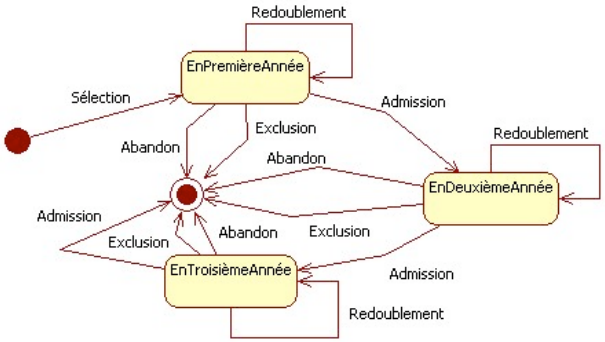
Chapitre 5 : Diagrammes états-transitions
Éléments de correction

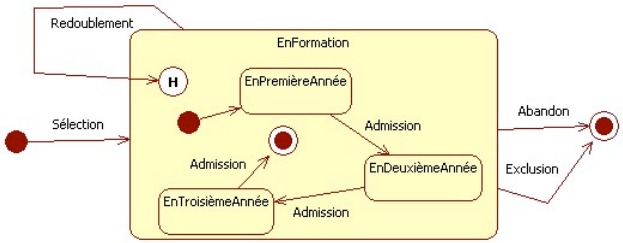
Il s'agit d'UN corrigé. D'autres réponses sont possibles, notamment là où une modélisation est demandée.

Exercice 5.1

a) Une réponse à cette question nous est fournie par l'automate suivant :



b) Le fait d'utiliser un automate hiérarchique va nous permettre de simplifier l'apparence de notre réponse, en recourant à un état-mémoire H, chargé de conserver le nom de l'état quitté lors de l'arrivée de l'événement *Redoublement* et en utilisant la factorisation de l'événement *Abandon* :

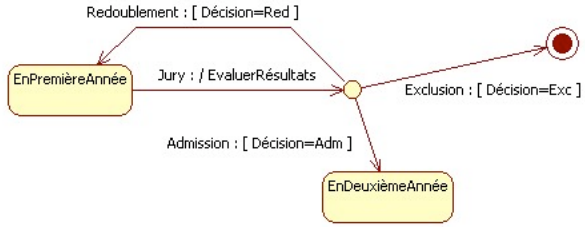


Quelque soit l'état atteint, *Abandon* provoque le passage à l'état final.

c) Le travail du jury puis l'analyse de la décision et, en fonction de celle-ci, la transition vers un état ou un autre, tout ceci doit faire penser à un point de jonction dynamique :

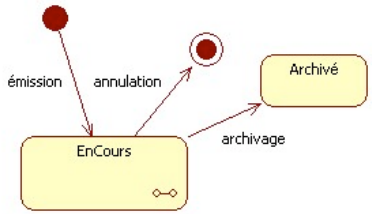
Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)

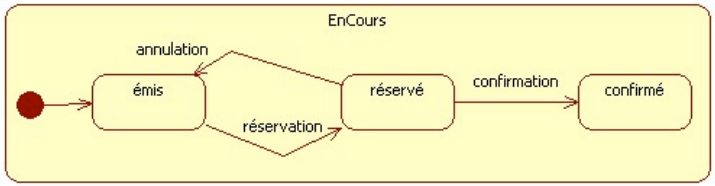


Exercice 5.2

L'évolution de l'état d'un billet est décrite par le diagramme états-transitions suivant :



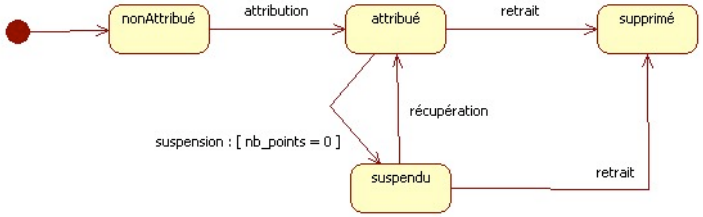
la petite icône contenue dans l'état *EnCours* signifiant que cet état est en fait un automate, un super-état. Ce super-état ressemble à ceci :



Nous aurions pu tout « concentrer » en un seul schéma, analogue à celui que nous avons réalisé pour l'exercice précédent.

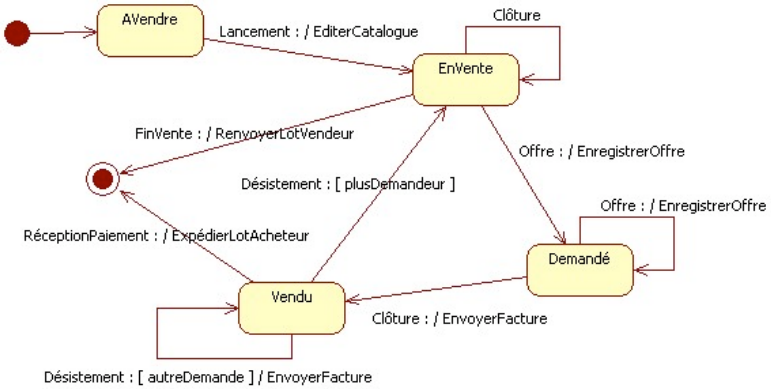
Exercice 5.3

L'évolution de l'état d'un permis est décrite par le diagramme états-transitions suivant :

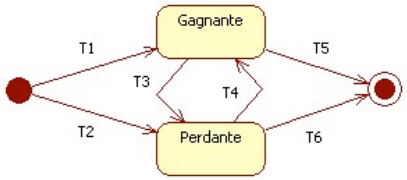


Exercice 5.4

a) Une réponse à la question est fournie par le diagramme états-transitions suivant :



b) La vie d'une offre (elle est numérotée 1 dans le schéma ci-après) est la sivante :



Les différentes transitions ont la signification suivante :

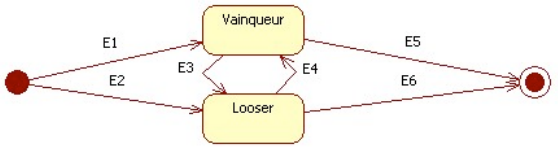
Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)

T1 : [Prix > PrixMax] Offre
T2 : [Prix ≤ Prixmax] Offre
T3 : [PrixOffre1 < PrixOffre2] Offre
T4 : Désistement
T5 : Clôture
T6 : Clôture

Le désistement évoqué dans la transition T4 est celui d'un collectionneur vainqueur différent de celui qui a fait l'offre dont on décrit la vie. Celle-ci (Offre 1) était le seconde (elle avait un prix juste inférieur de celui de l'offre 1).

c) La vie d'un acheteur potentiel se décrit ainsi :



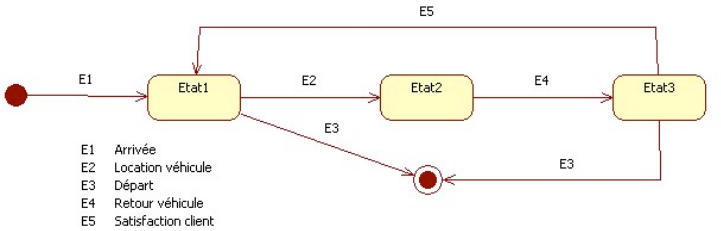
Les étiquettes ont la signification suivante :

E1 : [Prix > PrixMax] Offre
E2 : [Prix ≤ PrixMax] Offre
E3 : [PrixOffre 1 < PrixOffre2] Offre
E4 : Désistement
E5 : Clôture
E6 : RéceptionFacture / PayerFacture

Le désistement est l'œuvre d'un autre collectionneur que celui dont on décrit la vie. L'offre 2 est faite par un autre collectionneur.

Exercice 5.5

a) Une première réponse à la question réside dans l'automate suivant :



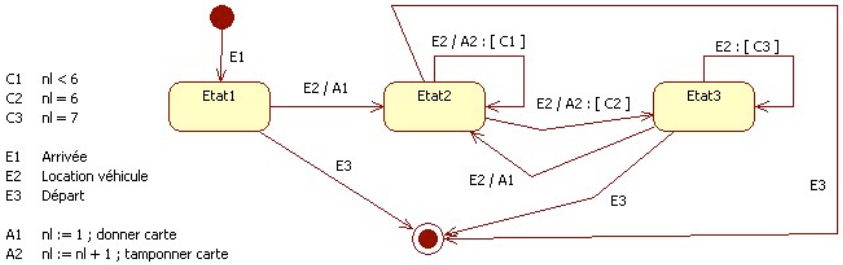
Les trois états que nous avons mis en évidence correspondent aux cas suivants :

- état 1 : le client veut louer un véhicule ; il est en train de préparer sa location (il remplit les papiers).
- état 2 : le client utilise le véhicule qu'il a loué.
- état 3 : le client a rendu le véhicule ; il est en train de finir sa location (il paye).

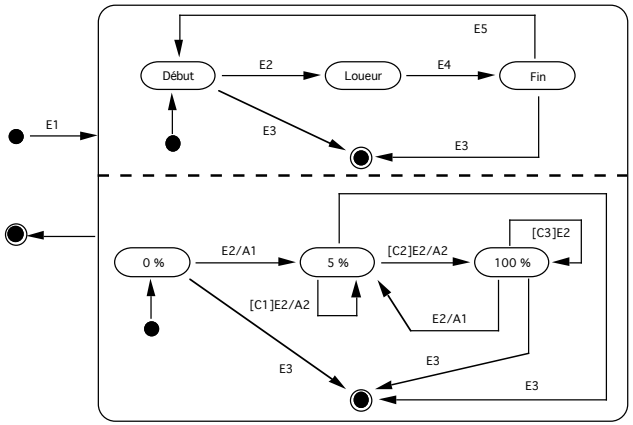
b) Nous pouvons mettre en évidence cinq états, un état initial (non nommé), un état final (non nommé) et trois états intermédiaires. Ces derniers correspondent aux cas où le client :

- ne bénéficie pas de la réduction de 5 % (état 1),
- bénéficie de la réduction de 5 % et d'une carte (état 2),
- a gagné le droit à une location gratuite (état 3).

L'automate correspondant est le suivant :



c) La réponse se situe au niveau du parallèlisme :



Remarque StarUML Nous n'avons pas utilisé le logiciel StarUML pour « faire » cette figure du fait du parallèlisme, mais un logiciel de dessin appelé MacDraw. Il est tout à fait possible de définir avec StarUML des états concurrents (il y a une propriété prévue pour

Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)

cela), mais l'effet de cette définition n'apparaît pas sur le schéma. On notera également la différence de notation entre les schémas produits. Avec StarUML, les gardes sont placées en fin d'étiquettes de transitions ; avec MacDraw, les gardes sont placées au début.

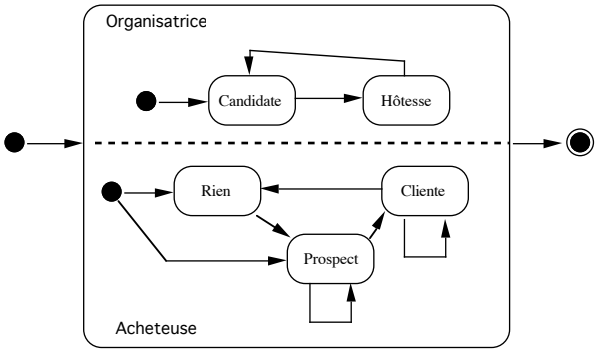
Exercice 5.6

Pour définir le type de diagramme nécessaire, il nous faut d'abord recenser les états par lesquels passe une personne. Ceux-ci sont au nombre de quatre : hôtesse, candidate, cliente et prospect. Plusieurs « règles » contrôlent les transitions :

- une personne peut être hôtesse sans avoir été prospect ou cliente.
- une hôtesse est d'abord candidate.
- une hôtesse peut être cliente ou prospect.
- une personne peut ne jamais avoir été hôtesse.

Les combinaisons autorisées sont multiples : {candidate}, {candidate et prospect}, {candidate et cliente}, {hôtesse}, {hôtesse et prospect}, {hôtesse et cliente}).

Le type de diagramme états-transitions à utiliser est donc (presque) une évidence : il faut un automate parallèle, avec d'un côté hôtesse et candidate et de l'autre prospect et cliente. Le résultat, pour ce qui est de la structure proprement dite, est alors le suivant :



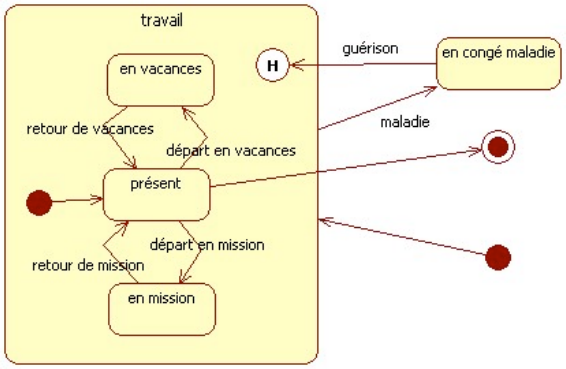
Remarque StarUML Nous n'avons pas utilisé le logiciel StarUML pour « faire » cette figure du fait du parallèlisme, mais un logiciel de dessin appelé MacDraw. Il est tout à fait possible de définir avec StarUML des états concurrents (il y a une propriété prévue pour cela), mais l'effet de cette définition n'apparaît pas sur le schéma.

Nous laissons aux étudiants lisant ce corrigé le soin de donner un nom aux diverses transitions.

Exercice 5.7

a) Une maladie peut se déclarer n'importe quand. Cette hypothèse donne la clé. Elle impose, à elle seule, le recours à un automate parallèle (avec une partie comprenant la maladie et une partie les trois autres états) ou un automate

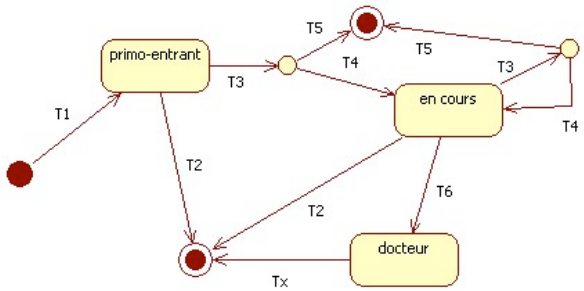
hiérarchique (avec un super-état regroupant les trois états, *en mission*, *en vacances*, *présent*). Nous préférons cette seconde solution :



Les actions seront associées à certaines transitions, comme ceci :

maladie / envoyer feuille maladie
retour de mission / faire compte-rendu de mission
départ en vacances / sauvegarder données

b) Le passage d'un doctorant au laboratoire se modélisera ainsi :



Les étiquettes sur les transitions correspondent à ceci :

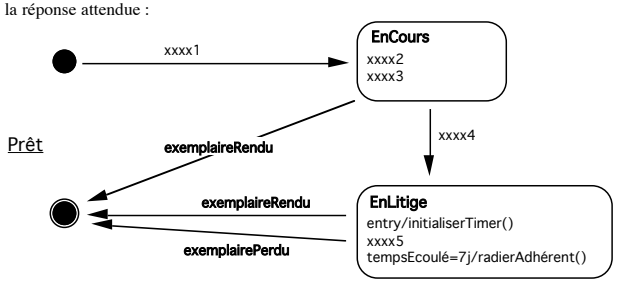
T1 : inscription / délivrer carte étudiant
T2 : abandon / rédiger démission
T3 : fin année / évaluer travail
T4 : [OK] / délivrer carte étudiant

T5 : [NON]
T6 : soutenance / payer un pot

La transition Tx n'est pas prévue dans notre énoncé. Elle peut être associée à un départ du laboratoire, à un rectrutement comme chercheur…

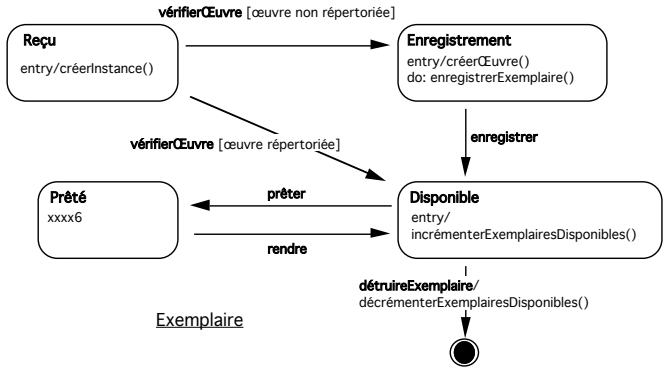
Exercice 5.8

Voici la réponse attendue :



Les cinq étiquettes prennent les valeurs suivantes :

xxxx1 : initialiserPrêt (unExemplaire : Exemplaire)
xxxx2 : entry / décrémenterExemplairesDisponibles ()
xxxx3 : entry / incrémenterNbEmprunts ()
xxxx4 : miseEnLitige ()
xxxx5 : envoyerAvertissement ()

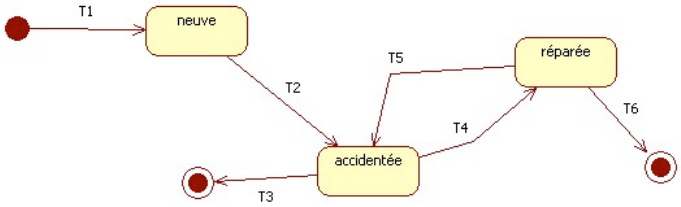


L'étiquette prenant la valeur suivante : xxxx6 : décrémenterExemplairesDisponibles ()

Remarque StarUML Pour réaliser ce schéma, nous avons renoncé à utiliser le logiciel StarUML. Nous avons rencontré, en effet, deux problèmes :
- le premier tient au fait qu'il n'est pas possible avec StarUML d'associer une opération à un événement. Les seules situations envisagées correspondent aux étiquettes *entry*, *do* et *exit* ; *on event* n'est pas prévu.
- le second est dû au type d'exercice qui est proposé. Il s'agit de remplacer des opérations anonymes, matérialisées par des xxxx, par des méthodes évoquées dans le texte. La pratique normale, avec StarUML, est de sélectionner une des opérations de la classe et de l'associer à la transition. Nous aurions dû « remplir » la classe d'opérations xxxx1, xxxx2 *ce* que nous n'avons pas souhaité faire.

Exercice 5.9

a) Pour répondre plus facilement aux deux premières questions, donnons un nom aux différentes transitions :



Ceci fait, nous pouvons apporter une réponse à la première question :

T1 : achat
T2 : accident
T3 : vente
T4 : réparation
T5 : accident
T6 : vente

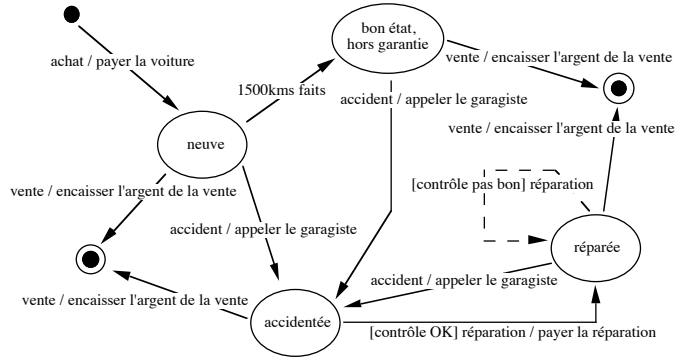
b) puis à la deuxième :

T1 : achat / payer la voiture
T2 : accident / appeler le garagiste
T3 : vente / encaisser l'argent de la vente
T4 : réparation / payer la réparation
T5 : accident / appeler le garagiste
T6 : vente / encaisser l'argent de la vente

c) L'automate ci-dessous correspond à ce qui était attendu :

Projet ENRC-DVD
MIAGE

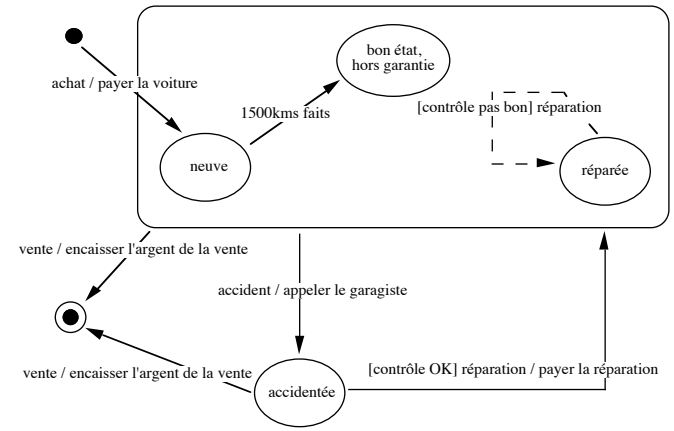
Module UML
(Outils de modélisation)



Remarque StarUML Nous n'avons pas utilisé le logiciel StarUML pour « faire » cette figure, mais un logiciel de dessin appelé MacDraw. La transition « autour » de l'état réparée, en boucle, est en effet mise en évidence sous forme d'un trait pointillé. Cette mise en évidence spéciale ne peut se faire avec StarUML… ou, du moins, n'avons-nous pas trouvé comment faire…

NB : la boucle en pointillés peut ne pas être figurée.

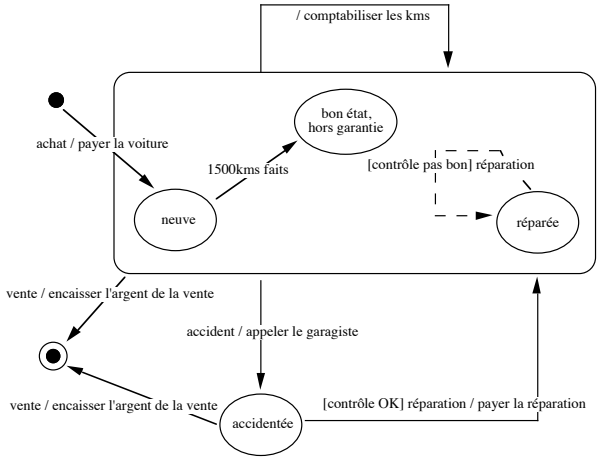
d) Une voiture peut à tout moment être vendue ou avoir un accident. Il faut donc placer un super-état et deux transitions sortantes :



Remarque StarUML Nous n'avons pas utilisé le logiciel StarUML pour « faire » cette figure, mais un logiciel de dessin appelé MacDraw. La transition « autour » de l'état

réparée, en boucle, est en effet mise en évidence sous forme d'un trait pointillé. Cette mise en évidence spéciale ne peut se faire avec StarUML… ou, du moins, n'avons-nous pas trouvé comment faire…

e) Le plus simple est de placer cette transition « sur » le super-état :



Remarque StarUML Nous n'avons pas utilisé le logiciel StarUML pour « faire » cette figure, mais un logiciel de dessin appelé MacDraw. La transition « autour » de l'état réparée, en boucle, est en effet mise en évidence sous forme d'un trait pointillé. Cette mise en évidence spéciale ne peut se faire avec StarUML… ou, du moins, n'avons-nous pas trouvé comment faire…

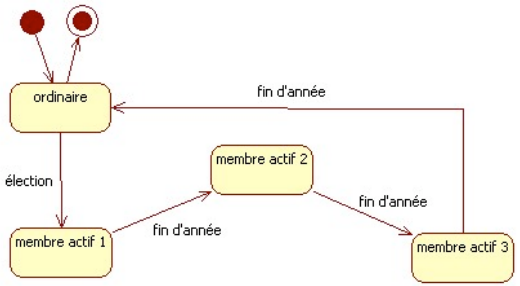
f) Un automate de cette sorte correspond à un produit d'automates. La vie de cette voiture passe donc par des états composés d'un état « du haut » et d'un état « du bas ». La voiture peut donc être réparée en marche !

Exercice 5.10

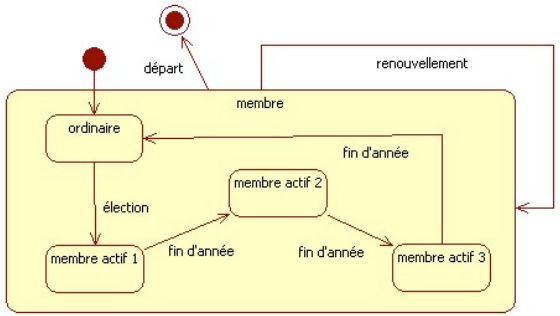
Le texte peut conduire aux « états » suivants : *membre*, *membre actif*, *membre du bureau*, *sortant*. Une brève analyse conduit à abandonner l'état *membre du bureau*, plus proche d'un type. Le diagramme états-transitions définissant l'évolution d'un membre est donc le suivant :

Projet ENRC-DVD
MIAGE

Module UML
(Outils de modélisation)



Chaque membre dans un des états actifs est 1, 2, 3 est dans d'un sous-type *Pres*, *Vpres1*, *Vpres2*… *SecrAdj* ou *Simple*. L'arrivée et le départ d'un membre ne sont pas spécifiés dans le sujet, mais on peut imaginer l'automate ci-après :



----------fin du texte----------