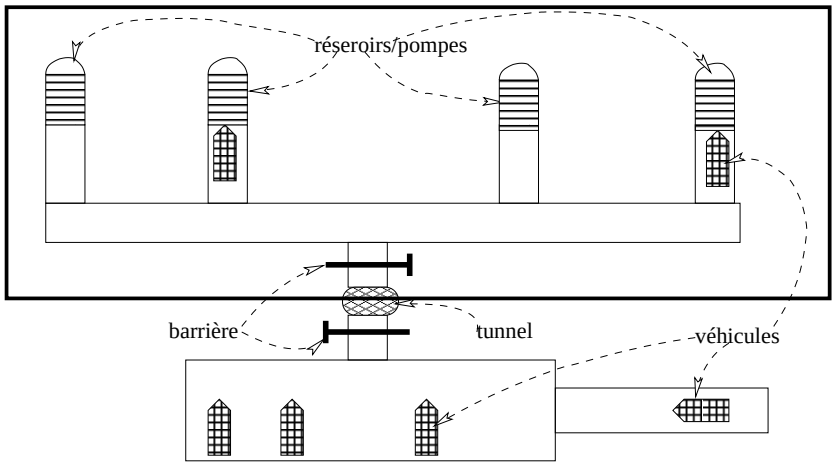
Master 1 Informatique. ***Langages de spécification de logiciel***

Gestion d'une station service

Une station service possède des pompes avec des carburants différents. Les réservoirs de carburant se trouvent dans une zone où on peut accéder par un tunnel avec une voie. Devant le tunnel il y a un espace d'attente qui est accessible par une route aussi avec une seule voie. Chaque véhicule qui arrive demande un certain type de carburant.



Pour s'approvisionner en carburant un véhicule approche de l'entrée de l'espace d'attente et envoie un signal qui donne son nom (qui est unique). Pour le contrôleur ce nom est la valeur de l'entrée $arrive :\rightarrow (NomV \cup \{undef\})$. Le contrôleur ajoute cette information dans la liste d'attente. La valeur du signal *arrive* est égale au nom du véhicule qui arrive pendant une certaine durée lorsque le véhicule est sur la voie d'arrivée. Après cette durée, et sûrement avant l'arrivée d'un autre véhicule, le signal devient $undef$. S'il y a une pompe disponible et le tunnel est vide alors le contrôleur cherche le premier élément de la liste d'attente dont le type de carburant est approprié et envoie un signal $go(m, p)$ au véhicule m pour qu'elle aille à la pompe p et envoie un signal *open* pour ouvrir la barrière. Ce signal *open*, ainsi que le signal *go*, doit durer un temps Δ. Les noms de véhicules constituent une sorte $NomV$.

La sortie des véhicule de la zone d'approvisionnement est gérée par un autre contrôleur et cette question ne concerne pas le contrôleur à décrire. L'interaction est assurée par le signal $tFree$ qui dit que le tunnel est vide et notre contrôleur peut ouvrir la barrière s'il veut. Une autre entrée du contrôleur est $pDisp$. $pDisp(p) = true$ dit que la pompe p est disponible. Le type de carburant d'une pompe est donnée par la fonction $typeP$, et le type de carburant d'un véhicule est fourni par la fonction $typeM$. Les types de carburant constituent la sorte $TypesCarb$.

On suppose que le contrôleur dispose des opérations suivantes sur la liste d'attente (nous n'avons pas besoin d'expliciter cette liste) : $add(x)$ (ajoute l'élément x à la fin de la liste), $delete(x)$ (enlève l'élément x de la liste), $numVéh(m)$ (le numéro du véhicule m dans la liste s'il est là, sinon $undef$), $nomVéh(k)$ (le nom du $k$ème véhicule dans la liste d'attente si la liste contient au moins k éléments et $undef$ sinon). Remarquons que $nomVéh(numVéh(m)) = m$ si $numVéh(m) \neq undef$ et $(numVéh(nomVéh(k))) = k$ si $nomVéh(k) \neq undef$. La liste d'attente n'apparaît pas explicitement, donc nous n'introduisons pas une sorte pour les listes de noms de véhicules.

Si plusieurs pompes avec le carburant demandé sont disponibles le contrôleur envoie le véhicule à la pompe ayant le numéro minimal (le contrôleur dispose des fonctions mathématiques habituelles sur les ensembles finis d'éléments que nous considérons, en particulier, d'une fonction min). Il ne faut pas envoyer un véhicule dans le tunnel, même s'il est libre, si le signal *go* est valable pour un autre véhicule.

1. Décrivez une ASM en utilisant seulement des **if-then**-opérateurs exécutés en parallèle. Le vocabulaire de cette ASM doit être aussi donné.
2. Décrivez les requis sur le système en français, en utilisant le vocabulaire temporisé du système.
3. Dessinez un diagramme Statechart (d'États-Transitions) pour le système.

Réponses.

1. ASM.

Vocabulaire :

Sortes :
- $NomV$ (nom de véhicule). Variables de sorte $NomV$: m.
- $TypesCarb$ (type de carburant)
- $Pompes$ (pompes). Variables de sorte $Pompes$: p.

Fonctions :
Externes statiques :
- $typeP : Pompes \to TypesCarb$ (retourne le type de carburant d'une pompe)
- $typeM : NomV \to TypesCarb$ (retourne le type de carburant d'un véhicule)
- $add : NomV \to void$ (ajoute un véhicule dans la liste d'attente)
- $delete : NomV \to void$ (supprime un véhicule de la liste d'attente)
- $numVéh : \big(NomV \to \mathbb{N} \cup \{undef\}\big)$ (le numéro de x dans la liste d'attente ; si le véhicule m n'est pas dans la liste alors $numVéh(m) = undef$).
- $nomVéh : \mathbb{N} \to \big(NomV \cup \{undef\}\big)$ (le nom du $k$ème véhicule dans la liste d'attente ; si la liste contient moins que k éléments alors $nomVéh(k) = undef$).

Entrées (externes dynamiques) :
- $arrive :\to \big(NomV \cup \{undef\}\big)$
- $tFree :\to Bool$
- $pDisp : Pompes \to Bool$

Sorties :
- $go : NomV \times Pompes \to Bool$
- $open :\to Bool$

Auxiliaires :
- $dl :\to (\mathcal{T} \cup \{\infty\})$ ('deadline' pour mesurer le temps d'ouverture de la barrière)
- $\mu :\to NomV$ (fonction qui mémorise le véhicule envoyée dans le tunnel pendant Δ temps)
- $\pi :\to Pompes$ (fonction qui mémorise la pompe à laquelle le véhicule a été envoyée aussi pendant Δ temps)

Initialisation : $arrive = undef$, $tFree = pDisp =$ **false**, $open =$ **false**, $go(m, p) =$ **false** pour tout m et p, $dl = \infty$, μ et π n'importe quelles.

Explication du programme. Si un véhicule arrive et il n'est pas dans la liste d'attente alors on la met dans la liste (c'est la 1ère ligne du programme ci-dessous).

Pour comprendre la 2ème ligne supposons pour un instant que le tunnel est libre et $\neg\, go$ pour tout argument. Supposons que la liste n'est pas vide et il y a des pompes avec le carburant demandé par des véhicules de la liste d'attente. Il faut choisir le premier véhicule dans la liste pour la quelle il y a une pompe avec le carburant approprié et l'envoyer à la pompe avec le plus petit numéro parmi les pompes appropriées disponibles. Nous allons exprimer tout ça en utilisant les notations $VéhcPomp$, Θ et J introduites ci-dessous.

$VéhcPomp =_{df} \{(m, p) : numVéh(m) \neq undef \wedge pDisp(p) \wedge typeP(p) = typeM(m))\}$
est l'ensemble des paires (m, p) véhicule-pompe telles que le véhicule m est dans la liste d'attente et il y a une pompe p disponible avec le carburant approprié.

$\Theta =_{df} \min\{k : \exists\,(m, p)\,\big((m, p) \in VéhcPomp \wedge k = numVéh(m)\big)\}$,
$\eta = nomVéh(\Theta)$.

Donc Θ est le numéro du premier véhicule dans la liste à servir et η est son nom.

$J =_{df} \min\{p : (\eta, p) \in VéhcPomp\}$
est la pompe à utiliser pour le véhicule η avec le numéro Θ.

On suppose que $\min \emptyset = undef$.

Maintenant sans aucune hypothèse particulière on peut dire que si le tunnel est libre et aucun véhicule n'est en cours de passage du tunnel (i.e., *go* est faux pour tous les véhicule et pompes) et $VéhcPomp \neq \emptyset$ (i.e., on peut servir quelque véhicule) alors on mémorise η et J et envoie le véhicule η à la pompe J.

Cet envoi est maintenu pendant Δ temps (Ligne 3).

Programme.

if $arrive \neq undef \wedge numVéh(arrive) = undef$ **then** $add(arrive)$
if $tFree \wedge \forall m \forall p \neg go(m,p) \wedge VéhcPomp \neq \emptyset$
then $[go(\eta, J),\ \mu := \eta,\ \pi := J,\ delete(\eta),\ open := \mathbf{true},\ dl := CT + \Delta]$
if $CT \geq dl$ **then** $[go(\mu, \pi) := \mathbf{false},\ open := \mathbf{false},\ dl := \infty]$

2. Requis.

Sûreté.

On n'envoie pas un véhicule dans le tunnel qui n'est pas libre.

On n'envoie pas un véhicule s'il n'y a pas de pompe libre avec le carburant du même type que le carburant de ce véhicule.

On n'envoie pas un véhicule à un moment t si un autre véhicule a été envoyé à un moment t' tel que $|t - t'| < \Delta$. (Définition d'un moment t d'envoi d'un véhicule m à une pompe p : $\forall m \forall p \neg go^{\circ}(t, m, p)$ et $\exists \varepsilon > 0 \forall \tau \in (t, t + \varepsilon)\ go^{\circ}(\tau, m, p)$.)

Vivacité.

Si pour chaque type de carburant il y a une pompe disponible infiniment souvent alors chaque véhicule qui arrive sera servi.

On peut ajouter que les véhicules sont servis selon le principe de file : "first-in-first-out".

3. Diagramme d'états-transitions.