Test_air_traffic_control

Author: Linjing Shen

Major: Computational Sience

- **Predicate**：A predicate is a **Boolean expression** that computes `True` or `False` based on input.
- `emergency` itself is a **single boolean value** that already represents `True` or `False` directly, without the need to calculate or combine other conditions.

Predicate:

```
P1 = runway_available and safe_speed and not emergency and safe_weather a
P2 = runway_available and safe_speed and not emergency and (acceptable_tr
P3 = emergency and priority_status
```

They are consist of several sub predicates as below:

p1:

```
runway_available = runway_clear or alternate_runway_available
```

p2:

```
safe_speed = plane_speed < landing_speed_threshold
```

p3:

```
not emergency
```

p4:

```
safe_weather = wind_speed <= max_wind_speed and visibility >= min_visibil
```

p5:

```
acceptable_traffic = airport_traffic <= max_air_traffic
```

p6:

```
traffic_override = priority_status and airport_traffic <= max_air_traffic
```

p7:

```
weather_override = priority_status and not safe_weather
```

Task 1: Active clauses

Active Clause Coverage (ACC): For each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses c_j , $j \neq i$ so that c_i determines p.

According to the definition, an active clause means that, with other conditions fixed, when the value of that condition changes, the boolean result of predicate will be changed as follow.

Thus, the three active clauses are:

- **airport_traffic** in **acceptable_traffic**
- **plane_speed** in **safe_speed**
- **runway_clear** in **runway_available**

Task 2: Test suite for CAC

Here is P1's test suite based on these three active clauses:

1. Clause: `runway_clear` in `runway_available`

```
runway_available = runway_clear or alternate_runway_available
```

Parameter	**t1 (True)**	**t2 (False)**
runway_clear	True	False
alternate_runway_available	False	False
plane_speed	100	100
emergency	False	False
wind_speed	20	20
visibility	2000	2000
airport_traffic	2	2
priority_status	False	False
Expected Result (p)	True	False

2. Clause: `plane_speed` in `safe_speed`

```
safe_speed = plane_speed < landing_speed_threshold
```

Parameter	**t1 (True)**	**t2 (False)**
runway_clear	True	True
alternate_runway_available	False	False
plane_speed	120	160

emergency	False	False
wind_speed	20	20
visibility	2000	2000
airport_traffic	2	2
priority_status	False	False
Expected Result (p)	True	False

3. Clause: `airport_traffic` in `acceptable_traffic`

```
acceptable_traffic = airport_traffic <= max_air_traffic
```

Parameter	**t1 (True)**	**t2 (False)**
runway_clear	True	True
alternate_runway_available	False	False
plane_speed	100	100
emergency	False	False
wind_speed	20	20
visibility	2000	2000
airport_traffic	4	6
priority_status	False	False
Expected Result (p)	True	False

Task 3: Inactive clauses

Inactive Clause Coverage (ICC): For each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses c_j , $j \neq i$ so that c_i does not determine p.

According to the above description, an inactive condition means that, with other conditions fixed, when the value of that condition changes, it does not change the value of the whole predicate.

Thus, the two inactive clauses are:

- **wind_speed** in **safe_weather**
- **airport_traffic** in **traffic_override**

Here is P1's test suite based on these two inactive clauses:

1. Clause: `wind_speed` in `safe_weather`

```
safe_weather = wind_speed <= max_wind_speed and visibility >= min_visibil
```

Fix `visibility` to `False`, make `safe_weather = False`, then the value of `safe_weather` will not change regardless of the value of `wind_speed`.

Parameter	t1 (True)	t2 (False)
runway_clear	True	True
alternate_runway_available	False	False
plane_speed	100	100
emergency	False	False
wind_speed	20	50
visibility	500	500
airport_traffic	2	2
priority_status	False	False
Expected Result (p)	False	False

2. Clause: `airport_traffic` in `traffic_override`

```
traffic_override = priority_status and airport_traffic <= max_air_traffi
```

Fix `priority_status` to `False` so that `traffic_override = False`, then the value of `traffic_override` will not change regardless of the value of `airport_traffic`.

Parameter	t1 (True)	t2 (False)
runway_clear	True	True
alternate_runway_available	False	False
plane_speed	100	100
emergency	False	False
wind_speed	50	50
visibility	500	500
airport_traffic	2	10
priority_status	False	False
Expected Result (p)	False	False

Differences between active and inactive clause test suites

Characteristics of the CACC Test Suite:

- For **active clauses**, changing the value of a clause will cause the value of the predicate `p` to change.
- CACC focuses on the correlation between conditions.
- Test cases are constructed for each identified active clause, ensuring that predicate results change when the clause takes true or false values. In `test_cacc_plane_speed`, scenarios are set up with aircraft speed within the safety threshold (`120`) and exceeding the threshold (`160`), comprehensively covering various impacts of speed conditions on

landing decisions and rigorously verifying the decision logic's accurate response to speed factor changes.

Characteristics of the Restricted Inactive Clause Test Suite:

- For **inactive clauses**, changing the value of a clause does not affect the value of the entire predicate.
- The restricted inactive clause test verifies that certain conditions are redundant and do not affect the result under fixed circumstances.
- For each inactive clause, specific scenarios are selected where changing the clause value maintains a constant predicate result while other conditions remain fixed. For example, in `test_inactive_visibility`, when the priority status is `False`, adjusting airport traffic flow values keeps the landing decision unchanged, thoroughly validating that airport traffic flow is irrelevant to the decision under this priority condition.