VACSIM

Validation de la commande des systèmes critiques par couplage simulation et méthodes d'analyse formelles

Tâche 5

Validation formelle de propriétés quantitatives : Approche par contraintes

Livrable L5.1

Apport de la bio-informatique à la vérification des systèmes complexes

Version 1

Appel :	**PROGRAMME INGENIERIE NUMERIQUE & SECURITE 2011**
Numéro d'agrément :	A**NR-11-INSE-004**
Thématique:	**Systèmes embarqués et ingénierie du logiciel**
Objectif:	**Validation par simulation de partie opérative**
Date de démarrage du projet:	**01.10.2011**
Durée :	**42 mois**

Synthèse

Le projet **VACSIM** (Validation de la commande des systèmes critiques par couplage simulation et méthodes d'analyse formelle), référencé ANR-11-INSE-004, étudie les avantages respectifs des techniques de simulation, en incluant des modèles des processus commandés, et des méthodes d'analyse formelles, pour la validation de la commande des systèmes critiques. Ce projet est structuré en 6 tâches.

La tâche 5, « Validation formelle de propriétés quantitatives : Approche par contraintes », a pour objectif de contribuer à l'avancée des techniques de validation par résolution de systèmes de contraintes. Elle est découpée en deux sous-tâches qui abordent des aspects complémentaires de la la validation des systèmes critiques : la vérification de propriétés quantitatives pour la sous-tâche T5.1 et la localisation d'erreur pour la sous-tâche T5.2.

Ce livrable L5.1 du projet VACSIM est issu des travaux de la sous-tâche T5.1 : « Génération et résolution des systèmes de contraintes ». Ce livrable explore les techniques de « model checking », au sens large, mises en œuvre en bio-informatique, et notamment les techniques de simplification de modèles complexes. L'objectif est d'évaluer les possibilités de transférer ces techniques dans le cadre de la validation des systèmes embarqués du projet VACSIM. Dans ce livrable, nous identifions en particulier une technique dont l'adaptation à la vérification des programmes semble prometteuse.

Ce livrable L5.2 a été rédigé par l'I3S.

Table des matières

1 Validation « bio-inspirée » pour VACSIM

La modélisation des réseaux génétiques a exploré différentes voies pour traiter le problème de la complexité au sens large de ces réseaux. L'objectif de ce rapport est de présenter une synthèse de ces voies et de présenter quelques pistes qui mériteraient d'être approfondies dans le cadre du projet VACSIM.

Un des problèmes majeurs en bio-informatique réside dans l'ampleur du nombre de paramètres et la difficulté d'identification de ces paramètres. Différentes voies ont été explorées en bio-informatique pour essayer de résoudre ce problème. Des plus statiques aux plus dynamiques (dynamique mathématique puis dynamique biologique) on peut notamment mentionner :

- L'étude de l'impact de la structure du réseau sans considérer les paramètres : il s'agit d'une part d'analyser l'importance des cycles dans le graphe d'influence, et d'autre part d'évaluer l'importance des modes de mise à jour des variables ;
- Les techniques de réduction de modèles guidée par une propriété étudiée : concrètement, on se focalise sur un sous-graphe qui « pilote » les propriétés d'intérêt ou on cherche à réduire progressivement le nombre de gènes/variables dans le graphe d'interactions ;
- Les logiques temporelles ;
- Les logiques de Hoare ;
- La suggestion d'expériences à partir de modèles.

Dans la suite de ce document, nous allons d'abord étudier les points forts et les limitations de ces approches dans la perspective de les transférer et les adapter pour la validation des systèmes embarqués dans le cadre du projet VACSIM. Nous allons pour cela essayer de mettre en évidence leurs principales caractéristiques et présenter un exemple d'usage de ces techniques pour analyser les principales causalités dans un réseau génétique.

Nous conclurons ce rapport par l'identification d'une technique qu'il nous semble pertinent d'approfondir dans le cadre de la vérification de programmes ; en particulier sur deux applications réelles, fournies par les industriels du projet et qui sont décrites en annexe, et pour lesquelles nous essayerons de mettre en évidence les analogies structurelles avec les questions biologiques mentionnées ci-dessus.

2 Modèles discrets de réseaux génétiques

À la fin des années 1970, les biologistes ont commencé à faire appel à des modélisations dites « qualitatives » des réseaux génétiques afin de comprendre les chaînes de causalités qui gouvernent le comportement d'une cellule à partir des interactions entre ses gènes. Au fil des années, c'est le cadre introduit par le biologiste René Thomas qui s'est imposé comme le plus adapté et, face à des tailles de réseaux toujours croissantes, cette modélisation a été formalisée au début des années 2000 ([BCRG04]) afin de pouvoir lui appliquer des techniques de vérification et validation similaires à celles du génie logiciel dans le cadre des logiciels critiques. Depuis, plusieurs méthodes spécifiques aux réseaux génétiques ont été développées.

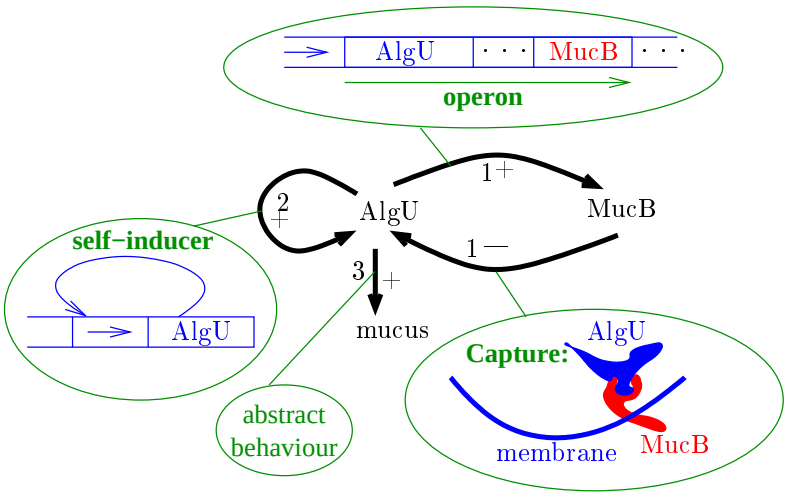
Dans les grandes lignes, cette modélisation repose sur un graphe orienté, où les nœuds représentent les gènes (parfois également des conditions d'environnement ou des observations biologiques du système) et où les arcs représentent l'action d'un gène sur un autre. Les arcs sont étiquetés :

- avec un signe + ou − selon que le gène source de l'arc est connu respectivement comme un activateur ou un inhibiteur du gène cible,

– et avec un seuil entier : les arcs sortants d'un gène donné sont en effet numérotés de 1 à n dans l'ordre d'actions du gène sur ses cibles (en supposant que le gène parte d'un niveau d'activité nul et augmente progressivement sa production de protéines jusqu'à son maximum).

On trouvera la définition précise de ce graphe d'interactions dans [BCRG04], section 2.

Voici un exemple simple. Il s'agit de la production de mucus chez *Pseudomonas aeruginosa*, un microbe mortel pour les malades de mucoviscidose à cause de l'encombrement des poumons par le mucus qu'ils produisent :



AlgU et MucB sont les principaux gènes impliqués. AlgU active MucB par le biais d'un opéron. La protéine produite par MucB est membranaire et elle capture la protéine produite par AlgU, ce qui la séquestre à la périphérie de la cellule et l'empêche de remplir ses fonctions ; MucB est donc un inhibiteur de AlgU. Enfin, la protéine produite par AlgU favorise directement l'activité du gène AlgU et, à haut niveau de concentration, engage la production de mucus.

Le graphe d'interactions est une description *statique* du réseau ; son comportement *dynamique* au cours du temps (c'est-à-dire les variations des taux de concentration des protéines que les gènes considérés produisent) résulte des forces relatives entre les actions dont est l'objet chacun des gènes du graphe. Les actions subies peuvent être contradictoires, elles sont rarement constantes au cours du temps, elles peuvent faire émerger des comportements complexes, avec plusieurs « bassins d'attraction », des comportements oscillatoires et parfois même des situations chaotiques. Ces « forces relatives » d'actions des gènes les uns sur les autres font appel à un nombre considérable de paramètres (cf. sections 3 et 4 du même article) et ces paramètres ne sont malheureusement pas directement mesurables en biologie moléculaire. L'activité d'identification des paramètres, même dans ce cadre discret qui est une simplification de la réalité biologique, est le problème majeur de la modélisation des réseaux génétiques. Dans les grandes lignes, on peut dire que le nombre de paramètres à identifier est exponentiel avec le nombre de gènes du réseau génétique. C'est pour cette raison que sont développés plusieurs méthodes, algorithmes et théorèmes fondamentaux afin de cerner les comportements possibles d'un réseau génétique avec une connaissance imparfaite de la valeur des paramètres.

3 Exploitation de la structure d'un réseau génétique

Une première idée naturelle qui a donné lieu à d'assez nombreux résultats fondamentaux est d'inventorier les comportements qui sont impossibles pour une structure du graphe d'interactions donnée. Dans cette optique, les deux résultats les plus connus sont les suivants :

- *Pour que le réseau génétique puisse exhiber plusieurs bassins d'attraction distincts, il est nécessaire que le graphe d'interactions contienne un cycle dit « positif », c'est-à-dire contenant un nombre pair d'inhibitions.* On peut même déterminer les lignes de « cols » entre les bassins d'attraction en étudiant les seuils mis en jeu dans les cycles qui causent ces bassins d'attraction.
- *Pour qu'un réseau génétique puisse exhiber un comportement oscillatoire sans possibilité d'échappement, amorti ou non, il est nécessaire que le graphe d'interactions contienne un cycle « négatif », c'est-à-dire contenant un nombre impair d'inhibitions.* De même, on peut déduire la position de l'état d'équilibre instable au centre des oscillations à partir de l'étude des seuils des cycles négatifs opérationnels.

Les recherches actuelles dans cette voie [CRA+13], qui ont vu des résultats importants tomber très récemment, portent sur l'impact des intersections entre cycles sur la dynamique du système.

La politique de mise à jour des taux de concentration discrets des gènes à chaque pas de temps est également un aspect important pour l'étude des comportements possibles ou impossibles. Dans le cadre des réseaux génétiques, la politique de mise à jour totalement asynchrone (un seul taux de concentration de protéine est mis à jour à chaque étape) est reconnue comme celle qui est la plus fidèle au comportement *in vivo* et c'est donc celle qui est universellement utilisée. Cependant, pour améliorer l'efficacité des simulations, il est d'un fort intérêt de savoir quels sont les cas de figure où plusieurs gènes/protéines peuvent être mis à jour de manière synchrone (stratégies dites « bloc-synchrones »). Il s'avère en fait que la connaissance de ces synchronismes « neutres » a également des fortes conséquences sur les comportements possibles ou impossibles à atteindre à partir d'un graphe d'interactions donné.

4 Réduction de modèles de réseau génétique

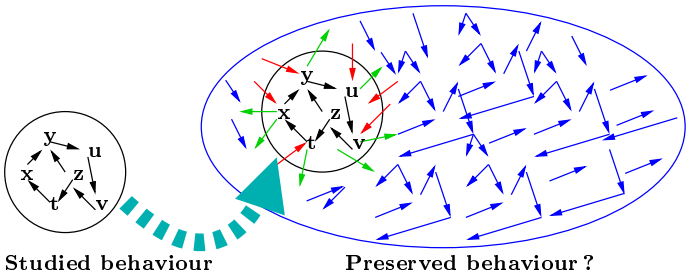
Lorsqu'un biologiste investit dans une modélisation « *in silico* », c'est qu'il souhaite valider une hypothèse biologique suffisamment complexe pour que l'expérimentation biologique seule ne suffise pas à la prouver. Il faut dans ce cas non seulement exploiter la structure du réseau génétique mais aussi identifier les paramètres qui en gouvernent finement la dynamique. Les connaissances biologiques et les résultats d'expériences sont alors autant de contraintes sur les paramètres et l'on poursuit les expériences jusqu'à ce que toutes les valeurs des paramètres compatibles avec ces contraintes vérifient l'hypothèse biologique. La validation d'hypothèse biologique se fait donc par détermination progressive des valeurs possibles des très nombreux paramètres du réseau génétique.

Du fait que le nombre de paramètres varie exponentiellement avec la taille du réseau, une idée naturelle est de réduire la taille du réseau de telle sorte que la réponse à l'hypothèse reste la même dans le modèle réduit. On peut même reformuler l'hypothèse biologique à valider pour l'adapter au réseau réduit :

$$\begin{array}{c}\text{Réseau } R_1\\ \text{satisfait } H_1\end{array} \iff \begin{array}{c}\text{Réseau } R_2\\ \text{satisfait } H_2\end{array} \iff \begin{array}{c}\text{Réseau } R_3\\ \text{satisfait } H_3\end{array} \iff \dots$$

Parmi les techniques de réduction, on peut mentionner celle de Naldi [NRTC09] qui permet de supprimer un gène du réseau génétique sans modifier les états stables du système, sous réserve que ce gène n'ait pas d'influence directe sur lui-même. La méthode est par conséquent valide si l'hypothèse porte sur les états d'équilibre du système.

Une autre méthode consiste à valider l'hypothèse biologique sur un sous-réseau ayant un impact majeur relativement à cette hypothèse, puis à prouver que par plongement du sous-réseau dans le réseau d'origine, on ne remet pas en cause la propriété validée.



Les conditions nécessaires et suffisantes peuvent être assez simplement traduites en contraintes sur les paramètres [BT09, S09, MAC+11]. Certaines de ces techniques sont parfois à rapprocher des techniques d'interprétation abstraite dans le cadre de la validation de programmes.

5 Logique temporelle appliquée aux réseaux génétiques

Il s'agit là de la première technique formelle qui fut appliquée aux réseaux génétiques. Les premiers travaux remontent au début des années 2000 [BCRG04] et ce sont eux qui ont motivé la formalisation complète du cadre de modélisation discrète de René Thomas. Depuis, l'idée a été énormément optimisée et nous passerons sous silence les aspects techniques pour ne présenter que l'idée générale.

Là encore, nous exploitons le fait que si un biologiste investit dans une modélisation *in silico*, c'est qu'il souhaite valider une hypothèse biologique complexe. Par ailleurs, cette hypothèse est en partie étayée par des connaissances biologiques. Le travail de modélisation du réseau génétique consiste alors non seulement à établir le graphe des activations et inhibitions d'un gène sur un autre, mais aussi à formaliser l'hypothèse biologique et les connaissances biologiques. La plupart de ces propriétés biologiques portent sur les comportements possibles du système biologique de sorte que la logique temporelle est le langage le plus adapté pour les formaliser. Certains systèmes préfèrent utiliser CTL [KCRB09], d'autres LTL [MGC+07] mais le principe reste le même :

- Après inventaire des gènes susceptibles d'être impliqués dans la fonction biologique étudiée, on fait l'inventaire des interactions probables entre ces gènes et il en résulte un ensemble de graphes d'interactions possibles.
- Les paramètres qui régissent la dynamique du réseau génétique ne sont généralement pas connus, de sorte qu'il n'est pas rare d'avoir un nombre de dynamiques possibles supérieur à plusieurs tera-modèles à cette étape du processus.
- On formalise en logique temporelle les connaissances biologiques jugées pertinentes pour le problème abordé.
- On formalise en logique temporelle l'hypothèse biologique qui motive les recherches des biologistes.

- On utilise des stratégies d'énumération de modèles sophistiquées et incrémentales afin de soumettre chaque modèle potentiel à des logiciels de preuve : Model Checking et résolution de contraintes sont les outils typiquement les plus utilisés.
- En conditions réelles, l'ensemble des modèles qui satisfont à la fois les connaissances biologiques et les hypothèses *est vide* lors des premières modélisations et formalisations des propriétés biologiques. Ceci malgré l'immense espace des paramétrages possibles. Il faut plusieurs itérations pour mettre au point la modélisation du réseau de manière biologiquement crédible. Il n'est pas rare qu'un gène ou une interaction considérés dans un premier temps par les biologistes comme négligeables ou non pertinents s'avèrent critiques dans le fonctionnement du système. Ces itérations (à l'issue desquelles l'ensemble de modèles est non vide) sont majeures pour l'étude du phénomène biologique car elles permettent d'explorer les propriétés du système complexe que l'on modélise et c'est l'occasion pour les biologistes de faire progresser leur expertise de la question.
- Il arrive aussi que l'ensemble de modèles compatibles avec les connaissances et les hypothèses reste vide à l'issue des itérations. L'hypothèse doit alors être reconsidérée par les biologistes.
- Lorsqu'il n'est pas vide, on a prouvé que l'hypothèse biologique est cohérente et crédible. On n'a pas encore prouvé qu'elle est validée par l'objet biologique réel. C'est généralement le point de départ qui motive des investissements pour effectuer des expériences « à la paillasse » qui permettraient de valider l'hypothèse (cf. section 7).

6 Logique de Hoare et analyse de traces expérimentales

Certaines questions biologiques et certaines cellules biologiques peuvent bénéficier de conditions expérimentales favorables qui permettent *in vivo* une observation « en temps réel » (i.e. avec des pas de temps assez rapprochés) de l'activité des gènes. On dispose alors de traces expérimentales du système étudié qui permettent de dire, à chaque pas de temps et pour chaque gène, s'il est stable, s'il augmente ou s'il diminue.

Dans ce cas, on peut réduire considérablement le nombre de modèles dynamiques à énumérer car les traces observées expérimentalement fournissent des indications exploitables sur les valeurs possibles des paramètres du réseau génétique. En effet, si un gène augmente à un pas de temps donné, cela signifie que le paramètre qui s'applique à ce gène à ce temps là de l'expérience est supérieur à la valeur courante du niveau d'expression du gène.

Il faut alors « pister » le long d'une trace les valeurs des activateurs et inhibiteurs de chaque gène ; de cette information, on extrait les jeux de paramètres qui peuvent s'appliquer à chaque instant et le sens de variation du gène engendre donc une inéquation.

Il s'avère qu'une extension de la logique de Hoare, dédiée à la théorie des réseaux génétiques discrets, permet d'automatiser complètement l'extraction du système d'inéquations qui résulte d'un ensemble de traces expérimentales [Kha10].

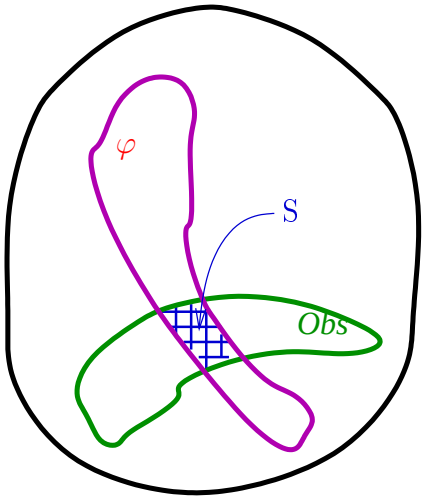
7 Suggestion d'expériences biologiques à partir de modèles

La logique temporelle n'a pas seulement eu un fort succès en biologie parce qu'elle permet de mettre au point les modèles de réseaux génétiques ; elle offre de plus un véritable « pont » entre la modélisation informatique et les expérimentations « humides » effectuées sur l'objet biologique lui-même. En effet, l'observation directe d'une expérience peut la plupart du temps s'exprimer

en logique temporelle ; par exemple « *Après avoir placé les cellules dans un état initial ayant telle et telle propriétés, on observe que tel ou tel gène commence à s'exprimer, puis que tel autre diminue son activité* » est un type d'observation très courant et cette affirmation ne pose pas de difficulté majeure à traduire en formule de logique temporelle.

Mieux : parmi toutes les formules de logique temporelle qui pourraient être écrites, une bonne connaissance des capacités expérimentales d'un laboratoire de biologie permet d'approximer de manière raisonnable l'ensemble des formules temporelles qui peuvent être testées comme vraies ou fausses au moyen d'une expérience. Si l'on note *Obs* cet ensemble de formules expérimentables (en vert dans la figure ci-dessous) alors il devient possible d'utiliser la logique formelle pour guider le choix des expériences aptes à valider une hypothèse biologique :

- Notons φ l'hypothèse biologique que les biologistes souhaitent valider (en rouge dans la figure). La formule φ n'appartient pas à *Obs* car sinon la question serait aisément tranchée par une seule expérience et ne serait donc pas un problème difficile pour les biologistes. En revanche, grâce aux connaissances biologiques, on peut considérer l'ensemble des conséquences de l'hypothèse : $Th(\varphi)$ en violet dans la figure.
- L'ensemble des expériences susceptibles de valider l'hypothèse est évidemment $S = Obs \cap Th(\varphi)$ (en bleu dans la figure). Cet ensemble est malheureusement généralement trop grand pour pouvoir envisager de faire toutes les expériences ; il est même infini.
- Dans un premier temps, on vérifie que l'hypothèse biologique est en adéquation avec les capacités expérimentales : par des méthodes liées à la théorie des automates, on prouve que S implique φ, compte-tenu des connaissances biologiques. Si tel n'est pas le cas, c'est que l'hypothèse considérée par les biologistes est trop ambitieuse relativement à leurs capacités expérimentales.
- Enfin dans un second temps, on cherche à extraire un sous-ensemble fini et de taille raisonnable E de S tel que E implique S. C'est un ensemble d'expériences adéquat pour valider l'hypothèse. Là encore, la théorie des automates aide à faire cette sélection et des heuristiques sont souvent nécessaires.

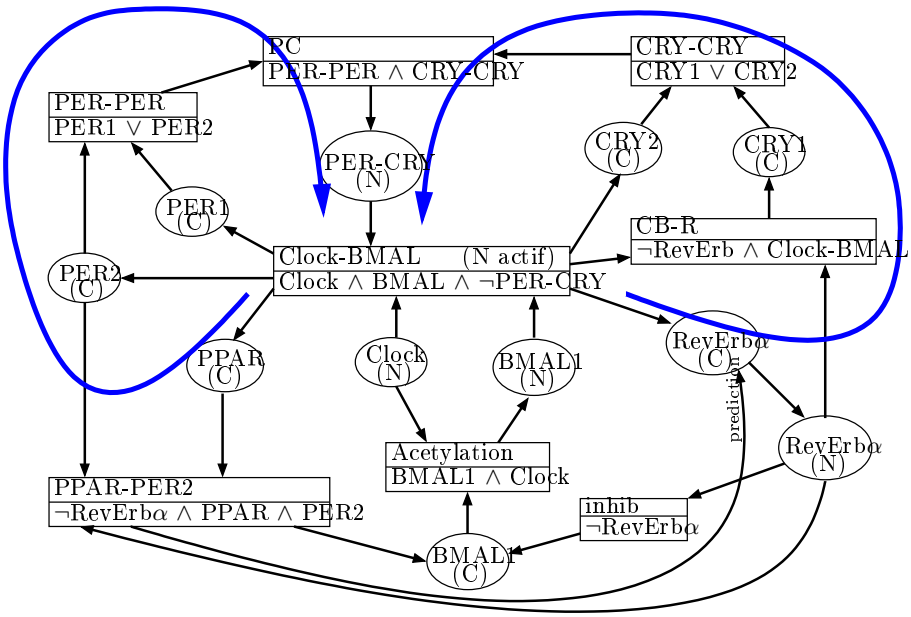


Une vulgarisation de cette approche est décrite dans [BCG07].

8 Horloge circadienne et jet-lag

Les différentes méthodes et techniques présentées dans ce document permettent d'aborder de manière raisonnée un problème biologique *a priori* complexe à propos d'un réseau génétique. Les comportements possibles d'un système complexe comme les systèmes biologiques rendent illusoire l'idée de modéliser complètement une cellule ou même une fonction biologique dans toute sa richesse. La démarche est donc d'orienter à la fois la modélisation et les techniques de validation vers l'hypothèse à valider et elle seule. Ainsi, partant d'un réseau génétique généralement un peu trop complexe pour autoriser des raisonnements rationnels, les propriétés générales des réseaux génétiques sont exploitées pour simplifier la question, simplifier le modèle considéré, dégager les incohérences de manière automatique et enfin défricher les stratégies expérimentales capables de valider l'hypothèse considérée.

Un exemple de taille abordable (mais assez révélateur du niveau de compréhension des phénomènes complexes que fournissent les techniques de modélisation en biologie) est celui de l'*horloge circadienne*. Il s'agit du cycle éveil-sommeil, ici chez les mammifères, donc en particulier chez l'Homme. Ce cycle est relativement central et son impact dans le domaine de la santé est considérable (en fort lien avec la rapidité de guérison, importance de la chronothérapie, fort pourcentage de la population travaillant en horaires décalés, jet-lag, *etc*). Une douzaine de gènes contrôlent le cycle éveil-sommeil. La figure ci-dessous représente de manière simplifiée le réseau génétique correspondant :



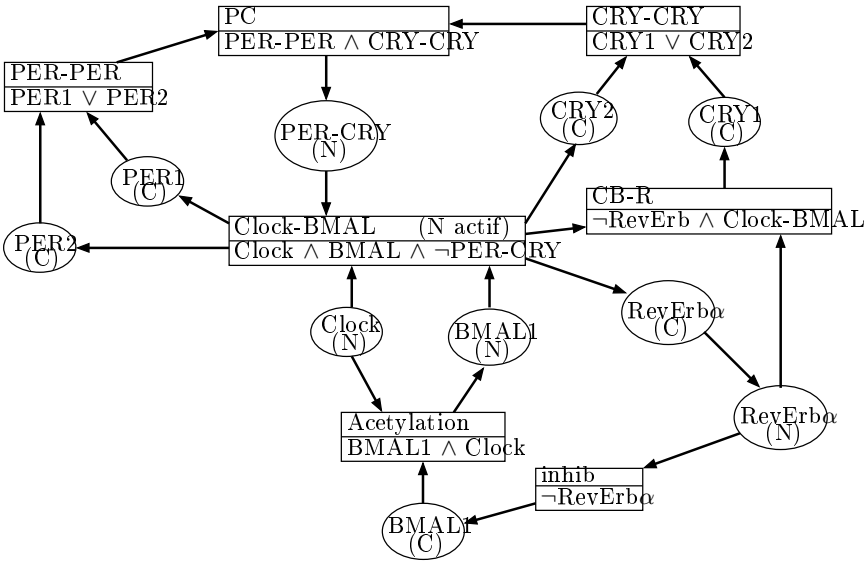
Les formules dans des rectangles indiquent des modes de coopération connus entre les gènes (formules du premier ordre). Les « gènes de l'horloge » sont principalement Clock, BMAL, PER, CRY, PPAR et RevErbα et plusieurs variantes seulement partiellement représentées ici (ovales). Ce graphe d'interactions est un inventaire simplifié mais qui reste raisonnable des connaissances biologiques « statiques ».

De toute évidence, ce graphe ne permet pas de se faire une idée du mode de fonctionnement de l'horloge circadienne, même si l'on précise que de jour les nœuds portant un (N) sont partiellement inhibés par la lumière. L'hypothèse biologique à valider ici est que cette simplification qui ne

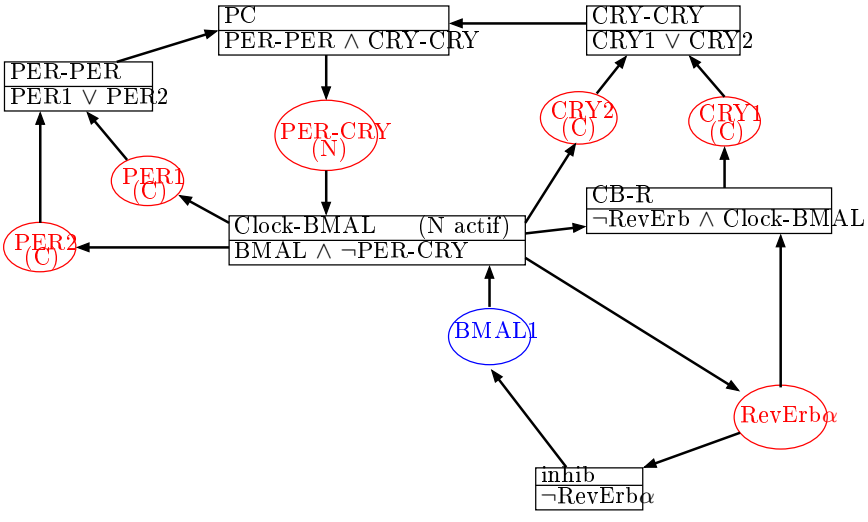
considère qu'une sous-partie des gènes impliqués suffit à apporter au système une résistance aux variations de durée du jour (saisons) et le retour à un cycle normal après un jet-lag.

Les biologistes précisent par ailleurs que le cycle éveil-sommeil est principalement causé par les protéines PER et CRY, qui forment des complexes protéiques PER-CRY de différentes natures, et que lorsque PER-CRY peut entrer dans le noyau de la cellule, symbolisé par le (N), il inhibe les gènes Clock et BMAL. On a donc affaire à deux cycles qui s'intersectent en PER-CRY (en bleu sur la figure) et ces cycles sont négatifs (nombre impair d'inhibitions, c'est-à-dire de négations le long du cycle) donc potentiellement porteurs de l'oscillation éveil-sommeil.

Par des techniques de plongement et en utilisant quelques connaissances biologiques complémentaires, on peut démontrer que le sous-graphe suivant résistera aux saisons et au jet-lag si et seulement si le graphe global le fait (technique de plongement de la section 4) :



Dès lors, les techniques de suppression de gènes (partie « Naldi » de la section 4) s'appliquent :



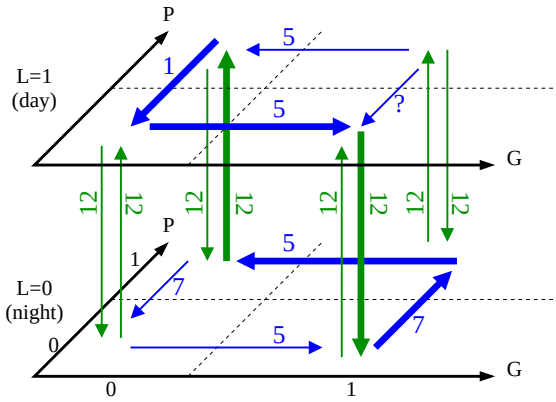
Et enfin, on montre qu'il est possible de fusionner les inhibiteurs de Clock et BMAL (en rouge),

de sorte que l'on prouve que nos hypothèses seront valides dans le graphe complexe du début si et seulement si elles le sont dans le graphe simplifié suivant (à gauche) :

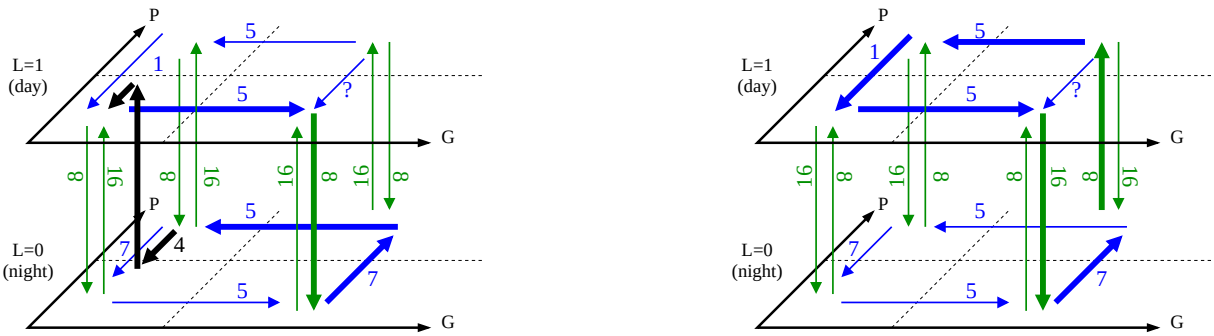
et l'action de la lumière étant d'empêcher le passage des protéines inhibitrices dans le noyau, on complète le graphe (à droite) avec une inhibition de la lumière sur les protéines nucléaires.

La présence d'un cycle négatif (les gènes activent des protéines qui les inhibent) rend crédible l'existence d'oscillations. En l'absence de cycle après simplification, l'hypothèse aurait été réfutée (et donc devant l'évidente existence du cycle éveil-sommeil, cela aurait indiqué des oublis dans le graphe d'interactions de départ).

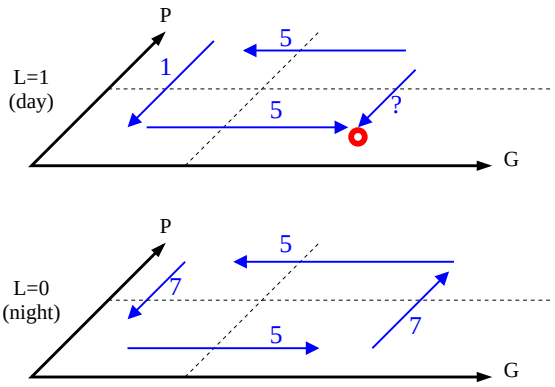
Le degré sortant de chacun des nœuds du graphe d'interactions simplifié étant égal à 1, il s'agit d'un réseau booléen (valeurs 0 ou 1). Tracer l'espace des états devient alors facile, et (modulo une extension avec délais de la théorie des réseaux génétiques qu'on trouvera dans [CBD+12]) on peut annoter les changements d'états par le temps nécessaire à leur accomplissement :



Horizontalement, G représente globalement les gènes de l'horloge (0=inactifs, 1=actifs). En profondeur, P représente l'entrée des protéines dans le noyau de la cellule (0=absentes, 1=présentes). Verticalement, les alternances jour-nuit L en vert prennent 12 heures (0=peu ou pas de lumière, 1=lumière du jour).
Les transitions en gras indiquent le cycle suivi pour une alternance jour-nuit de 12h-12h. Il devient alors facile de prouver que ce cycle perdure en hiver (8h-16h à gauche) et en été (16h-8h) :



Concernant le retour à un cycle « normal » après un jet-lag, le modèle simplifié permet une clarté de vue que le modèle de départ n'aurait jamais permise. Si l'on supprime les transitions verticales d'alternance jour-nuit, on constate que durant la nuit, le cycle n'est pas beaucoup perturbé, alors que durant le jour il est « mis en attente » dans un état stable dont il ne peut pas s'échapper (G=1,P=0,L=1) :



Ceci explique qu'il soit plus facile de rattraper un jet-lag en allongeant la journée plutôt qu'en l'écourtant : un allongement de la journée fait attendre le cycle dans son état stable, alors qu'un raccourcissement excessif de la nuit risque de faire passer le système dans l'état de lumière au milieu de la transition inverse du « sens de rotation » (dont on connaît mal la durée comme l'indique le point d'interrogation), créant les difficultés qu'on connaît.

En résumé, comme on le voit en partie sur ce petit exemple, les techniques développées dans le domaine des réseaux génétiques font beaucoup appel à des transformations du modèle à valider, ou bien à des théorèmes connus sur l'impact des boucles de rétroaction, ou encore à des transformations de la propriété à valider. Ces techniques rendent possibles des validations de propriétés qui seraient inaccessibles sur le réseau complexe de départ.

9 Potentiels et apports possibles « bio-inspirés » pour la validation des logiciels embarqués

Les logiques de Hoare ont déjà été largement explorées pour la validation des logiciels embarqués et cette piste nous semble donc peu prometteuse. Par ailleurs, l'expérimentation biologique est similaire au test et elle ne permet donc pas d'effectuer une validation pleine et entière. Par contre, le graphe d'interactions est central dans toutes les autres techniques et nous semble la voie la plus intéressante à explorer. L'idée est d'extraire un graphe d'interactions à partir du code à valider puis d'appliquer des techniques, par exemple de réduction, inspirées de la biologie en utilisant la post-condition d'une manière similaire à l'"hypothèse biologique" des réseaux génétiques. L'objectif est de vérifier si les résultats sont meilleurs que le slicing classique ; en d'autres termes, s'il est possible de réduire davantage le graphe de flot de contrôle sans perdre d'information pertinente.

Concrètement, nous proposons d'évaluer l'apport de ces techniques sur deux applications réelles que nous avons déjà traitées dans le cadre des projets TESTEC et VACSIM en liaison avec des partenaires du monde industriel, à savoir : un programme de gestion des clignotants et le programme de contrôle d'un système anti-blocage des roues (ABS).

Le programme de gestion des clignotants (Flasher Manager) est une application industrielle temps-réel provenant d'un fabricant automobile. L'application a été conçue et simulée en Simulink [1]. Cependant, concrètement, l'application est un programme C embarqué sur un des calculateurs d'une voiture. Le programme C est généré automatiquement à partir du modèle Simulink, sans garantie formelle sur la correction de la génération. Cette application est intéressante pour la vérification logicielle du fait de sa complexité qui provient à la fois de la fonction C générée et du grand nombre de cycles de la boucle temps-réel nécessaire pour la vérification des propriétés. Nous nous sommes intéressés à quatre propriétés définies par notre partenaire industriel. Une description détaillée du système et des propriétés est fournie en Annexe A. Notre stratégie de model checking borné DPVS [CLV+12] nous a permis de prouver la validité d'une propriété et de générer des contre-exemples pour une autre propriété. Pour deux propriétés, notre outil n'a pas pu répondre dans le temps imparti fixé à 10 minutes.

Le programme de contrôle d'un système anti-blocage des roues (ABS) est une application logicielle temps-réel qui s'exécute sur un calculateur embarqué dans un véhicule. Le modèle du système a été conçu en Simulink et le code embarqué a été automatiquement généré à partir du modèle Simulink. Le code comporte des opérations sur les entiers et sur les flottants. Il consiste en une boucle infinie qui lit les entrées et calcule la sortie toutes les 0.01 s. Nous nous sommes intéressés à une propriété de sûreté P_1 définie par notre partenaire industriel. La propriété spécifie que l'ABS entre en mode de freinage régulé dès qu'une roue glisse plus qu'un seuil fixé. Notre outil de vérification des programmes flottants rAiCp [PMR12,PMR13] nous a permis de montrer que la propriété P_1 était valide pour une durée d'activation de 20 s du système. Cette durée, établie en accord avec notre partenaire, correspond à une estimation prudente d'une durée d'activation réaliste pour un véhicule avec une vitesse maximale de 180 kmh. Une description plus détaillée du système et des propriétés est fournie en Annexe B.

Nous nous proposons de travailler sur deux autres propriétés de l'ABS, P_2 et P_3, pour évaluer l'apport possible des techniques utilisées en bio-informatique sur les graphes d'interactions. Ces propriétés sont la formalisation d'une autre exigence du partenaire industriel portant sur le comportement de l'ABS lors du blocage d'une roue. Les propriétés du Flasher Manager que DPVS a échoué à traiter dans le temps imparti pourront aussi compléter l'évaluation des techniques issues de la bio-informatique.

1. Simulink est une plate-forme de modélisation de systèmes dynamiques multidomaines.

10 Bibliographie

[BCG07] G. Bernot, J.-P. Comet, J. Guespin : *Méthode informatique de découverte du fonctionnement d'un réseau de régulation biologique,* Biofutur (Mensuel de vulgarisation), numéro spécial biologie intégrative, Num.275, p.22-25, 2007.

[BCRG04] G. Bernot, J-P. Comet, A. Richard, J. Guespin : *Application of formal methods to biological regulatory networks: Extending Thomas' asynchronous logical approach with temporal logic,* J. of Theoretical Biology (JTB), Vol.229, Issue 3, p.339-347, 2004.

[BT09] G. Bernot, F. Tahi : *Behaviour Preservation of a Biological Regulatory Network when Embedded into a Larger Network,* Fundamenta Informaticae, IOS Press Amsterdam, Vol.91, Issue.3-4, p.463-485, ISSN:0169-2968, 2009.

[CBD+12] J.-P. Comet, G. Bernot, A. Das, F. Diener, C. Massot, A. Cessieux : *Simplified models for the mammalian circadian clock.,* Book chapter In Proc. of the Evry Spring school on Modelling complex biological systems in the context of genomics, May 21st-25th, 11th Edition, Amar, Képès, Norris Ed., EDP Science pub., p.85-106, 2012.

[CLV+12] H. Collavizza, N. Le Vinh, O. Ponsini, M. Rueher, A. Rollet : *Constraint-Based BMC: A Backjumping Strategy,* International Journal on Software Tools for Technology Transfer, DOI: 10.1007/s10009-012-0258-6, springer, 2012.

[CRA+13] J.-P. Comet, A. Richard, J. Aracena, L. Calzone, J. Demongeot, M. Kaufman, A. Naldi, H. Snoussi, D. Thieffry : *On circuit functionality in Boolean networks,* Bulletin of Mathematical Biology, 2013.

[KCRB09] Z. Khalis, J.-P. Comet, A. Richard, G. Bernot : *The SMBioNet Method for Discovering Models of Gene Regulatory Networks,* Invited review, Genes Genomes and Genomics, A. Mansour Ed., Global Science Books, Vol.3, Special Issue 1, p.15-22, ISSN:1749-0383, ISBN 978-4-903313-33-7, 2009.

[Kha10] Z. Khalis : *Logique de Hoare et identification formelle des paramètres d'un réseau génétique,* PhD thesis, Genopole - University of Evry-Val d'Essonne, 2010.

[MAC+11] M. Mabrouki, M. Aiguier, J.-P. Comet, P. Le Gall, A. Richard : *Embedding of biological regulatory networks and properties preservation,* Mathematics in Computer Science 5(3):263-288, special issue, 2011.

[MGC+07] D. Mateus, J.-P. Gallois, J.-P. Comet, P Le Gall : *Symbolic modeling of genetic regulatory networks,* J. of Bioinformatics and Computational Biology, 5(2B):627-640, 2007.

[NRTC09] Naldi, Remy, Thieffry, Chaouiya : *A reduction of logical models of regulatory networks yields insight into dynamical properties,* CMSB'09, LNBI 5688:266-280, Springer, 2009.

[PMR13] O. Ponsini, C. Michel, M. Rueher : *Verifying floating-point programs with constraint programming and abstract interpretation techniques,* `http://hal.archives-ouvertes.fr/hal-00860681`, submitted, 2013.

[PMR12] O. Ponsini, C. Michel, M. Rueher : *Refining abstract interpretation based value analysis with constraint programming techniques,* CP'12, LNCS 7514:593-607, springer, 2012.

[S09] H Siebert : *Dynamical and structural modularity of discrete regulatory networks,* Computational Models for Cell Processes, CompMod 2009, Eindhoven, Netherlands, volume 6 of EPTCS, pages 109-124, 2009.

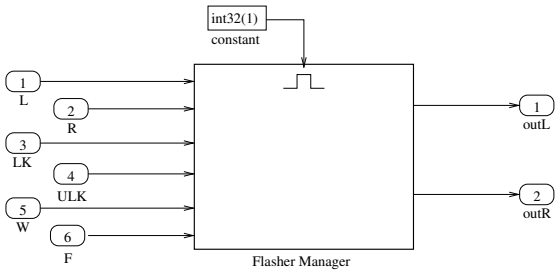


Figure 1: Flasher Manager simplified Simulink model

A Flasher Manager

We first describe the Simulink module of the Flasher Manager, then the properties we had to check. A description of this application—with all source code—can be found at `http://hal.archives-ouvertes.fr/hal-00720921`.

A.1 Description of the Simulink module

The Flasher Manager is a controller that drives several functions related to the flashing lights of a car. Each function is enabled by some input commands, activates one or two flashing lights, and is described by its duration and its flashing period (i.e., time-units required to oscillate from 1 to 0 or 0 to 1). The next subsections detail these points for the three main functionalities of the Flasher Manager. Figure 1 shows a simplified Simulink model (i.e., inputs/outputs) and Fig. 2 provides a more detailed model.

A.1.1 Direction change

When the driver indicates a direction change, Boolean input `R` or `L` rises from 0 to 1. The corresponding light (respectively driven by the `outR` or `outL` output) then oscillates between on/off states with a period of 6 time-units (typically 3 seconds). Thus, an output sequence of the form [111000] is repeated on one of the lights. Then, when the input falls back to 0, the corresponding output light stops flashing. The light starts oscillating immediately when the command is enabled, and stops immediately when the command is disabled. These are the `Flashers_left` and `Flashers_right` functions.

A.1.2 Lock and unlock of the car

The driver has the ability to lock and unlock the car from the distance using an RF-key. The state of the unlock and lock buttons of the key is reported to Boolean inputs `ULK` and `LK` respectively.

When an RF-key is pressed, the manager indicates the state of the doors to the user using the following convention:

– If the unlock button is pressed while the car is unlocked, nothing shall happen.

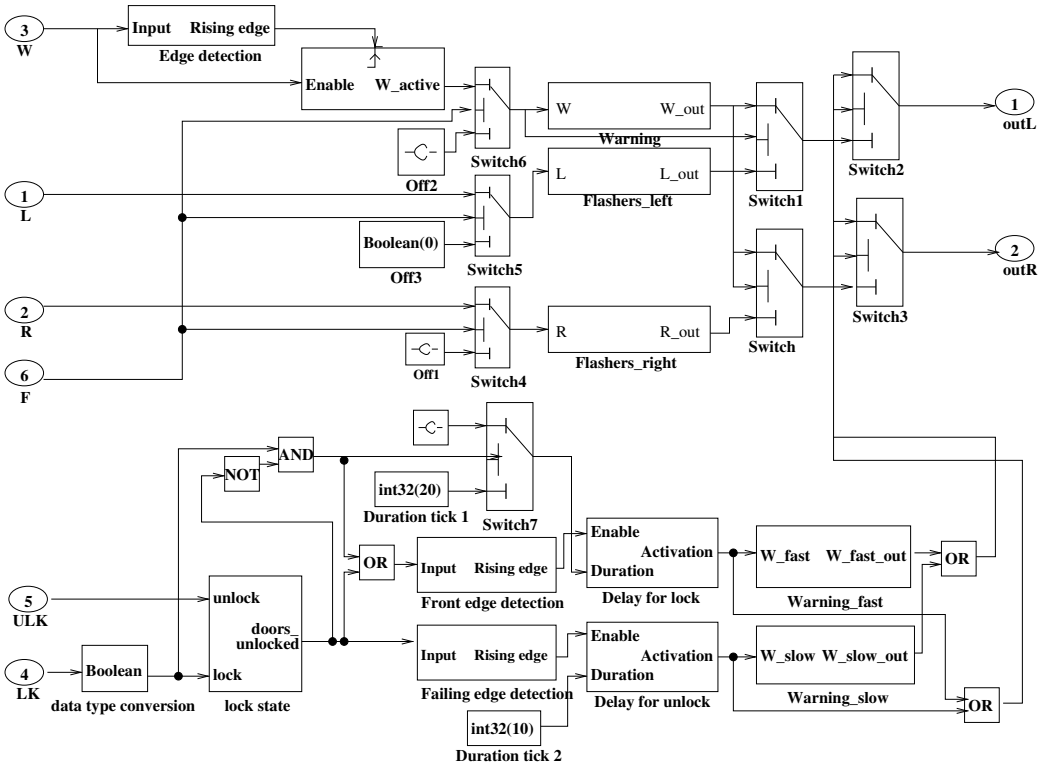


Figure 2: Flasher Manager detailed Simulink model

- If the unlock button is pressed while the car is locked, both lights shall flash with a period of 2 time-units during 20 time-units (fast flashes for a short time). More precisely, when the `ULK` input is activated, the oscillation starts two cycles after the activation, produces an output sequence of the form [10101...010] on both lights and stops on the 22^{nd} cycle after the activation. This is the `Unlock_flash` function.
- If the lock button is pressed while the car is unlocked, both lights shall go on for 10 time-units, and then shall go off for another 10 time-units, producing an output sequence of the form [11111111110000000000] on both lights.
- If the lock button is pressed while the car is locked, both lights shall flash during 60 time-units with a period of 2 time-units (fast flashes for a long time). More precisely, when the `LK` input is activated, the oscillation starts the next cycle, produces an output sequence of the form [10101...010] on both lights that stops on the 61^{st} cycle. This is the `Lock_flash` function. It is typically used to locate the car in an over-filled place.

Note that in the initial state, the doors are locked.

A.1.3 Warning function

Finally, the driver has the ability to press the warning button. When the warning is on, both lights flash with a period of 6 time-units (slow flashes). This is the `Warning` function. The `W` input is a *push-down* button. In the initial state of the manager, the warning is off. A rising edge of `W` activates the warning and the next rising edge of `W` deactivates the warning.

A.2 Properties

We checked four properties of the Flasher Manager module.

Property 1: "`Warning` function has priority over other flashing functions."

Property 2: "When the warning button has been pushed and then released, the `Warning` function resumes to the `Flashers_left` (or `Flashers_right`) function, if this function was active when the warning button was pushed."

Property 3: "When the `F` signal (for flasher active) is off, then the `Warning`, `Flashers_left`, and `Flashers_right` functions are disabled. On the contrary, all the functions related to the lock and unlock of the car are maintained."

Property 4: "Lights should never remain lit infinitely."

Restrictions For checking the four properties, we adopted the following restrictions in accordance with the designers of the Flasher Manager module:

1. `L` and `R` inputs cannot both be TRUE on the same cycle;
2. `LK` and `ULK` inputs cannot both be TRUE on the same cycle.

First restriction means that we do not consider a degraded use when the lever of the indicators is damaged. Second restriction excludes a misuse of the RF-key.

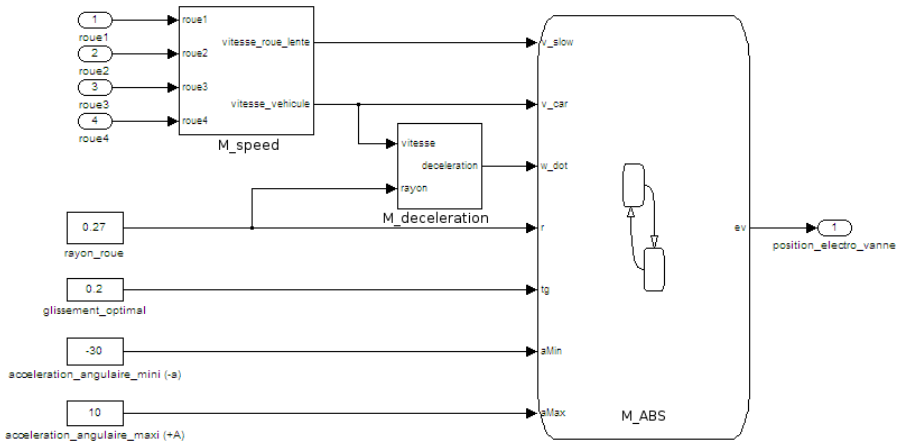


Figure 3: ABS Simulink model

B Anti-lock Braking System

B.1 Description

ABS prevents wheel lock when braking. It monitors wheel speed through sensors and acts on an hydraulic valve. The valve can have three positions:

- In position 0, or *passing mode*, the pressure on the brake pedal is transmitted right through to the brakes.
- In position 1, or *maintaining mode*, the valve moves to block the brake line. The pressure on brakes remain constant even if the driver tries to push the brake pedal harder.
- In position 2, or *reduction mode*, the valve moves to release pressure from the brake line.

The Simulink model of ABS is in Fig. 3. On the left, module M_speed computes the vehicle speed, v_{car}. The vehicle speed is the mean of the rotational speed of the wheels. This module also determines the speed v_{slow} of the slowest wheel. Then, module M_deceleration computes the deceleration of the vehicle from the difference of speed between two samples. Finally, module M_ABS implements the control of the valve.

M_ABS is described by the state diagram of Fig. 4. Initially, the valve is in passing mode. ABS looks for the tendency to lock of a wheel. It computes the skidding rate of the slowest wheel as $r_s = 1 - \frac{v_{\text{slow}}}{v_{\text{car}}}$. ABS tries to maintain the optimal rate $r_o = 20\%$ [2]. When r_s is greater than r_o, ABS starts controlling braking. It first enters maintaining mode. If the tendency to lock is confirmed, it enters reduction mode and releases pressure until the wheel accelerates again. Then, ABS alternates phases in passing mode to increase pressure and phases in reduction mode to release pressure, always going back to maintaining mode in between. The phase changes are determined by a minimum acceleration threshold aMin to leave the reduction mode and a maximum acceleration threshold aMax to enter the passing mode. As soon as the tendency to lock disappears, ABS stops controlling braking. Moreover, when ABS controls braking, the vehicle speed cannot be computed from the wheel speeds anymore: it is computed from the last known speed and decreased assuming a constant deceleration equals to the last one known.

2. Actually, optimal rate depends on the road surface and varies between 30% and 10% .

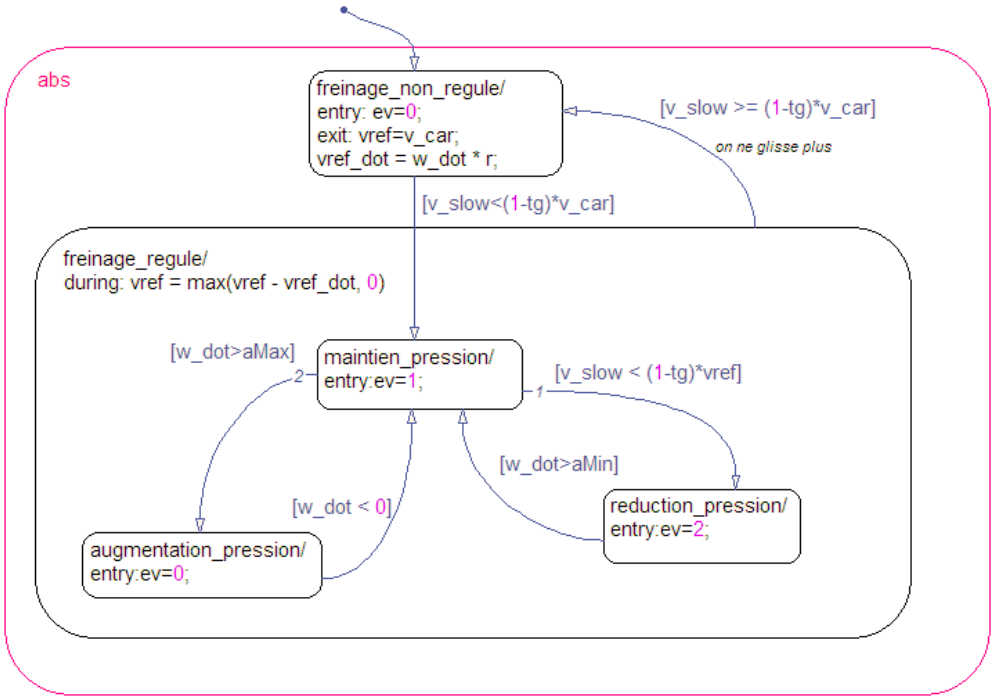


Figure 4: ABS Simulink state diagram

B.2 Properties

Our industrial partner had specified two informal properties:

- ABS enters controlled braking as soon as skidding rate is greater than 20% (P_1).
- As long as a wheel is locked, the valve must be in position 2 (reduction mode). Since this statement is ambiguous, we refined the property into two new properties:
  - As soon as a wheel is locked, the valve must be in position 2 (P_2).
  - If the slowest wheel is locked during two consecutive samples, the valve must be in position 2 (P_3).

For property P_1, the state of the ABS has to be read in an internal variable abs_state of module M_abs. This variable should take one of two predefined values: CONTROLLED or UNCONTROLLED. The assertion to be checked for P_1 is then:

$$(v_{\text{slow}} < 0.8 * v_{\text{car}}) \implies (\text{abs\_state} = \text{CONTROLLED})$$

The position of the valve is modeled in source code by variable valve, which takes value 0, 1, or 2 accordingly to position number. Thus, the property P_2 was formalized as the following assertion:

$$(v_{\text{slow}} = 0) \implies (\text{valve} = 2)$$

Property P_2 implies property P_3: if property P_2 does not hold, we will have to check property P_3 too. For property P_3, we need to store the current value of the slowest wheel in order to access it at the next sample time. Hence, we introduced variable old_v_{slow} to store this value. Property P_3 is then formalized as the following assertion:

$$((\text{old\_}v_{\text{slow}} = 0) \wedge (v_{\text{slow}} = 0)) \implies (\text{valve} = 2)$$

The first unfolding of the real time loop corresponds to an initialization cycle of the system. Thus, the properties were checked after this first cycle. Moreover, we were not provided with a model of the vehicle behavior. We only assumed that the wheel speed was comprised between 0 and 180 kilometers per hour.