Aslam CADER

TP4 : Heidi

Syntaxe abstraite :

De Heidi :

Ordre = {deponer ; dretg ; sanester ; davent ; davos ; plaun ; returnar ; sa fermar}

---
x € Ordre

x € Ordre
---
x € listOrdre

h € Ordre, t € listOrdre
---
h :: t € listOrdre

De Tita :

Whistle = {Court ; Whee ; Who ; Wheeo ; Hee ; Long}

---
pause € Pause

---
x € Whistle

---
x € Pause

// condition d'arrêt
h € Pause
---
t € séancedetravail

h € Whistle
---
t € séancedetravail

h € Whistle, t € séancedetravail
---
h :: t € séancedetravail

h € Pause, t € séancedetravail
---
h :: t € séancedetravail

<u>Sémantique</u> :
Heidi vers Tita (h to t)

Deponer € Ordre, Court € Whistle

Deponer -h to t-> Court :: Court

Dregt € Ordre, {Whee, Who} € Whistle

Dregt -h to t-> Whee :: Who

Sanester € Ordre, { Wheet , Wheet } € Whistle

Sanester -h to t-> Wheet :: Wheet

Davent € Ordre, { Wheet , Wheeo } € Whistle

Davent -h to t-> Wheet :: Wheeo :: Wheet :: Wheet ;

Davos € Ordre, { Who, Hee } € Whistle

Davos -h to t-> Who :: Hee :: Who

Plaun € Ordre, Hee € Whistle

Plaun -h to t-> Hee :: Hee :: Hee :: Hee

Retunar € Ordre, {Whee , Wheet }€ Whistle

Retunar -h to t-> Whee :: Whee :: Wheet

Sa fermar € Ordre, Long € Whistle

Sa fermar -h to t-> Long

Tita to Heidi :

order € Order, Whistle € Whistles, order -h to t-> order

Whitsle -t to h-> order

Optimisation

Redéfinissons les whistles
Whistle = { Wheeo ; Hee ; Wheet }

Deponer € Ordre, {Wheeo, Hee, Wheet} € Whistle

Deponer -h to t-> Wheeo -- Hee -- Wheet

Dregt € Ordre, {Hee, Wheet} € Whistle

Dregt -h to t-> Hee -- Wheet

Sanester € Ordre, { Wheet , Wheeo } € Whistle

Sanester -h to t-> Wheet -- Wheeo

Davent € Ordre, { Wheet , Hee } € Whistle

Davent -h to t-> Wheet -- Hee -- Wheet

Davos € Ordre, { Wheet, Wheeo } € Whistle

Davos -h to t-> Wheet -- Wheeo -- Wheet

Plaun € Ordre, {Wheet, Wheeo} € Whistle

Plaun -h to t-> Wheet -- Wheeo -- Wheeo

Retunar € Ordre, {Wheeo , Wheet }€ Whistle

Retunar -h to t-> Wheeo – Wheet

Sa fermar € Ordre, {Wheeo} € Whistle

Sa fermar -h to t-> Wheeo -- Wheeo

---

order € Order, Whistle € Whistles, order -h to t-> order

Whitsle -t to h-> order

Preuve
Chaque ordre est séparé par une pause.
Il n'y a pas d'ambigüité entre les ordres (pas d'ordre qui ont les mêmes coups de sifflets).
Par conséquent, chaque élément traduit et retraduit directement sans erreur.

Accélération

```
[/* ACCELERATION */

% traduction

htoT([plaun, dretg, plaun, deponer, safermar], X).

% traduction2

ttoH([wheet, wheeo, wheeo, hee, wheet, wheet, wheeo, wheeo, wheeo, hee, wheet,
wheeo, wheeo], X).

% résultat obtenu par mon programme:

% X = [sanester, deponer, sanester, safermar, dretg, safermar]

% X = [sanester, deponer, plaun, deponer, safermar]
```

% X = [plaun, dretg, sanester, safermar, dretg, safermar]

% X = [plaun, dretg, plaun, deponer, safermar] % interprétation de départ

On remarque qu'il est possible d'interpréter différemment les ordres car les pauses sont absentes.

On a eu 4 interprétations pour l'exemple du sujet.

Problèmes
Oui, il est possible de faire :

Voici le code qui nous permet d'avoir toutes les possibilités qui renvoie un élément différent quand il y a 3 ordres à la suite :
htoT([A, B, C], Y), ttoH(Y, [D, E, F]), not(A = D), not(B = E), not(C = F).

Plus généralement, on peut faire :
htoT(X, Y), ttoH(Y, A), not(X = A).

Mais cela prend trop de temps à calculer car le programme passe par les solutions ou X = A (ie. Pour chaque itération qu'il calcule, il arrive jusqu'au résultat avant de l'ignorer : selon moi).