EPU GB 4ème année **TD Techniques de simulation No. 2** 25 mars 2024

Analyse par réseau de Petri d'une voie métabolique

Objectifs. L'objectif de ce TD est de mettre en œuvre les différentes techniques vues en cours sur les réseaux de Petri. L'exemple provient de l'article suivant :
M. Heiner and I. Koch, Petri Net Based Model Validation in systems biology, ICATPN 2004, LNCS 3099, pp. 216–237.

L'accumulation de féculents ou glucides complexes dans la pomme de terre est devenu un point crucial en biotechnologie. La conversion du sucrose *via* des phosphates hexose constitue le principal flux dans le métabolisme du carbone chez la pomme de terre. Presque tous les gènes impliqués directement dans ces réactions ont été séquencés, cependant, un grand nombre de questions fondamentales restent ouvertes. Les 16 équations suivantes caractérisent la voie métabolique :

R_1	$SuSy(sucrosesynthetase)$	$Suc + UDP$	$\leftrightarrow$	$UDPglc + Frc$
R_2	$UGPase(UDPglucosepyrophosphorilase)$	$UDPglc + PP$	$\leftrightarrow$	$G1P + UTP$
R_3	$PGM(phosphoglucomutase)$	$G6P$	$\leftrightarrow$	$G1P$
R_4	$FK(fructokinase)$	$Frc + ATP$	$\rightarrow$	$F6P + ADP$
R_5	$PGI(phosphoglucoseisomerase)$	$G6P$	$\leftrightarrow$	$F6P$
R_6	$HK(hexokinase)$	$Glc + ATP$	$\rightarrow$	$G6P + ADP$
R_7	$Inv(invertase)$	Suc	$\rightarrow$	$Glc + Frc$
R_8	$Glyc(b)(Glycolysis)$	$F6P + 29ADP + 28P_i$	$\rightarrow$	$29ATP$
R_9	$SPS(sucrosephosphatesynthase)$	$F6P + UDPglc$	$\leftrightarrow$	$S6P + UDP$
R_{10}	$SPP(sucrosephosphatephosphatase)$	$S6P$	$\rightarrow$	$Suc + P_i$
R_{11}	$NDPkin(NDPkinase)$	$UDP + ATP$	$\leftrightarrow$	$UTP + ADP$
R_{12}	$SucTrans(sucroseTransporter)$	$eSuc$	$\rightarrow$	Suc
R_{13}	$ATPcons(b)(ATPconsumption)$	ATP	$\rightarrow$	$ADP + P_i$
R_{14}	$StaSy(b)(starchsynthesis)$	$G6P + ATP$	$\rightarrow$	$starch + ADP + PP$
R_{15}	$AdK(adelylatekinase)$	$ATP + AMP$	$\leftrightarrow$	$2ADP$
R_{16}	$PPase(pyrophosphotase)$	PP	$\rightarrow$	$2P_i$

Exercice 1 : (Construction du réseau de Petri – avec l'outil Tina)

Construire le réseau de Petri à partir de la description du système dans le simulateur/analyseur de réseau de Petri. Le logiciel TINA (Time Integrated Net Analyser) est disponible à l'adresse `http://www.laas.fr/tina`.

Notons que pour l'étude de ce réseau, deux places (*eSuc* et *starch*) rendent l'étude des comportements stationnaires difficiles. Nous allons donc rajouter deux transitions qui n'ont pas de signification biologique :

— une transition pour consommer une molécule de féculent,
— une transition pour créer un *eSuc*.

Exercice 2 : (**Les P-invariants**)

Combien y a-t-il des P-invariants ? On utilisera TINA pour le calcul des P-invariants. A quoi correspond chacun de ces P-invariants ?

Exercice 3 : (**Les T-invariants**)

Etude des T-invariants.

1. Calculez tous les T-invariants à l'aide de TINA. Combien en trouvez-vous ?
2. Parmi ces T-invariants, lesquels étaient évidents ? Combien de transitions comportent-ils ?
3. Y a-t-il des transitions qui n'apparaîssent pas dans les invariants ? Que concluez-vous ?
4. Y a-t-il des transitions qui ne sont impliquées dans aucun T-invariant non-trivial ?
5. Simplifiez le réseau de Petri en supprimant les transistions obtenues à la question précédente. Recalculez les P-invariants et T-invariants. Concluez.
6. Est-ce que la synthèse de féculent est possible sans glycolyse en régime stationnaire ? Justifiez votre réponse sur l'étude des T-invariants. Est-ce que cette réponse est biologiquement explicable ?
7. La glycolyse qui produit l'ATP nécessaire, a besoin de sucrose. Ce sucrose peut-il être cleavé par Inv ? par Susy ? par les 2 à la fois (en régime stationnaire) ?

Exercice 4 :

Est-ce que ce réseau de Petri est borné ? Si ce n'est pas le cas, comment faire pour le borner ? Implémentez dans tina ce réseau de Petri borné, et vérifiez qu'il est bien borné en utilisant l'analyse d'atteignabilité.

Exercice 5 : (**Utilisation des ADT sets**)

Dans cet exercice, on partira du réseau de Petri créé à l'exercice 1 avant toute modification.

- Calculez les ADT sets. Pour cela on pourra écrire un petit programme python qui construira automatiquement les ADT sets.
  - écrire une fonction `extraireListesTransitions` qui extrait à partir du fichier donné par `tina`, la liste des transitions.
  - écrire une fonction `extraireLesTInvariants` qui extrait à partir du fichier donné par `tina`, la liste des T-invariants. On utilisera la fonction `re.sub()` du module `import re` (regular expression) pour supprimer les coefficients des transitions.
  - écrire une fonction `memeclasse` qui évalue si 2 éléments sont dans la même classe (la liste des T-invariants est passée en paramètre).
  - écrire une fonction `calculCE` qui construit la classe d'équivalence d'un élement.
  - enfin, la fonction `calcul_ADTsets`, appelle la fonction `calculCE` pour chaque transition, et construit ainsi la liste de toutes les classes d'équivalence.
- A quoi correspondent-ils ? Peut-on les expliquer tous ?
- Construire le réseau simplifié.
- Recalculer les T-invariants.