Sémantique d'Esterel

exec(t,P) = (t',E,b)

t : instruction de départ
t' : instruction d'arrivée
P : signaux présents
E : signaux émis
b : booléen de terminaison

Séquence :

exec(`t;u`,P) =
soit exec(`t`,P) = (`t'`,E,b) ;
si b = faux alors (`t';u`,E,faux) sinon
(`u'`,E+E',b'), où exec(`u`,P) = (u',E',b')

Sémantique - 2

Parallélisme :

exec(t||u,P) = (t'||u' ,E_1+E_2,b_1 et b_2)

où exec(t,P) = (t' ,E_1,b_1) et exec(u,P) = (u' ,E_2,b_2)

Pause et nothing :

exec(nothing,P) = (nothing,ø,vrai)
exec(pause,P) = (nothing ,ø,faux)

Loop :

exec(loop t end,P) = exec(t;loop t end ,P)

Emit :

exec(emit S,P) = (nothing ,{S},vrai)

Sémantique - 3

Présence :

exec(`present S then t else u end`,P) =
si S dans P alors exec(`t`,P), sinon exec(`u`,P)

Préemption :

exec(`abort t when S`,P) = (`abort t' when immediate S`,E,b)
où exec(`t`,P) = (`t'`,E,b)

exec(`abort t when immediate S`,P) =
exec(`present S else abort t when S end`,P)

Sémantique - 4

Signaux :

exec(`signal S in t end`,P) =

- **Sol = ø ;**
- **soit exec(`t`,P + {S}) = (`t'`,E,b) ;**
  si S dans E alors mettre (`signal S in t' end`,E-{S},b) dans Sol ;
- **soit exec(t,P - {S}) = (`t'`,E,b) ;**
  si S non dans E, alors mettre (`signal S in t' end`,E,b) dans Sol ;
- **si Sol ne contient qu'un seul élément, alors retourner celui-ci;**
- **sinon, erreur.**

hypothèses vérifiées

Programme correct

t =
```
signal S in
   present S then emit T end
||
   emit S
end
```

- **exec(**`present S then emit T end || emit S`**,{S}) =**
  (`nothing||nothing`**,{S,T},vrai)** **ok**
- **exec(**`present S then emit T end|| emit S`**,ø) =**
  (`nothing||nothing`**,{S},vrai)** **nok**

exec(t,ø) = (`signal S in nothing||nothing end`**,{T},vrai)**

Non déterminisme

```
t = signal S in
      present S then emit S end
    end
```

- **exec(**`present S then emit S end`**,{S}) = exec(** `emit S`**,{S}) =** **ok**
(`nothing`**,{S},vrai)**

- **exec(**`present S then emit S end`**,ø) = exec(** `nothing`**,ø) =** **ok**
(`nothing`**,ø,vrai)**

exec(t,ø) a deux solutions : erreur !

Absence de solution

t =
```
signal S in
   present S else emit S end
end
```

- exec(`present S else emit S end`,{S}) = exec(`nothing`,{S}) = (`nothing`,ø,vrai) **nok**
- exec(`present S else emit S end`,ø) = exec(`emit S`,{S}) = (`nothing`,{S},vrai) **nok**

exec(t,ø) a zéro solutions : erreur !

Solutions non-causales...

t =
```
signal S in
   present S then emit T end;
   emit S
end
```

Puisque :

- **exec(`present S then emit T end`,{S}) = (`nothing`,{T},vrai)**
- **exec(`emit S`,{S}) = (`nothing`,{S},vrai)**

Alors :

exec(`present S then emit T end;emit S`,{S}) = (`nothing` ,{S,T},vrai)

Donc :

exec(t,ø) = (`signal S in nothing end`,{T},vrai)

C'est la seule solution : le programme est accepté

Conclusion

Sémantique “comportementale”

- **abstraite (*macro-step*) : un appel de exec = un instant**
- **simple (“structurelle”)**

mais

- **n’exprime pas comment trouver la solution**
- **accepte des programmes non causaux**
- **inefficace**

Conclusion - 2

Réaction retardée à l'absence (instant suivant) permet d'éliminer les problèmes de causalité

Privilégier les solutions où le signal est absent (pour éliminer les solutions non-causales)

Sémantique "concrète" exprimant comment calculer la solution

Programmation réactive