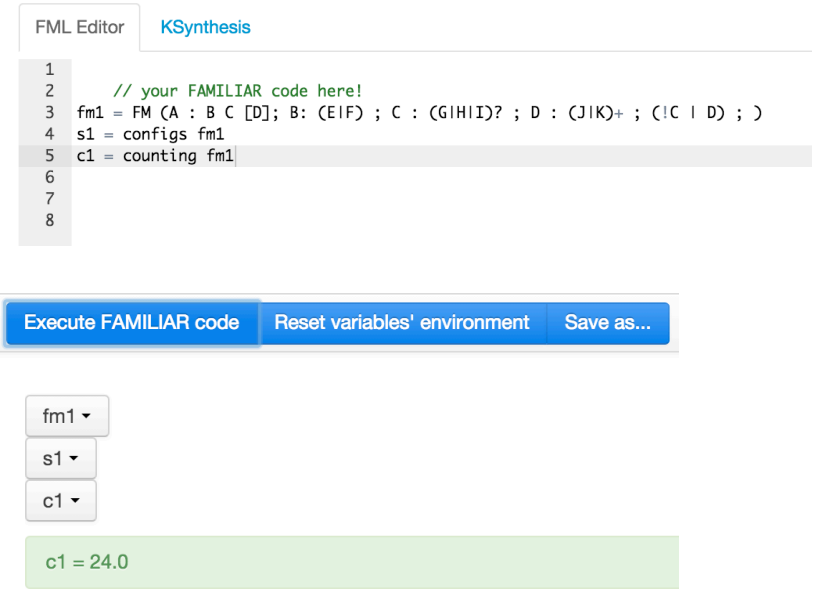
TP 3 : Ligne de produits, feature model

Prérequis / Installation

Nous allons utiliser un environnement et un langage pour éditer et raisonner sur des feature models -- FAMILIAR.
Il est donc nécessaire d'installer FAMILIAR: c'est une application Web écrite en Play (https://www.playframework.com) qui lance un serveur Web et qui est ensuite accessible via un navigateur à l'adresse :
http://127.0.0.1:9000/ide/familiar

Pour récupérer FAMILIAR : http://mathieuacher.com/teaching/MDI/fmlapp-1.0-SNAPSHOT.tgz

Il y a dans l'archive un **fmlapp**.bat (./fmlapp sous Linux/MacOS) dans le dossier **bin/** qui permet de lancer le serveur. Vous pouvez commencer le travail et exécuter votre premier script



Les « opérations » de FAMILIAR (configs, cores, counting, isValid, etc.) sont documentées ici:
https://github.com/FAMILIAR-project/familiar-documentation/blob/master/manual/

Exercice 1. Syntaxe et sémantique des feature models (avec un outil)

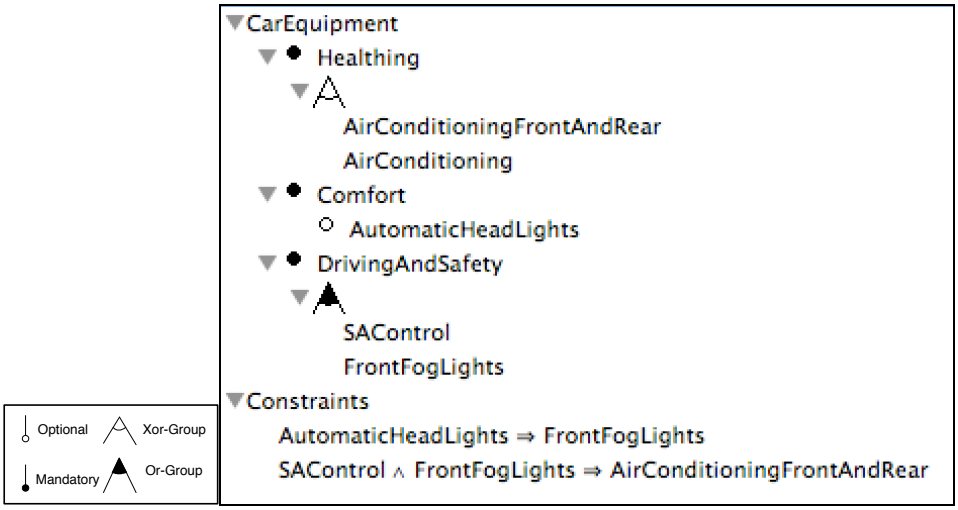


Figure 1.

Question #0: Spécifier le feature model de la Figure 1 avec FAMILIAR, en utilisant la notation interne textuelle : https://github.com/FAMILIAR-project/familiar-documentation/blob/master/manual/featuremodel.md

Question #1: Vérifier que votre spécification textuelle est conforme à la Figure 1 en réitérant les exercices du TD, notamment en produisant une énumération exhaustive de l'ensemble des configurations valides avec l'opération « configs »

Question #2: Quelles sont les features qui sont incluses dans n'importe quelle configuration? Utiliser l'opération « cores »

Question #4: Proposer un feature model avec une hiérarchie différente mais caractérisant le même ensemble de configurations. Vérifier le résultat en utilisant l'opération « compare »

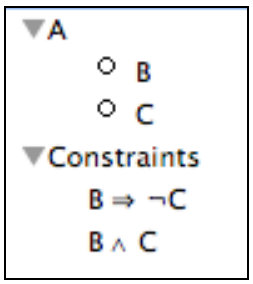


Figure 2

Question #5: Spécifier le feature model de la Figure 2 avec FAMILIAR. Vérifier que le nombre de configurations valides du feature model de la Figure 2 est 0 en utilisant l'opération « counting » et « isValid ».

Question #6: Que se passe-t-il si on relâche la première contrainte? Que se passe-t-il si on relâche la deuxième contrainte? Adresser la question avec les opérations de FAMILIAR.

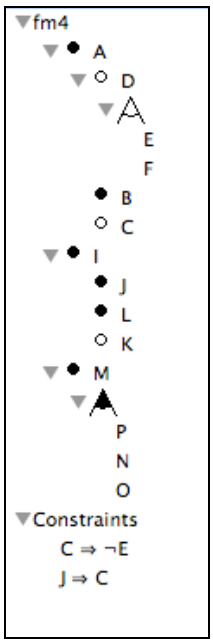


Figure 3

Question #7: Spécifier le feature model de la Figure 3 avec FAMILIAR.
Enumérez l'ensemble des configurations valides.
Que peut-on dire sur les features C et F ?
Que peut-on dire sur la feature E ?

Utiliser les opérations « deads » et « falseOptionals » pour vérifier vos dires.

Question #8: Corrigez les « anomalies », i.e., réécrire le feature model de manière à ce qu'il exprime le même ensemble de configuration mais cette fois-ci les informations de variabilité sont en adéquation avec les configurations valides du feature model.
Vérifier avec FAMILIAR que les anomalies sont bien corrigées, i.e., qu'elles ne sont plus présentes.

Question #9: Que se passe-t-il si on relâche la première contrainte? Que se passe-t-il si on relâche la deuxième contrainte? Réitérez la série de questions précédentes avec FAMILIAR pour chaque suppression de contrainte.

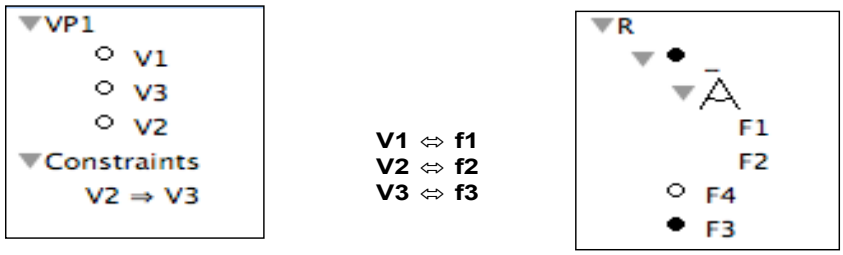


Figure 4

Une organisation veut proposer à ses clients des options de configuration (features) et décident de modéliser les configurations autorisées avec le feature model de gauche.

Question #10: Faîtes une énumération exhaustive de l'ensemble des configurations valides du feature model de gauche avec FAMILIAR.

Cette même organisation a une plateforme logicielle dans laquelle il y a de la variabilité – une feature est optionnelle et il y a deux alternatives d'implémentation. Le feature model de gauche représente cette variabilité logicielle.

Question #11: Faîtes une énumération exhaustive de l'ensemble des configurations valides du feature model de droite avec FAMILIAR.

L'idée de l'organisation est, qu'étant donné une configuration choisie par un client (et valide par rapport au feature model de gauche), il y a au moins une configuration logicielle correspondante (et valide par rapport au feature model de droite).
Ainsi le client peut configurer son produit sans aborder les détails techniques de la plateforme logicielle.

Il y a une correspondance entre le feature model de gauche et le feature model de droite sous la forme de contraintes.

Question #12: Faîtes une énumération exhaustive de l'ensemble des configurations valides du feature model de gauche et de droite une fois que la correspondance entre les deux a été établie via les contraintes. Que peut-on en conclure ?

Exercice 2. ffmpeg (analyse d'une ligne de produits)

Question #13 : Etudier la documentation et localiser la variabilité de ffmpeg (à la compilation ou à l'exécution)
Question #14 : Etudier le code source de ffmeg (http://git.videolan.org/?p=ffmpeg.git;a=tree) et localiser la variabilité de ffmpeg

Question #15 : Choisir une option de ffmpeg dans la documentation et expliquer comment elle est parsée et prise en compte dans le code.

Mathieu Acher 22/3/15 15:20
Commentaire [1]: https://www.ffmpeg.org/ffmpeg.html#Main-options

Mathieu Acher 22/3/15 15:20
Commentaire [2]: http://git.videolan.org/?p=ffmpeg.git;a=blob_plain;f=ffmpeg.c;hb=HEAD
http://git.videolan.org/?p=ffmpeg.git;a=blob_plain;f=ffmpeg.h;hb=HEAD

Exercice 3. Configurateur

Question #16 : Produire le feature model d'un configurateur de votre choix