1 Réseaux métaboliques : Flux Balance Analysis (FBA)

La plupart du temps, les connaissances moléculaires « locales » d'un réseau métabolique telles que les réactions élémentaires et leur stoechiométrie sont assez bien connues. Ce qui est nettement moins bien connu, ce sont les vitesses des réactions *in vivo*, entre autres parce que les mesures *in vitro* s'extrapolent assez mal *in vivo*, et surtout parce que cette vitesse est contrainte par les flux au sein du réseaux lui-même. On ne sait généralement que donner des valeurs minimale et maximale de vitesses crédibles pour chaque réaction.

La méthode FBA a pour objectif de calculer ces vitesses, au sein du réseau métabolique dans sa globalité, en supposant que le réseau « optimise » la production de certains métabolites. Une hypothèse commune est par exemple que la cellule optimise la production de composés utiles à sa biomasse.

À l'inverse, on peut aussi *vouloir* optimiser la production de certains métabolites d'intérêt (production de biofuel, remplacement des matières plastiques, production de molécules pharmaceutiques, *etc*). Dans ce cas, on peut par exemple considérer que la vitesse d'une réaction métabolique est liée à la quantité d'enzymes produite par les gènes. Dès lors, en comparant les vitesses du réseau métabolique « naturel » (e.g. hypothèse d'optimisation de biomasse) avec celles que FBA calcule pour optimiser la production d'intérêt, on peut modifier le génome pour renforcer ou réduire la présence de certains enzymes et donc augmenter la vitesse de certaines réactions d'intérêt. Une approche complémentaire est bien sûr de contrôler les flux des métabolites d'entrée (donc des réactions entrantes).

Plus avancé encore, on peut même contrôler (à la lumière des prédictions faites par FBA) les mutations d'une population de bactéries, par un jeu d'avantages sélectifs, pour obtenir une population qui optimise la production du produit d'intérêt. On peut aussi modifier le génome de certaines plantes ou algues dans la même optique. On touche là à la *biologie synthétique* également appelée *biologie de synthèse* par analogie avec la chimie de synthèse qui a, entre autres, ouvert la voie aux traitements des produits pétroliers ou à la production industrielle de molécules pharmaceutiques.

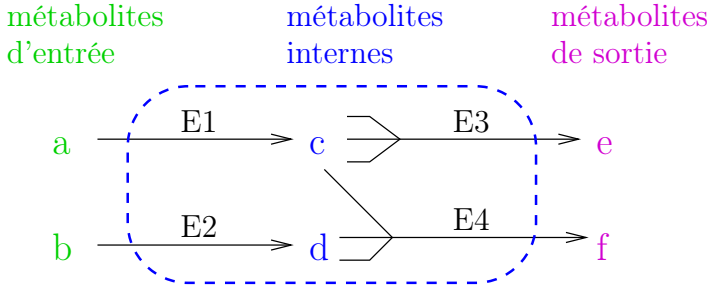
La méthode FBA se décompose en 6 phases :

1. Inventaire des réactions avec leur stœchiométrie.
2. Création d'une matrice qui encode le réseau constitué par toutes les réactions précédentes.
3. Extraction, à partir de la matrice, des contraintes qui traduisent qu'aucun substrat intermédiaire ne s'accumule ou ne se consomme dans la cellule (hypothèse « d'état stationnaire »).
4. Détermination des intervalles de vitesses crédibles pour chaque réaction.
5. Traduction sous forme de « poids » sur les réactions sortantes des hypothèses ou volontés d'optimisation précitées.
6. Maximiser, sous les contraintes de stationnarité précédentes, les vitesses des réactions sortantes pondérées par leur poids (outils de résolution de contraintes).

Nous allons détailler la méthode en suivant un exemple.

1.1 Inventaire des réactions

Supposons par exemple que l'enzyme E1 catalyse la transformation d'un substrat a en un produit c, que E2 transforme de même b en d, que E3 transforme trois c en un e et enfin que E4 transforme un c et deux d en un f.



On peut résumer la situation sous forme de règles :

E1 : a $\rightarrow$ c
E2 : b $\rightarrow$ d
E3 : 3c $\rightarrow$ e
E4 : c + 2d $\rightarrow$ f

où a et b sont les substrats d'entrée du réseau métabolique et où e et f sont ses produits de sortie.

1.2 Matrice du réseau

On peut construire une *matrice de stœchiométrie* S en mettant les réactions enzymatiques en colonnes et les métabolites en lignes. Les nombres entiers placés dans la matrice sont les coefficients stœchiométriques, en positif lorsque le métabolite est produit (à droite de la règle) et en négatif lorsqu'il est consommé (à gauche de la règle).

Pour notre exemple, cela donne la matrice S suivante :

	E1	E2	E3	E4
a	-1	0	0	0
b	0	-1	0	0
c	1	0	-3	-1
d	0	1	0	-2
e	0	0	1	0
f	0	0	0	1

1.3 Contraintes d'état stationnaire

L'hypothèse « d'état stationnaire » revient à supposer que les flux rendent le nombre de chaque métabolite interne constant. Autrement dit, une fois le flux métabolique établi, les métabolites internes ne s'accumulent pas et ne diminuent pas non plus en nombre. Cela signifie que la résultante des vitesses de production/consommation de chacun des métabolites internes est nulle [1].

Supposons que la catalyse de E1 se fasse à la vitesse v_1, celle de E2 à la vitesse v_2, *etc.* Si on note $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$ le vecteur des vitesses des réactions catalytiques, alors le produit $S.\vec{v}$ est le vecteur des vitesses résultantes des métabolites :

$$\begin{array}{cc} \text{a} \\ \text{b} \\ \text{c} \\ \text{d} \\ \text{e} \\ \text{f} \end{array} \begin{pmatrix} -1 \times v_1 + 0 \times v_2 + 0 \times v_3 + 0 \times v_4 \\ 0 \times v_1 + -1 \times v_2 + 0 \times v_3 + 0 \times v_4 \\ 1 \times v_1 + 0 \times v_2 + -3 \times v_3 + -1 \times v_4 \\ 0 \times v_1 + 1 \times v_2 + 0 \times v_3 + -2 \times v_4 \\ 0 \times v_1 + 0 \times v_2 + 1 \times v_3 + 0 \times v_4 \\ 0 \times v_1 + 0 \times v_2 + 0 \times v_3 + 1 \times v_4 \end{pmatrix}$$

Les métabolites internes sont c et d donc la contrainte d'état stationnaire se traduit par : $1 \times v_1 + 0 \times v_2 - 3 \times v_3 - 1 \times v_4 = 0$ et $0 \times v_1 + 1 \times v_2 + 0 \times v_3 - 2 \times v_4 = 0$, c'est-à-dire $v_1 - 3v_3 - v_4 = 0$ et $v_2 - 2v_4 = 0$, autrement dit $3v_3 = v_1 - v_4$ et $v_4 = \frac{v_2}{2}$. Et tout cela donne finalement : $v_4 = \frac{v_2}{2}$ et $v_3 = \frac{2v_1 - v_2}{6}$, de sorte que les seules inconnues à déterminer sont v_1 et v_2.

1.4 Vitesses crédibles

On détermine ensuite, à partir des connaissances biologiques sur ces réactions enzymatiques dans le cadre du réseau métabolique étudié, les vitesses maximales et minimales possibles de chaque réaction enzymatique. Si la réaction est réversible alors la vitesse minimale est négative. La vitesse maximale est toujours positive. Si la réaction n'est pas considérée comme réversible dans le réseau métabolique étudié alors la vitesse minimale est également positive.

On peut, par exemple, considérer les inégalités suivantes :

$$0 \leqslant v_1 \leqslant 30 \ , \quad 0 \leqslant v_2 \leqslant 12 \ , \quad 0 \leqslant v_3 \leqslant 4 \quad \text{et} \quad 0 \leqslant v_4 \leqslant 3 \ .$$

1.5 Poids sur les réactions sortantes

La plupart du temps, on cherche à optimiser la production d'un métabolite sortant ou d'un cocktail de métabolites sortants. Pour des raisons purement calculatoires, on considère plutôt un cocktail de *réactions sortantes*. On utilise des « poids » pour indiquer l'importance relative qu'on accorde aux différentes réactions sortantes.

Par exemple si c'est le produit f qui nous intéresse et que e n'a pas d'intérêt, on donnera un poids 0 aux réactions catalysées par E1, E2 et E3 et un poids 1 à celle catalysée par E4. Si au contraire e nous intéresse deux fois plus que f, alors on donnera un poids 0 aux réactions catalysées par E1 et E2, un poids 2 à celle catalysée par E3 et un poids 1 à celle catalysée par E4.

1. Ne pas confondre « état stationnaire » et « état d'équilibre », ce dernier représente un état thermodynamique où l'énergie de Gibbs est nulle, donc où la vitesse des réactions correspondantes est nulle.

On note $\vec{p} = (p_1, \cdots, p_n)$ le « vecteur ligne » des poids donnés à chaque réaction. Nos deux exemples précédents sont donc respectivement $\vec{p} = (0, 0, 0, 1)$ et $\vec{p} = (0, 0, 2, 1)$.

1.6 Optimisation

Pour optimiser la production souhaitée, il suffit alors d'optimiser la valeur du produit $\vec{p}.\vec{v} = p_1 \times v_1 + p_2 \times v_2 + \cdots$ parmi toutes les vitesses autorisées par les contraintes de stationnarité (cf. 1.3) et les vitesses crédibles (cf. 1.4).

Pour ce faire, il existe des logiciels de *programmation linéaire* qui permettent de résoudre des problèmes d'optimisation linéaire sur le polygone délimité par les contraintes précitées.

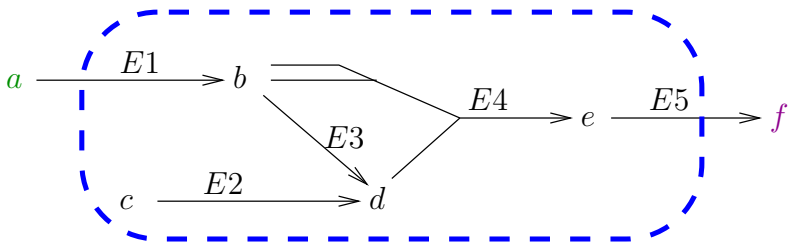
Notre exemple de petite taille peut se résoudre à la main :
- Si $\vec{p} = (0, 0, 0, 1)$ alors on veut simplement maximiser $v_4 = \frac{v_2}{2}$ c'est-à-dire maximiser v_2. Cependant v_2 ne peut pas atteindre son maximum 12 car il faut aussi que $v_4 = \frac{v_2}{2} \leqslant 3$. Si $v_2 = 6$ alors sachant $v_3 = \frac{2v_1 - v2}{6} \leqslant 4$, il faudra $2v_1 - 6 \leqslant 24$ soit $v1 \leqslant 15$, donc $v_2 = 6$ est l'optimum.
- Si $\vec{p} = (0, 0, 2, 1)$ alors on veut maximiser $2v_3 + v_4 = \frac{2v_1 - v_2}{3} + \frac{v_2}{2}$, c'est-à-dire maximiser $4v_1 + v_2$. C'est encore faisable à la main parce que les vitesses crédibles de la section 1.4 reviennent à imposer $0 \leqslant v_1 \leqslant 30$, $0 \leqslant v_2 \leqslant 6$ et $0 \leqslant 2v_1 - v2 \leqslant 24$... Cependant on commence à bien voir l'intérêt d'un logiciel d'optimisation linéaire !
  Pour cet exemple, v_1 a 4 fois plus de poids que v_2 pour optimiser selon notre vecteur de poids, donc on a intérêt à maximiser v_1. Malheureusement sa valeur maximale, 30, conduit à $2v_1 - v2 > 24$ car même si l'on prend la valeur maximale de v_2, $2 \times 30 - 6 = 54$. En fait, puisque $2v_1 \leqslant 24 + v_2$, il en résulte que $2v_1$ peut être au maximum égal à $24 + 6 = 30$. En prenant $v_2 = 6$ et $v_1 = 15$ toutes les contraintes sont respectées, on obtient ainsi l'optimum (donc $v_3 = 4$ et $v_4 = 3$).

2 TD sur FBA

La bactérie *P.exemplum* possède un réseau métabolique représenté ci-dessous qui produit le métabolite f.
f est une molécule qui possède un fort intérêt commercial alors que le nutriment a est de faible coût. On va donc appliquer des techniques de biologie synthétique sur la bactérie *P.exemplum* pour lui faire excréter f en aussi grande quantité que possible.

P.exemplum synthétique :

$E1 : a \to b \quad (v_1)$
$E2 : c \to d \quad (v_2)$
$E3 : b \to d \quad (v_3)$
$E4 : 2b + d \to e \quad (v_4)$
$E5 : e \to f \quad (v_5)$



L'objectif est d'évaluer les vitesses v_i optimales des différentes réactions afin de savoir si, d'une part, on peut faire des KO de gènes pour « alléger » le coût de prolifération de la bactérie, et d'autre part, s'il est rentable de dupliquer certains gènes d'enzymes pour augmenter la vitesse de réaction correspondante.

Exercice 1 : Construisez la matrice de stœchiométrie correspondante puis écrivez les contraintes d'état stationnaire sur les vitesses v_i qui rendent constantes les quantités de métabolites *internes*.

Exercice 2 : Simplifiez les contraintes d'état stationnaire et indiquez s'il est possible de faire des KO d'enzymes sans altérer la production de f.

Pour optimiser la production de f il faut maximiser v_5. Les contraintes sur les vitesses, connues par la littérature scientifique sur *P.exemplum* sauvage, sont les suivantes : $0 \leqslant v_1 \leqslant 33$, $0 \leqslant v_2 \leqslant 50$, $0 \leqslant v_3 \leqslant 20$, $0 \leqslant v_4 \leqslant 20$ et $0 \leqslant v_5 \leqslant 20$.

Exercice 3 : Donnez le jeu de vitesses qui optimise la production de f. Où se trouve le goulot d'étranglement ? Et quel gène serait le plus rentable à dupliquer ?

3 Logique propositionnelle

3.1 Les logiques

Lorsqu'on veut déléguer à l'ordinateur des « raisonnements » pour établir des propriétés sur des modèles biologiques, il faut faire appel à 3 choses :

1. une manière conventionnelle d'écrire les propriétés qui nous intéressent, de telle sorte qu'un algorithme simple puisse analyser le texte d'une propriété sans aucune ambiguïté ;
2. une définition de l'ensemble des modèles considérés et une convention rigoureuse sur ce que signifie chaque propriété qu'on a le droit d'écrire, lorsqu'elle est étudiée sur un modèle ;
3. pour chaque propriété et pour chaque modèle, un jeu de manipulations mécaniques pour (tenter de) déterminer si la propriété est vraie ou fausse pour ce modèle.

La définition de ces 3 volets s'appelle *une logique*. Le premier volet est appelé *la syntaxe*, le deuxième *la sémantique* et le troisième *les preuves*.

1. La syntaxe d'une logique est généralement obtenue en se donnant des *connecteurs logiques* comme par exemple la conjonction (notée « $\wedge$ »), la disjonction (« $\vee$ »), l'implication (« $\Longrightarrow$ ») ou la négation (« $\neg$ »), parfois des *quantificateurs* (« $\forall$ » et « $\exists$ ») ou des *modalités* (voir plus loin).
   Ensuite, selon la question abordée, on a besoin de syntaxe spécifique à cette question, comme par exemple des noms de protéines ou de contextes spécifiques (e.g. « `phosphorylé` ») : ces éléments de syntaxe supplémentaires et dépendants de la question traitée sont appelés une *signature*.
2. La sémantique d'une logique, pour une signature donnée, est simplement l'ensemble de *tous* les modèles potentiels de la question d'intérêt (même les moins crédibles), et une définition (interprétation) claire de ce que signifie chacun des éléments de la signature au sein de chaque modèle.
3. Enfin le volet des preuves est obtenu en décrivant toutes les manipulations mécaniques qu'on s'autorise pour vérifier si une propriété est vraie pour un ensemble de modèles donné. Par exemple, une transformation mécanique classique pour prouver « $\varphi \wedge \psi$ » et de prouver successivement « φ » puis « ψ ». Une autre transformation mécanique classique, pour prouver « $\varphi \Longrightarrow \psi$ » est de supposer que φ est vraie et de chercher à prouver ψ dans ce contexte.
   - Un point absolument majeur est qu'il faut assurer que toute propriété prouvée est effectivement vraie d'après la sémantique : c'est la *correction* des techniques de preuve.
   - Une propriété très appréciable est la *complétude*, c'est-à-dire que si une propriété est vraie selon la sémantique alors il existe une manière de le prouver avec les manipulations qu'on s'est données. Malheureusement certaines logiques utiles peuvent parfois ne pas être complètes.
   - Enfin une propriété plus rare mais vraiment pratique est la *décidabilité*, c'est-à-dire que, si une propriété est vraie selon la sémantique, alors non seulement sa preuve existe (complétude) mais de plus il existe un algorithme « pilote des manipulations mécaniques » capable de la trouver en un temps fini.

En logique, les propriétés exprimées selon la syntaxe de cette logique sont appelées des *formules*.

3.2 La logique propositionnelle

La logique propositionnelle est probablement la plus simple qui possède quelque utilité.

1. La syntaxe générale ne contient que les connecteurs logiques habituels : $\wedge$, $\vee$, $\neg$ et $\Longrightarrow$.
   Une signature ne peut contenir que des *propositions*, c'est-à-dire des propriétés fixes, sans aucune variable. Par exemple une application à la météo pourrait contenir des propositions comme p_1=« *Il pleut* » ou p_2=« *Le ciel est bleu* » ou encore p_3=« *Il gèle* ». En revanche elle ne permet pas des choses plus sophistiquées comme comparer des températures entre elles, mesurer exactement un niveau de précipitation ou autres. Seules des propositions « d'un bloc » sont admises, et de plus elles doivent être en nombre fini.
   Dès lors on peut considérer des formules comme φ = « $p_2 \Longrightarrow \neg p_1$ » par exemple.
2. Un modèle de la sémantique est alors défini comme une observation quelconque de la vérité ou fausseté de chaque proposition. Par exemple A=

p_1	p_2	p_3
vrai	faux	faux

est un modèle. De même B=

p_1	p_2	p_3
vrai	vrai	faux

est un modèle.
   On voit que la formule φ est vraie dans A mais fausse dans B. On dit alors que A *satisfait* φ et on le note $A \models \varphi$. De même B *ne satisfait pas* φ et on le note $B \not\models \varphi$.
3. Pour faire une preuve dans un modèle, on peut utiliser deux approches distinctes : soit travailler dans le modèle en question avec les tables de vérité des connecteurs logiques et on fait alors du *model checking*, soit faire des preuves syntaxiques par transformations de formules (« arbres de preuve »). On va se concentrer ici sur les tables de vérité des connecteurs logiques :

$\neg$	**vrai**	**faux**
	faux	vrai

$\wedge$	**vrai**	**faux**
vrai	vrai	faux
faux	faux	faux

$\vee$	**vrai**	**faux**
vrai	vrai	vrai
faux	vrai	faux

$\Longrightarrow$	**vrai**	**faux**
vrai	vrai	faux
faux	vrai	vrai

   On remarque que *faux* $\Longrightarrow$ *n'importe quoi* est toujours *vrai*.
   Grâce à ces tables, il n'est pas difficile d'écrire un petit programme qui établit la véracité de n'importe quelle formule propositionnelle.
   Pour les preuves par transformations syntaxiques de formules, on pourra se référer à Wikipedia par exemple. Il faut juste avoir en tête que la notation $\frac{\varphi_1 \quad \varphi_2 \quad \cdots \quad \varphi_n}{\varphi}$ signifie simplement que pour prouver φ, il suffit de prouver

successivement φ_1, φ_2, ... jusqu'à φ_n. Une telle notation sous forme de fraction s'appelle une *règle d'inférence* et peut être manipulée par ordinateur sans grande difficulté.

Exercice 4 : Prouvez avec les tables de vérité que dans l'exemple de la météo, $A \models \varphi$ et $B \not\models \varphi$.

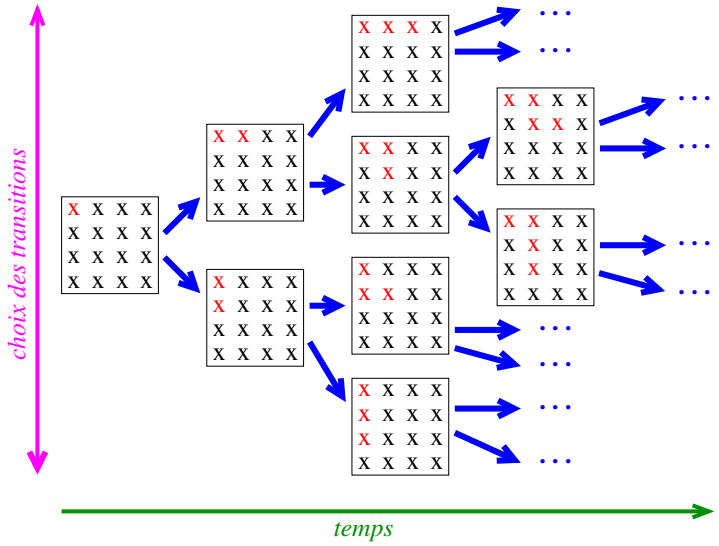
Exercice 5 : Prouvez que la formule suivante est toujours vraie, quelle que soit la véracité de la formule φ :

$$\neg\neg\varphi \iff \varphi$$

4 Logique temporelle CTL

Une limitation de la logique propositionnelle est qu'elle n'est pas prévue pour exprimer des propriétés sur le comportement d'un système au cours du temps. Or nous avons besoin de le faire pour exprimer des comportements biologiques comme des oscillations entretenues, une homéostasie, *etc.* qui sont des propriétés mettant en jeu le temps. Pour ces propriétés sur la *dynamique des systèmes*, il n'est pas toujours nécessaire d'aller jusqu'à prendre en compte un temps *chronométrique* (c'est-à-dire avec des mesures précises du temps passé entre deux événements). Souvent, un temps *chronologique* (ne traitant que l'ordre des événements) suffit : par exemple pour indiquer qu'un taux de concentration oscille entre « *fort* » et « *faible* », il suffit d'écrire deux formules qui signifient que (1) si l'état est *fort* alors plus tard il sera *faible* et (2) si l'état est *faible* alors plus tard il sera *fort*. Il n'est pas toujours nécessaire de connaître la durée de chaque oscillation.
La logique CTL (pour *Computation Tree Logic*) couvre précisément ce besoin.

On a vu en début de ce cours que les *automates* jouent un rôle majeur dans la modélisation des systèmes dynamiques biologiques. En général, un automate crédible pour la biologie n'est pas déterministe, de sorte que depuis chaque état de l'automate, plusieurs transitions vers un état suivant sont souvent possibles. Cela a pour conséquence que, partant d'un état initial donné, l'ensemble des trajectoires possibles du système a une forme d'arbre :



Dans ce schéma, le temps s'écoule de gauche à droite avec la succession des transitions et les choix non déterministes des transitions, qui forment pas à pas un chemin au cours du temps, sont effectués selon l'axe vertical. Par ailleurs chaque état représenté dans l'arbre possède des propriétés qui lui sont propres : par exemple la proposition « *le* 1^{er} X *de la* 2^{me} *ligne est rouge* » n'est vrai que dans les trois états du bas de la figure.
On peut considérer que chaque état est un modèle simple de la logique propositionnelle pour une signature à $4 \times 4 = 16$ propositions : p_{ij} = « *le* i^{me} X *de la* j^{me} *ligne est rouge* » (et la négation indique que le X est noir). Naturellement, l'ensemble des propositions intéressantes sur les états dépend du système modélisé.

Dès lors, il est naturel de se poser des questions sur la dynamique du système comme par exemple savoir si *le* 4^{me} X *de la* 4^{me} *ligne deviendra nécessairement rouge au bout d'un certain temps* ou bien *si l'on choisit bien son chemin il est toujours possible de colorier en rouge le* 3^{me} X *de la* 2^{me} *ligne*... Cela suppose de pouvoir parler du futur à partir de n'importe quel état initial et c'est là que les *modalités temporelles* entrent en jeux.

En CTL, une modalité est toujours constituée de deux lettres, la première concerne les choix verticaux et la seconde le temps qui s'écoule :

premier caractère	second caractère
A = for **A**ll path choices	X = ne**X**t state
	F = for some **F**uture state
E = there **E**xist a choice	G = for all future states (**G**lobally)
	U = **U**ntil

Ainsi par exemple la propriété « Si *le* 3^{me} X *de la* 2^{me} *ligne est rouge* alors, quoi qu'il arrive, dans l'état suivant *le* 3^{me} X *de la* 2^{me} *ligne est rouge* » va s'écrire : « $p_{32} \Longrightarrow \mathrm{AX}(p_{32})$ ».
La propriété *Le* 4^{me} X *de la* 4^{me} *ligne deviendra nécessairement rouge au bout d'un certain temps* s'écrit : « $\mathrm{AF}(p_{44})$ »
La propriété *Si l'on choisit bien son chemin il est toujours possible de colorier en rouge le* 3^{me} X *de la* 2^{me} *ligne* s'écrit : « $\mathrm{EF}(p_{32})$ ».
La propriété *Si le* 1^{er} X *de la* 2^{me} *ligne est rouge, alors, quoi qu'il arrive, il le restera toujours* s'écrit :
« $p_{12} \Longrightarrow \mathrm{AG}(p_{12})$ ».
La propriété *Dans tous les chemins le* 1^{er} X *de la* 2^{me} *ligne est noir jusqu'à ce qu'il devienne rouge* s'écrit : « $\mathrm{A}[\neg p_{12} \mathrm{U} p_{12}]$ ».

CTL présente quelques particularités :
- (A ou E)$[\psi \mathrm{U} \varphi]$ implique (A ou E)Fφ c'est-à-dire que φ doit nécessairement arriver un jour ou l'autre ;
- lorsque AFφ est vrai, cela n'implique pas que φ arrive au même moment pour tous les chemins : il peut y avoir des chemins courts et d'autres longs ;
- (A ou E)Fφ n'implique pas que φ reste vrai ultérieurement sur le chemin ;
- enfin *le futur inclût le présent* donc si φ est vrai dans un état initial alors (A ou E)Fφ y est également toujours vrai ! si l'on veut parler du future strict, il faut écrire AXAFφ ou EXEFφ.

On peut résumer tout cela par la figure ci-dessous dans laquelle, par convention, une transition est bleue lorsqu'elle aboutit à un état où φ est vraie, rouge si elle aboutit à un état où ψ est vraie, verte si un chemin qui l'emprunte participe à la véracité de la formule écrite, et noire sinon.