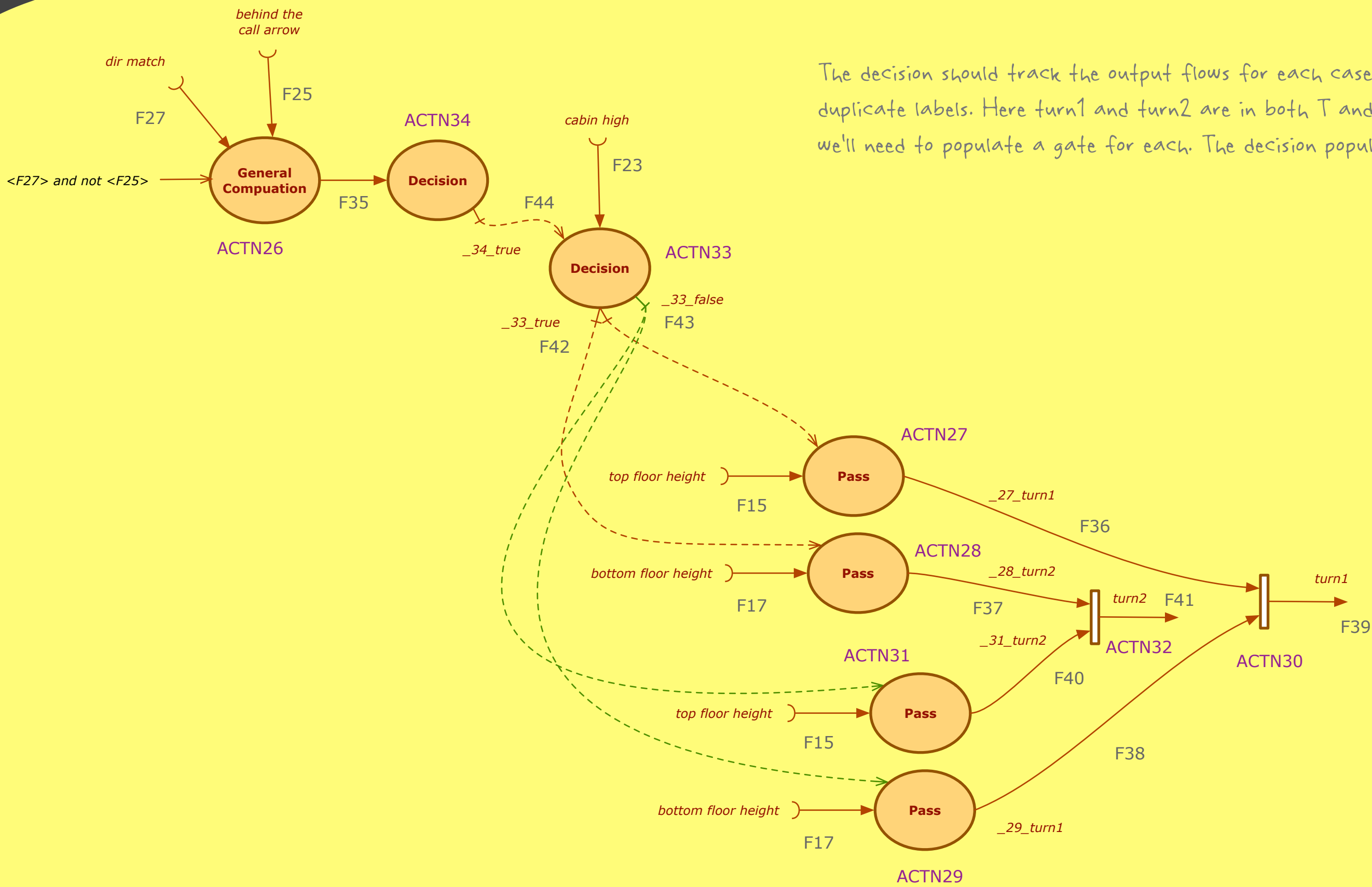
A10

```
dir match AND behind the call arrow? {
  countable aslevs::Accessible Shaft Level == all serviceable floors(Height < calling height AND Height > cabin height)
  num stops = .Count stops oneway( aslevs: countable aslevs, search dir: Travel direction )
  distance = cabin height - calling height
}
```



A10

```
dir match AND NOT behind the call arrow? {
  cabin high ? turn1, turn2 = top floor height, bottom floor height :
  |   turn1, turn2 = bottom floor height, top floor height
}
```



The decision should track the output flows for each case and detect any duplicate labels. Here turn1 and turn2 are in both T and F cases. This means that we'll need to populate a gate for each. The decision populates each Pass first.

A10

```
calling floor name != cabin floor name?
```

⋮

```
: ==>> Duration[0]
```

