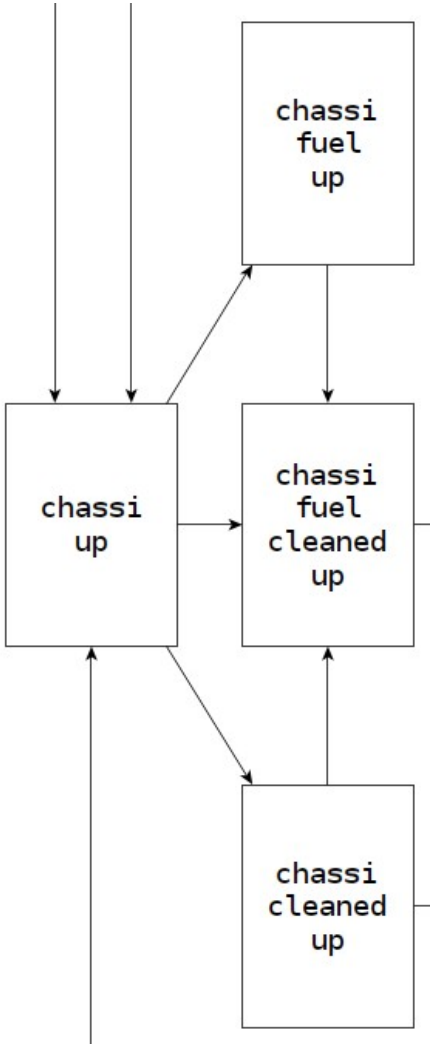
Структура Крипке

Безопасность авиаперевозок

Гончаров Марк
М01-305в

Пространство атомарных утверждений

fly	Самолёт летит
pilot	Пилот внутри самолёта
chassi	Шасси выдвинуты
fuel	Есть топливо
cleaned	Салон прибран
autopilot	Автопилот включён
up	Задание направления самолёту (локально для управления переменная)



Staff process

Рабочие, которые могут заменить топливо или почистить салон для следующего полёта

```
fair+ process Staff = "Staff"
begin StaffChange:
while (TRUE) do
 either
  if (~fly /\ ~fuel) then
     fuel := TRUE
  else
     skip;
  end if;
 or
  if (~fly /\ ~cleaned) then
     cleaned := TRUE
  else
     skip;
  end if;
 end either;
end while;
end process;
```

Pilot process

Он входит только в чистый салон, но может подождать, если рабочие ещё не залили топливо. Также он или авторабот посадят самолёт если уже пора и всё готово для этого (~up /\ chassi)

```
fair+ process Pilot = "Pilot"
begin PilotChange:
while (TRUE) do
 if (~pilot /\ ~fly /\ cleaned) then
    pilot := TRUE
 elsif (pilot /\ ~fly /\ fuel) then
    fly := TRUE
 elsif (fly /\ chassi /\ ~up /\ (pilot \/ autopilot)) then \* sit down
    fly := FALSE || pilot := FALSE || fuel := FALSE || cleaned := FALSE || up := TRUE || autopilot := FALSE
 else
    skip;
 end if;
end while;
end process;
```

Flight process

Мы можем просто сразу начать садить самолёт (up=false) или ещё убрать шасси, а потом их выдвинуть при посадке

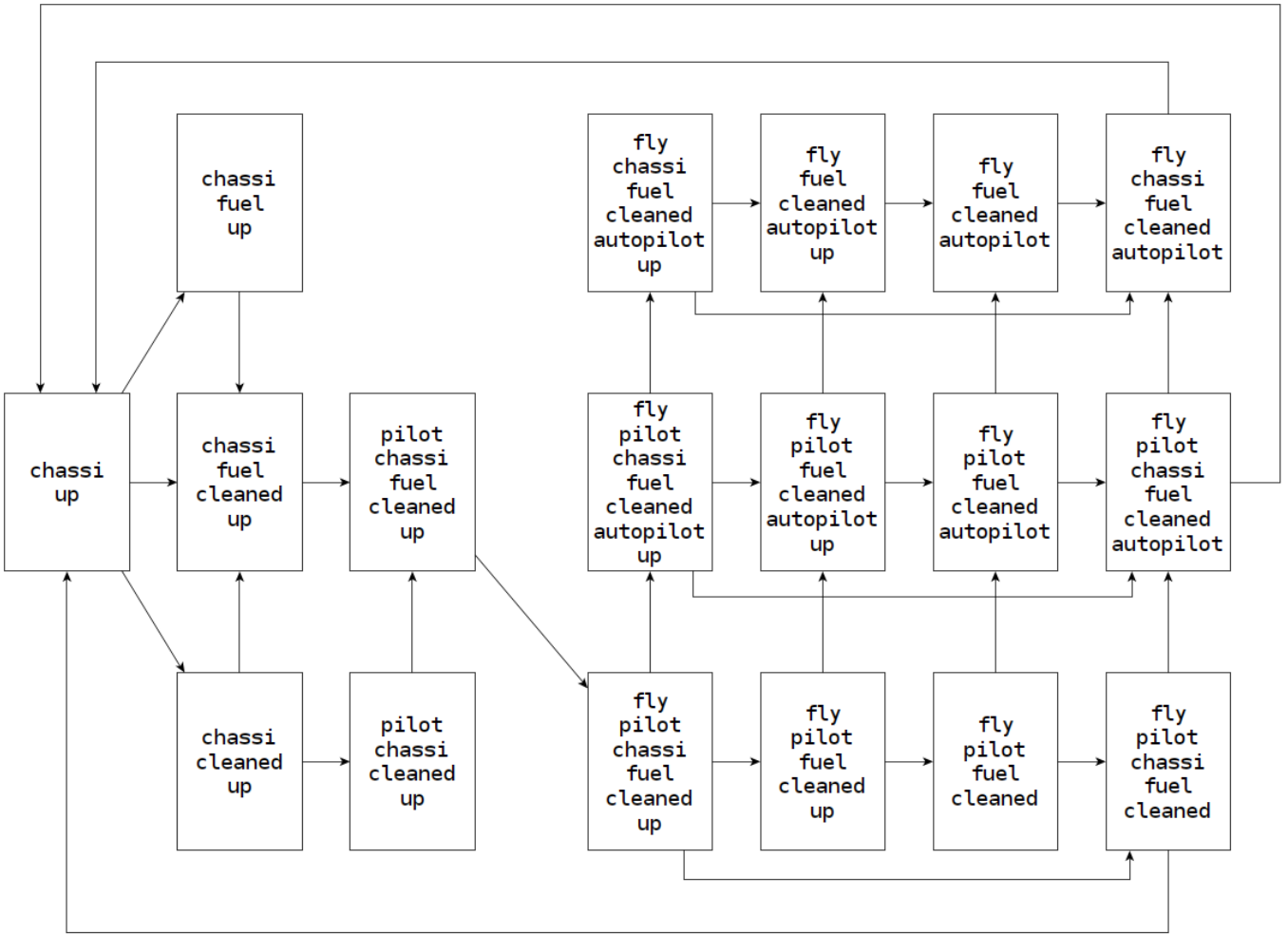
```
fair+ process Flight = "Flight"
begin FlightChange:
while (TRUE) do
 either
  if ((pilot \/ autopilot) /\ fly /\ chassi /\ up) then
     chassi := FALSE || up := FALSE
  elsif ((pilot \/ autopilot) /\ fly /\ ~chassi /\ ~up) then
     chassi := TRUE
  else
     skip;
  end if;
 or
  if ((pilot \/ autopilot) /\ fly /\ chassi /\ up) then
     up := FALSE
  elsif ((pilot \/ autopilot) /\ fly /\ ~chassi /\ ~up) then
     chassi := TRUE
  else
     skip;
  end if;
 end either;
end while;
end process;
```

Autopilot process

Пилот в полёте может выпрыгнуть (а может и нет), но обязательно до этого включив автопилот

```
fair+ process AutoPilot = "AutoPilot"
begin AutoPilotChange:
while TRUE do
 if (pilot /\ fly) then
    autopilot := TRUE
 elsif (autopilot /\ fly /\ pilot) then
    pilot := FALSE
 else
    skip;
 end if;
end while;
end process;
```

Пространство переходов



Свойство безопасности (safety)

Устанавливают, что нечто плохое, нежелательное, никогда не произойдет с системой

Свойство безопасности (safety)

1) На земле всегда нужно иметь шасси
2) В полёте не может закончится топливо
3) Пилот не заходит в нечищенную кабину
4) Кто-то всегда контролирует самолёт в движении – пилот или автопилот

```
define
stands == ~fly => chassi
enough_fuel == ~fly \/ fuel
clean_check == pilot => cleaned
somebody_control == fly => (pilot \/ autopilot)
end define;
```

Свойство безопасности (safety)

Пример нарушения (2):

кто-то мог неправильно посчитать кол-во бензина, необходимого для полёта и он кончился раньше

```
define
stands == ~fly => chassi
enough_fuel == ~fly \/ fuel
clean_check == pilot => cleaned
somebody_control == fly => (pilot \/ autopilot)
end define;
```

Свойство живости (liveness)

Устанавливают, что при некоторых условиях нечто "хорошее" в конце концов обязательно произойдет при любом развитии процесса.

Свойство живости (liveness)

1) Если пилот сел, то когда-нибудь самолёт полетит (пилот не может выйти и прити – код запрещает)
2) Каждый полёт окончится, самолёт не застрянет
3) Шасси мы задвигает когда-нибудь (для этого fair+)

```
start_fly == [](pilot ~> fly)
end_fly == [](fly ~> ~fly)
autopilot_sometimes == <>(autopilot)
```

Свойство живости (liveness)

Пример нарушения (3):
Пилот принципиально ездит всегда с выдвинутым шасси

```
start_fly == [](pilot ~> fly)
end_fly == [](fly ~> ~fly)
autopilot_sometimes == <>(autopilot)
```