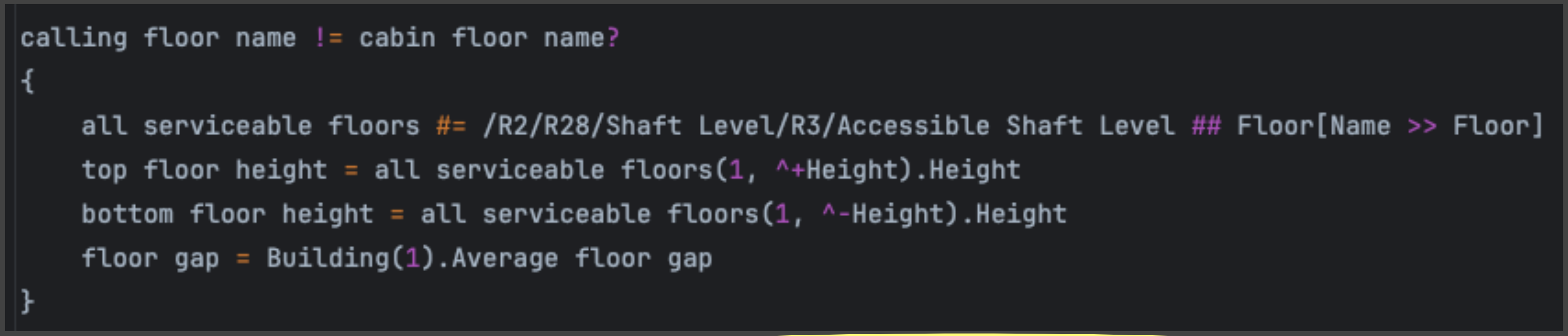
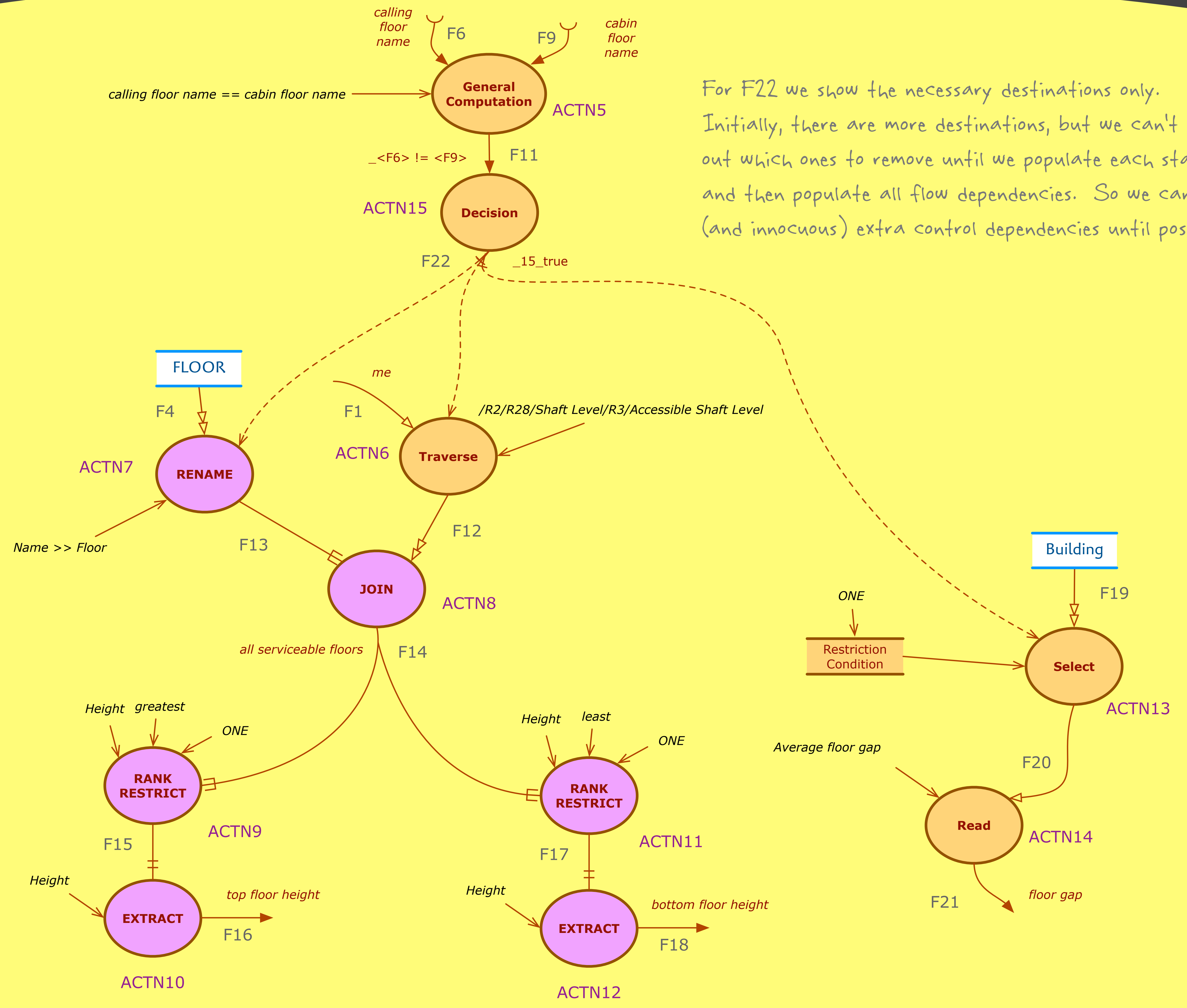
Cabin estimate delay method Data Flow Diagrams

Leon Starr
2025-11-30/1.3

A7



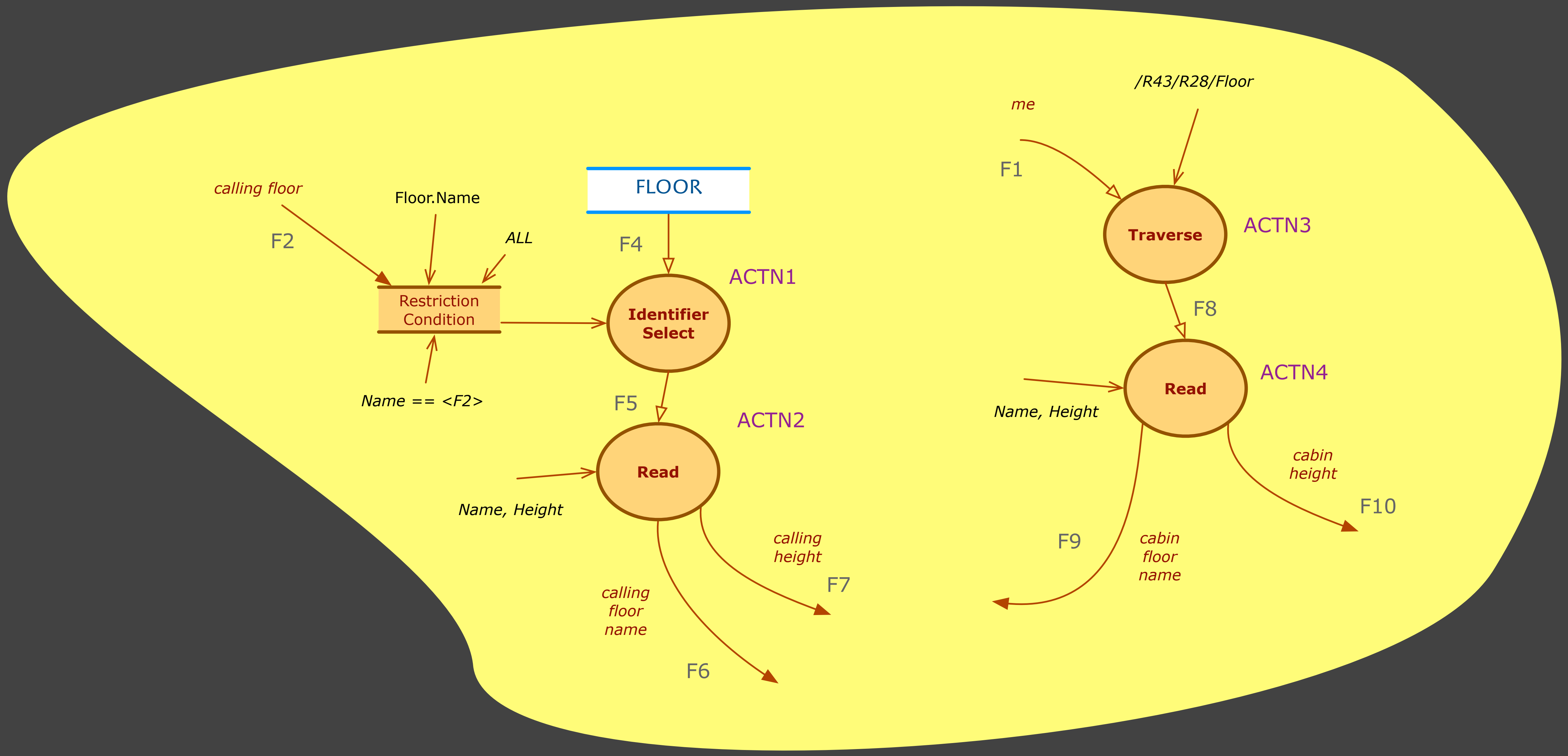
```
calling floor name != cabin floor name?
{
    all serviceable floors #= /R2/R28/Shaft Level/R3/Accessible Shaft Level ## Floor[Name >> Floor]
    top floor height = all serviceable floors(1, ^+Height).Height
    bottom floor height = all serviceable floors(1, ^-Height).Height
    floor gap = Building(1).Average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

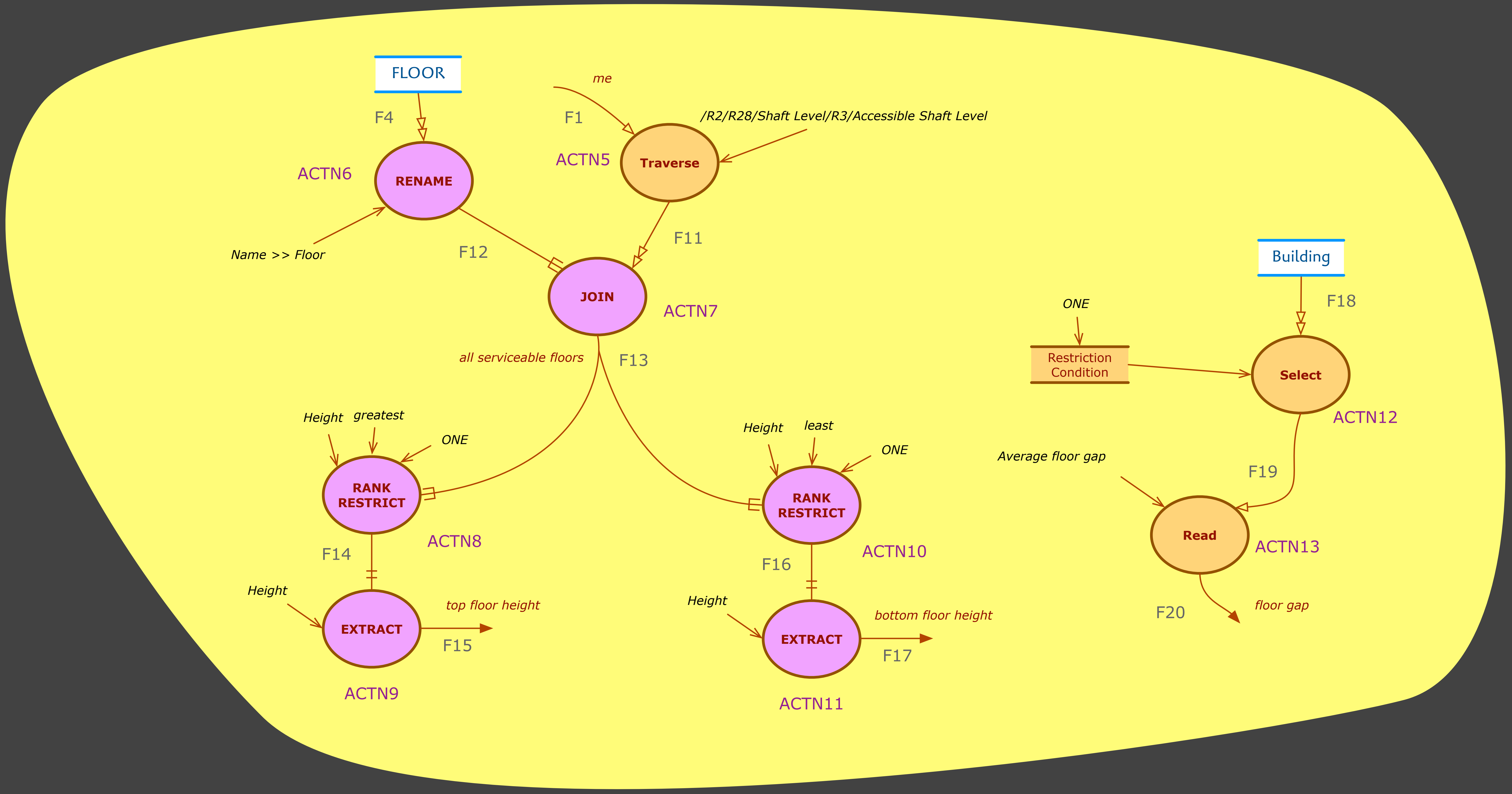
A7

```
--
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)
```



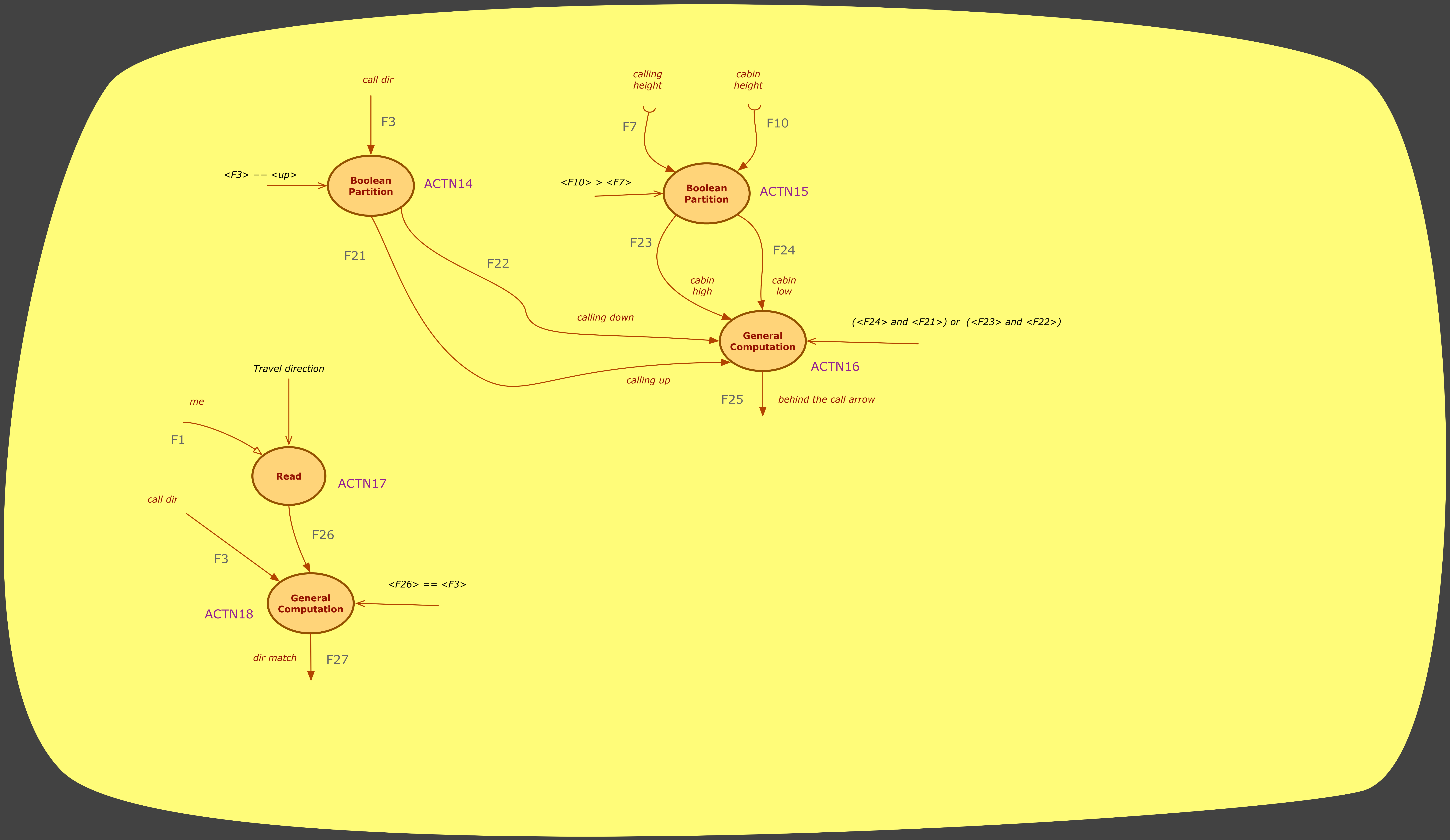
A7

```
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
```

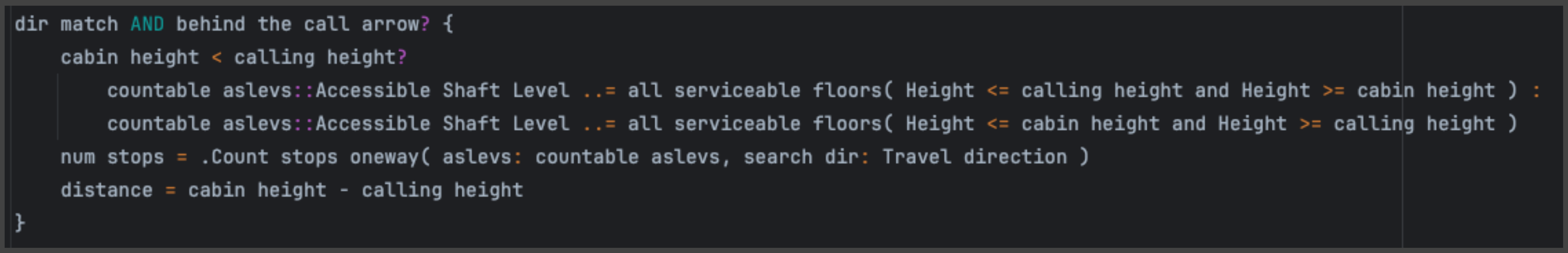


A7

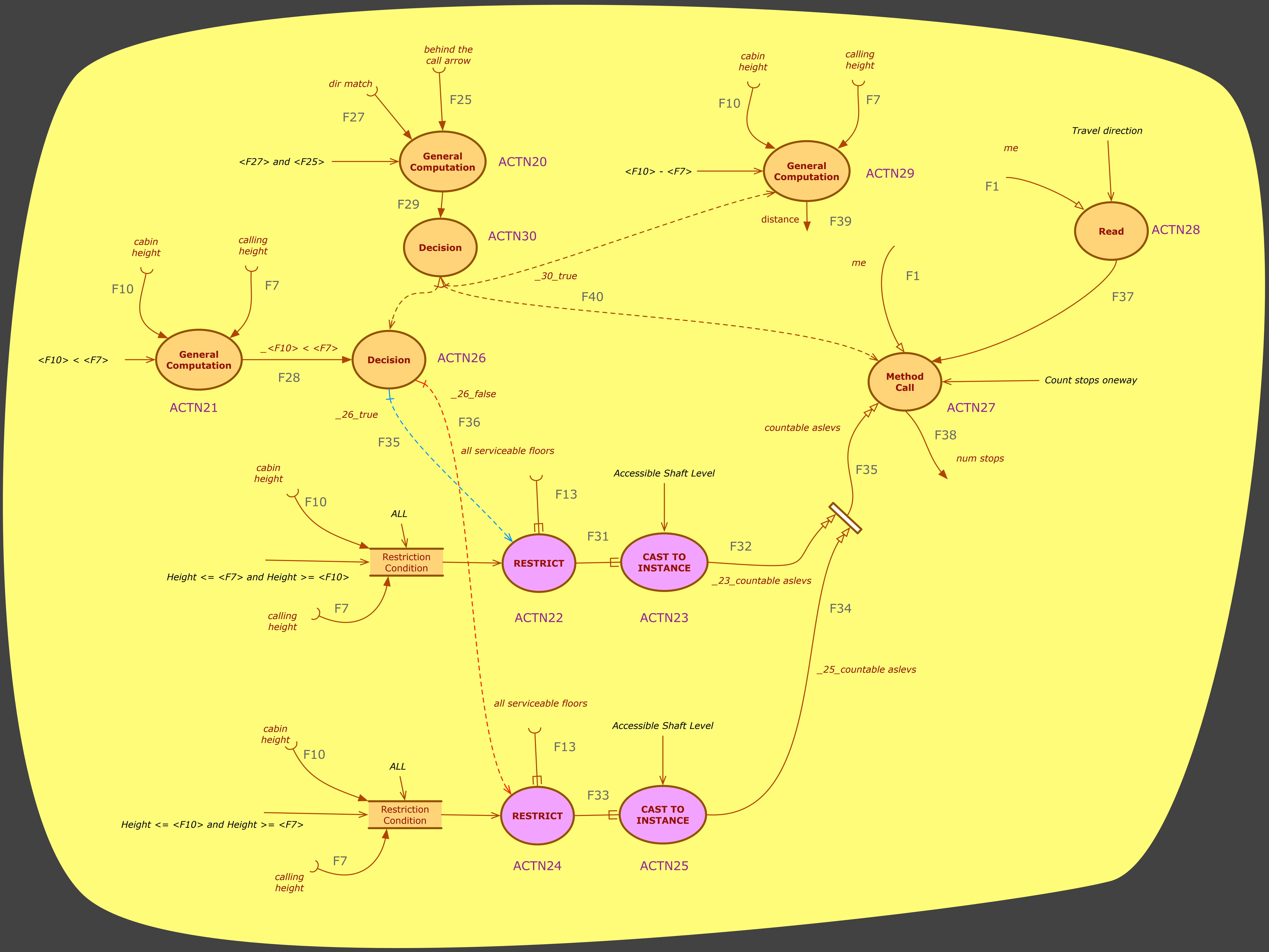
```
calling up, calling down = ^call dir == _up
cabin high, cabin low = cabin height > calling height

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



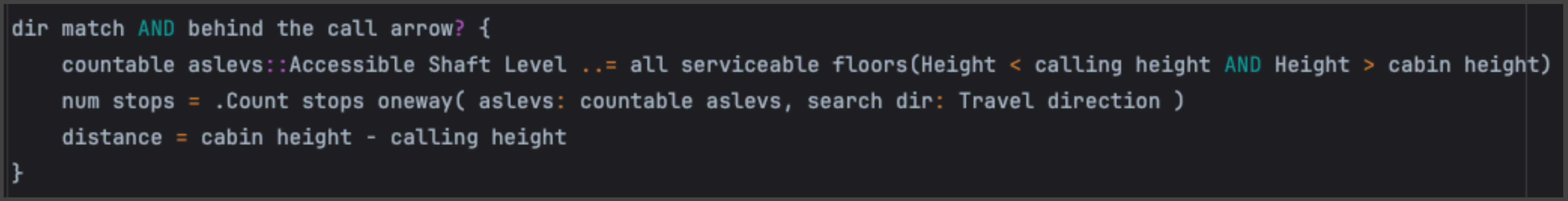
A7



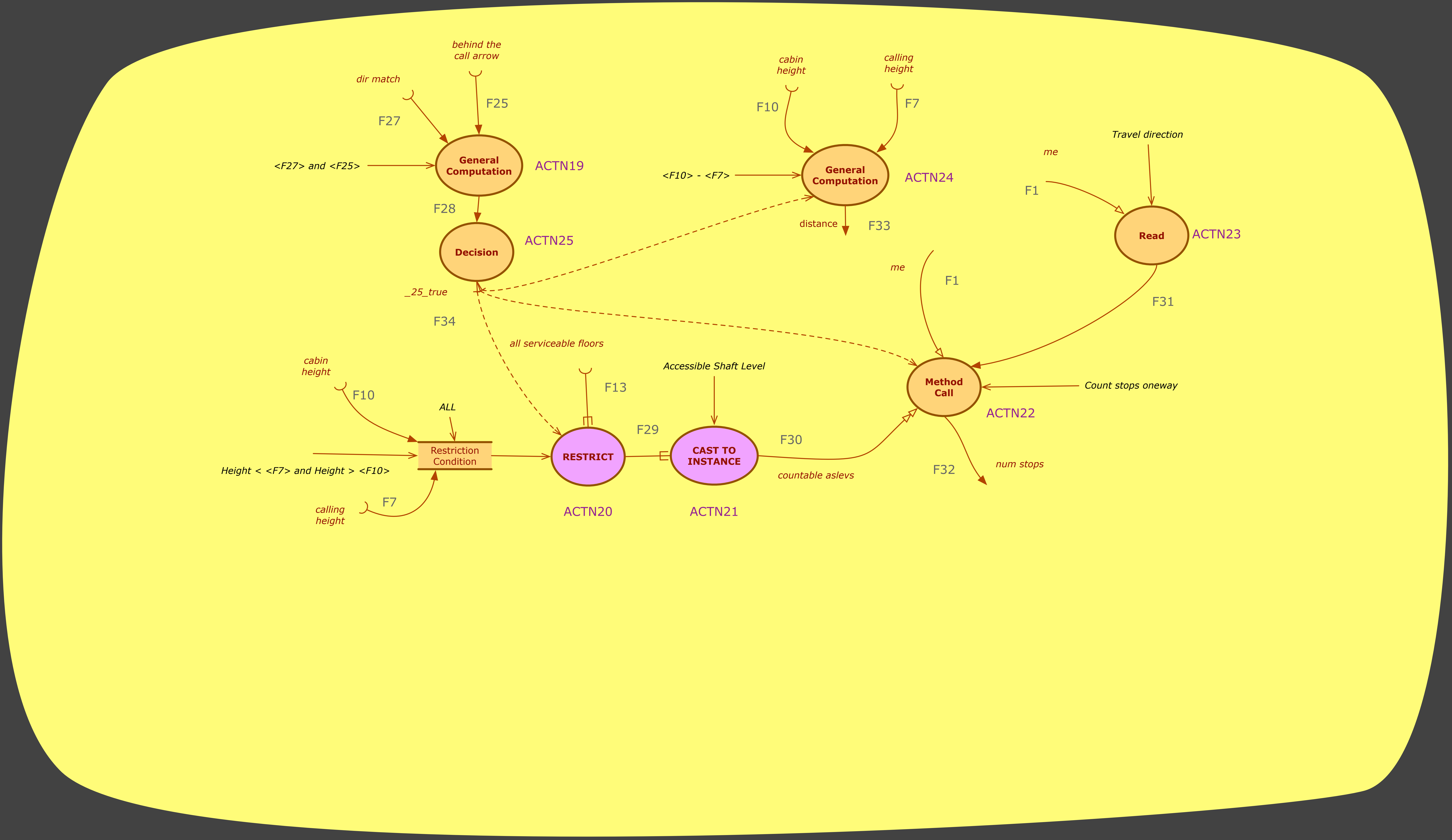
```
dir match AND behind the call arrow? {
    cabin height < calling height?
        countable aslevs::Accessible Shaft Level ..= all serviceable floors( Height <= calling height and Height >= cabin height ) :
        countable aslevs::Accessible Shaft Level ..= all serviceable floors( Height <= cabin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



A7

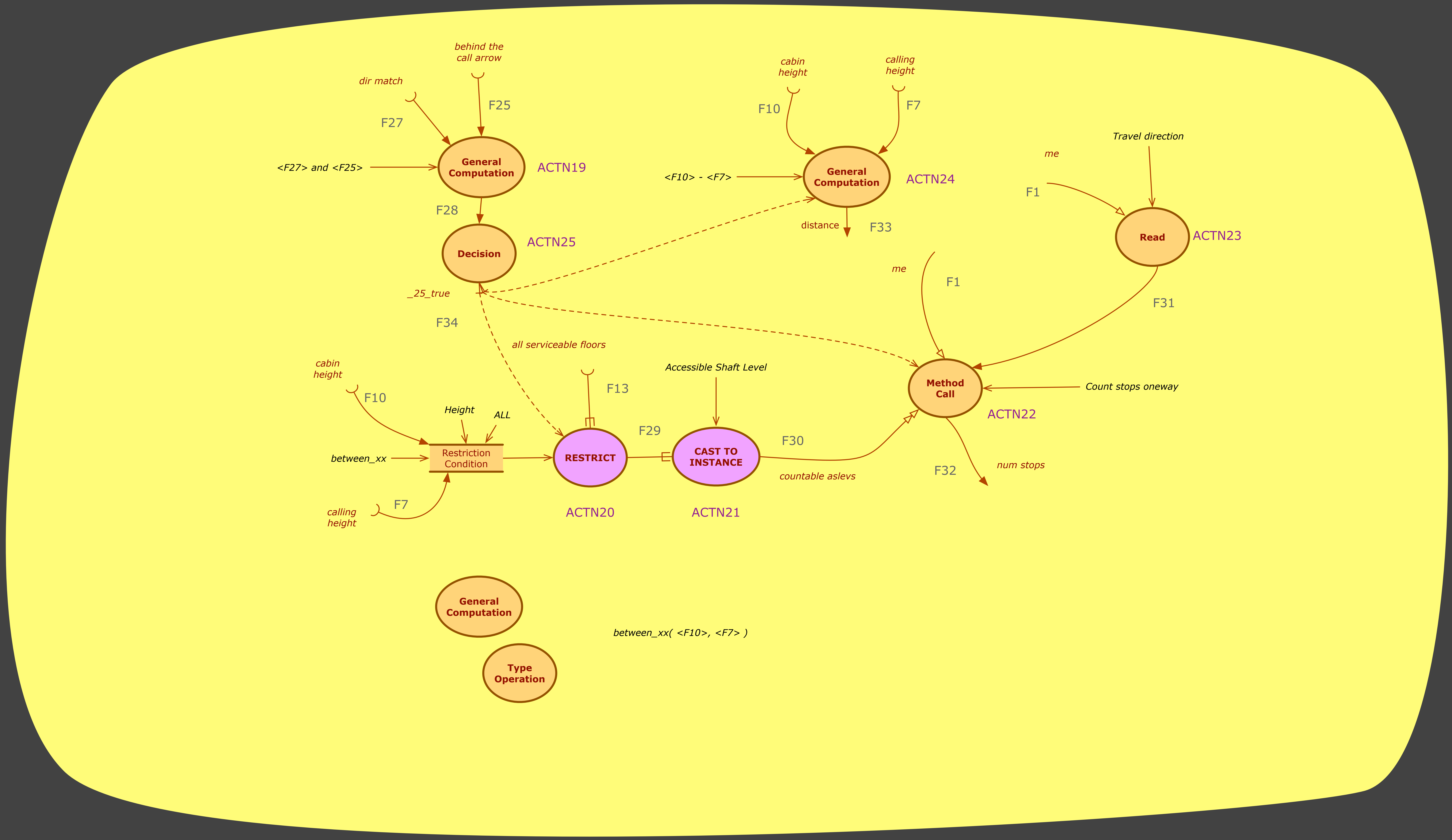


```
dir match AND behind the call arrow? {
    countable aslevs::Accessible Shaft Level ..= all serviceable floors(Height < calling height AND Height > cabin height)
    num stops = .Count stops oneway( aslevs: countable aslevs, search dir: Travel direction )
    distance = cabin height - calling height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



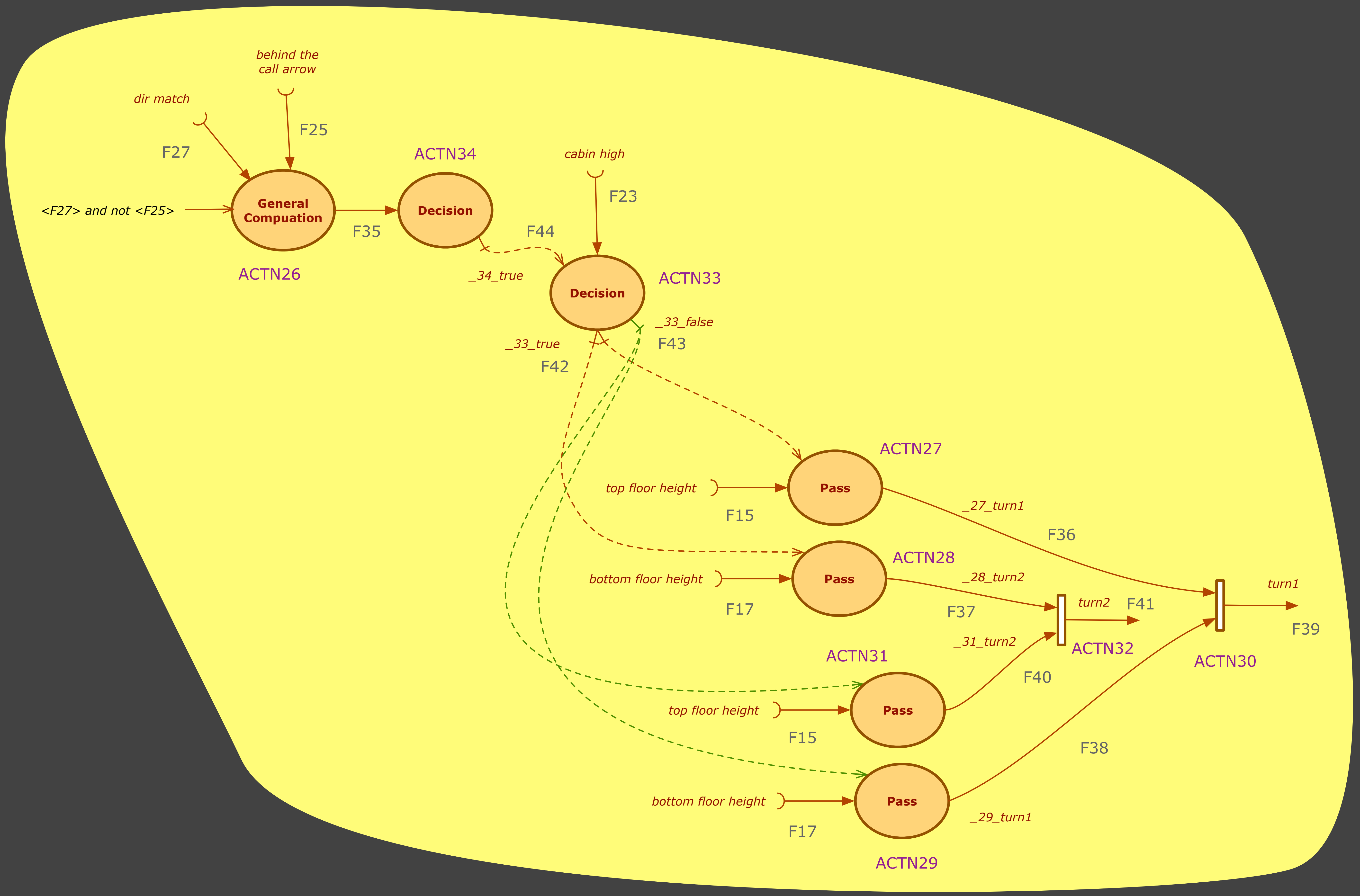
A7

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

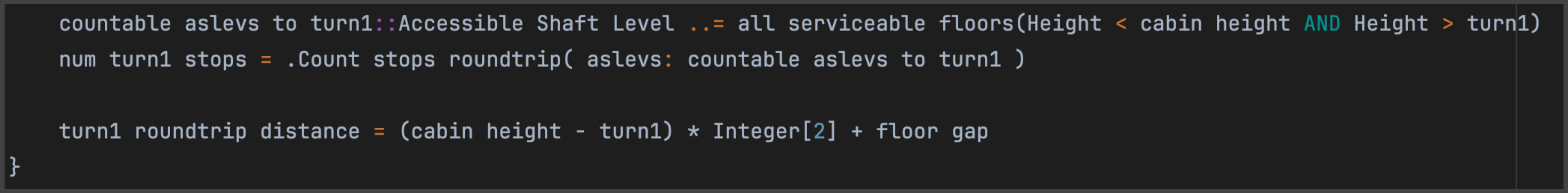


A7

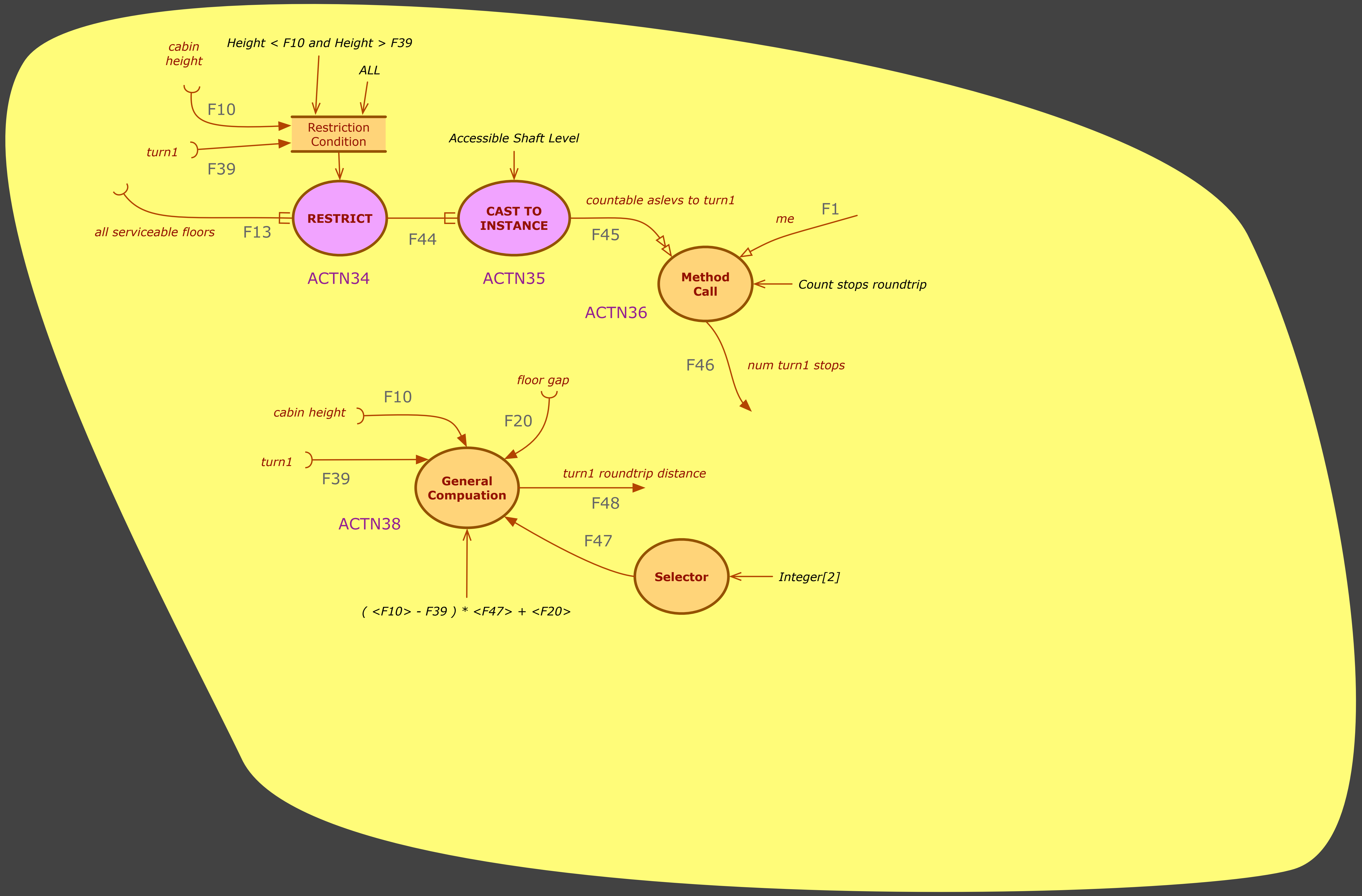
```
dir match AND NOT behind the call arrow? {
    cabin high ? turn1, turn2 = top floor height, bottom floor height :
        turn1, turn2 = bottom floor height, top floor height
```



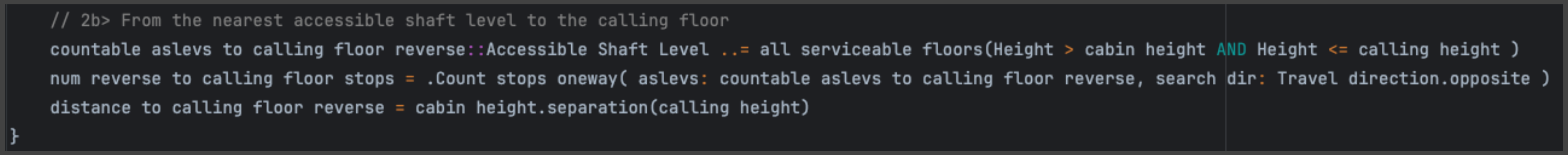
A7



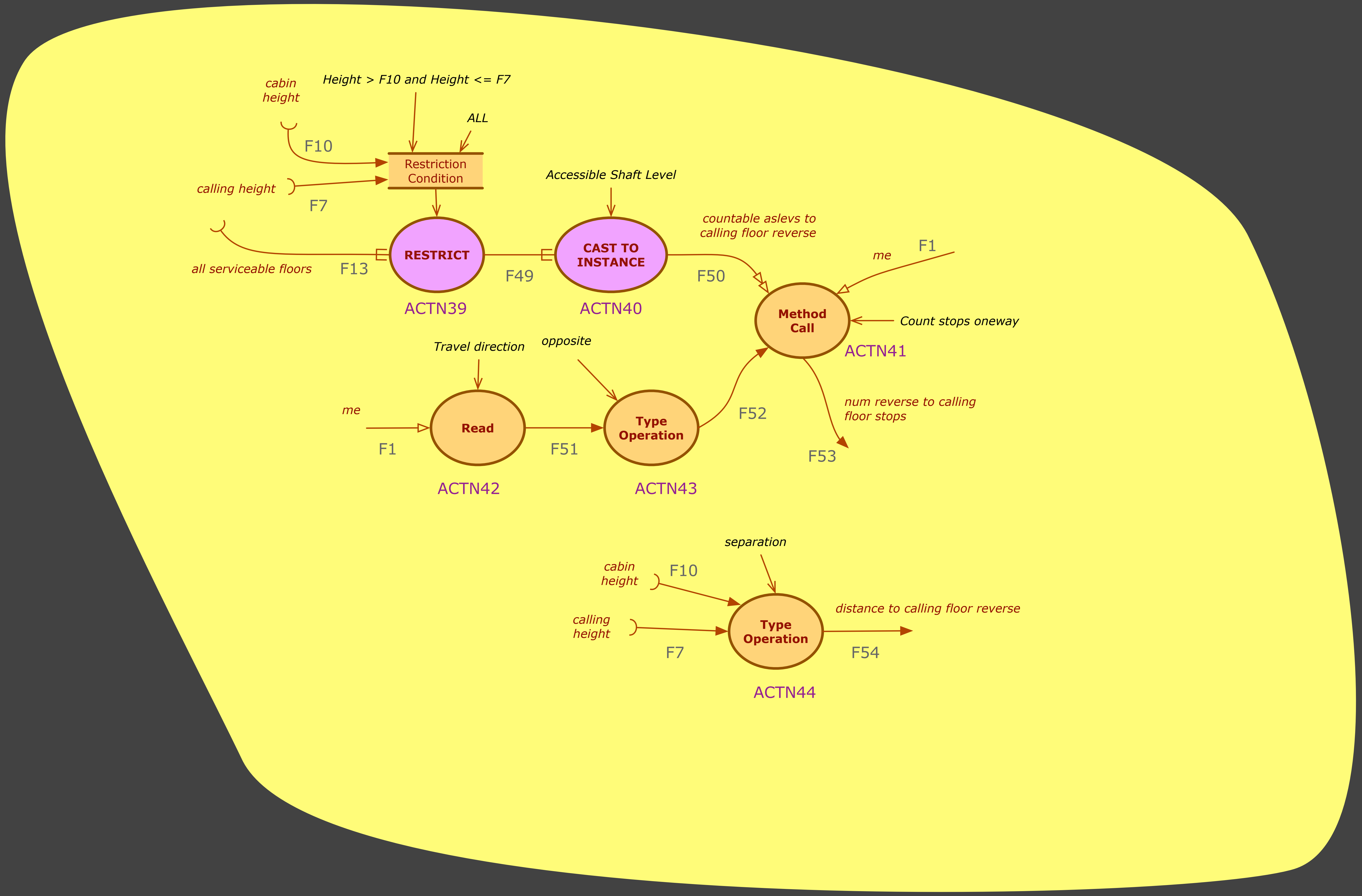
```
    countable aslevs to turn1::Accessible Shaft Level ..= all serviceable floors(Height < cabin height AND Height > turn1)
    num turn1 stops = .Count stops roundtrip( aslevs: countable aslevs to turn1 )

    turn1 roundtrip distance = (cabin height - turn1) * Integer[2] + floor gap
}
```



A7



```
    // 2b> From the nearest accessible shaft level to the calling floor
    countable aslevs to calling floor reverse::Accessible Shaft Level ..= all serviceable floors(Height > cabin height AND Height <= calling height )
    num reverse to calling floor stops = .Count stops oneway( aslevs: countable aslevs to calling floor reverse, search dir: Travel direction.opposite )
    distance to calling floor reverse = cabin height.separation(calling height)
}
```

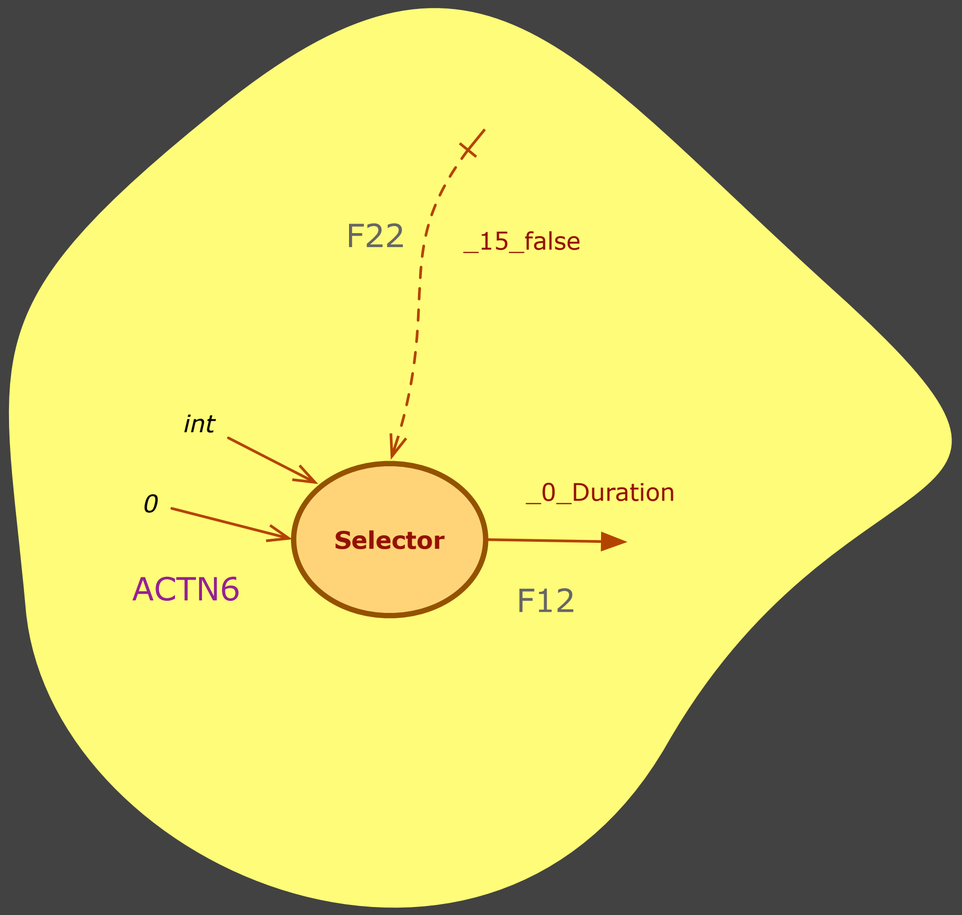


A7

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

●
●
●

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


F22
_15_false
int
0
Selector
_0_Duration
ACTN6
F12