TP 3 : Ligne de produits, feature model

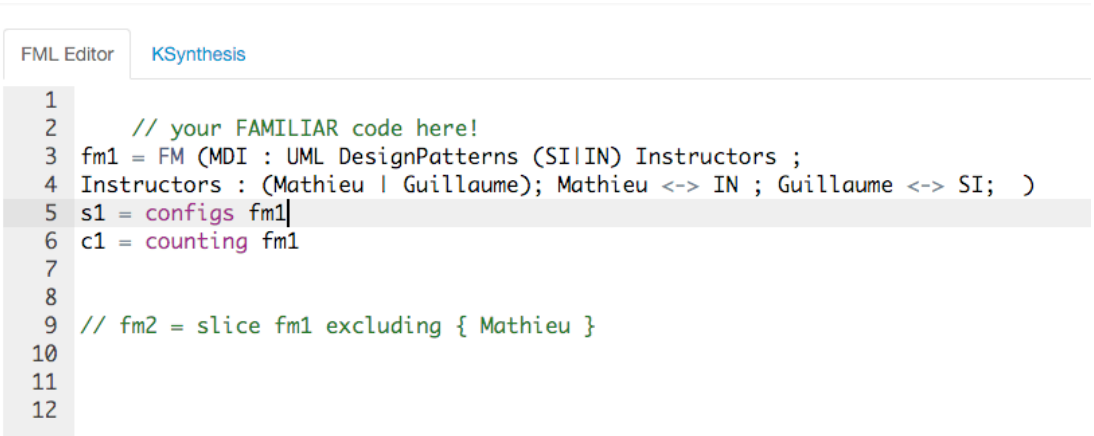
Prérequis / Installation

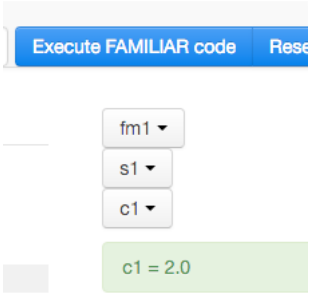
Nous allons utiliser un environnement et un langage pour éditer et raisonner sur des feature models -- FAMILIAR.
Il est donc nécessaire d'installer FAMILIAR: c'est une application Web écrite en Play (https://www.playframework.com) qui lance un serveur Web et qui est ensuite accessible via un navigateur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

Pour récupérer FAMILIAR: http://tinyurl.com/FAMILIAR-MDI1516
(archive zip de 119 Mo)

Il y a dans l'archive un **fmlapp** sous Linux/MacOS dans le dossier **bin/** qui permet de lancer le serveur. Pour les utilisateurs de Windows il y a un fichier **fmapp.bat** dans **bin/** mais il ne fonctionnera peut-être pas pour les raisons évoquées ici: https://support.microsoft.com/fr-fr/kb/830473
Donc il faut utiliser la ligne de commande suivante :
java -cp "./lib/*;" play.core.server.NettyServer

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)

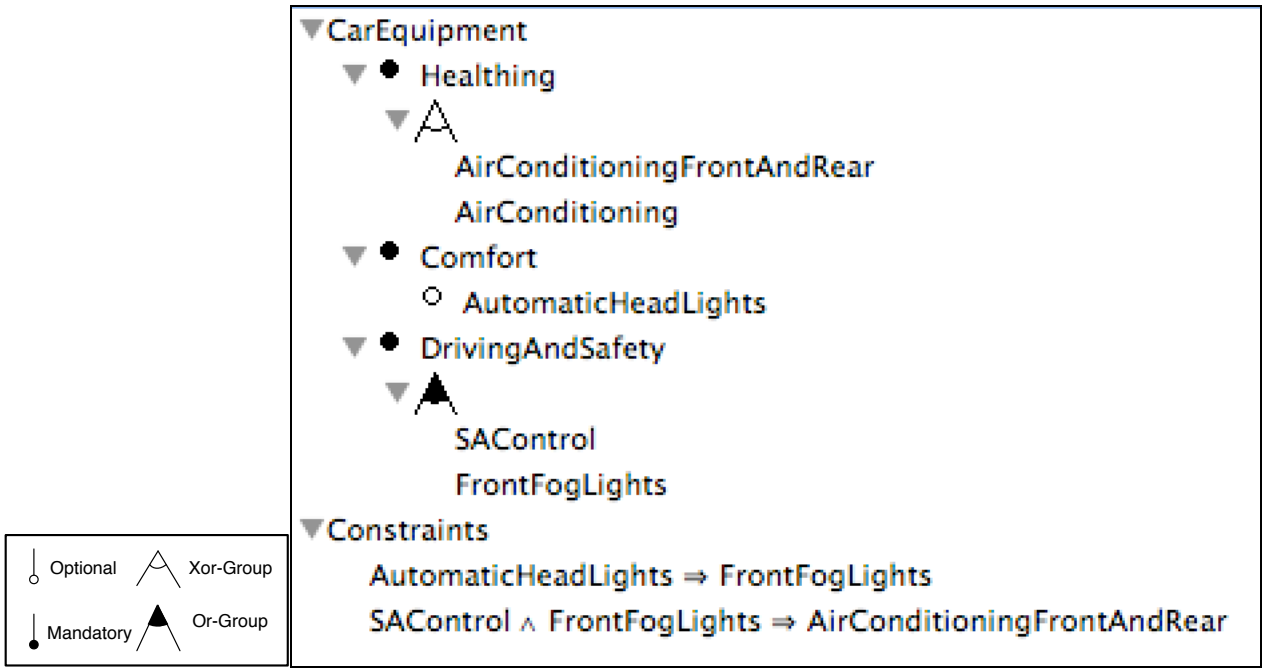


Figure 1.

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 »

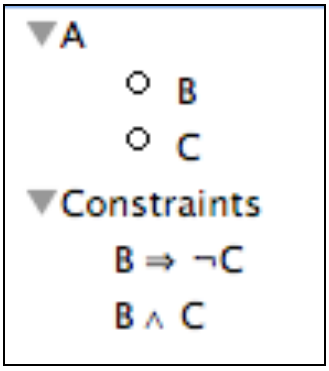


Figure 2

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.

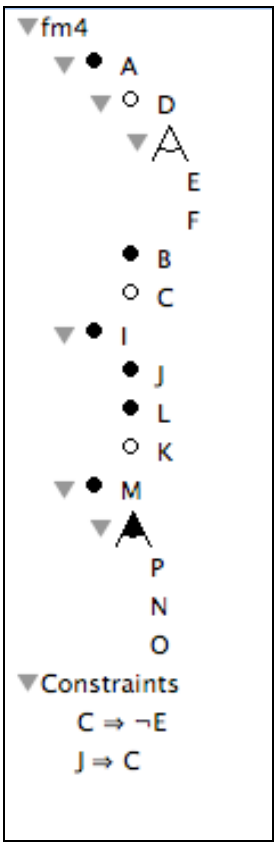


Figure 3

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.

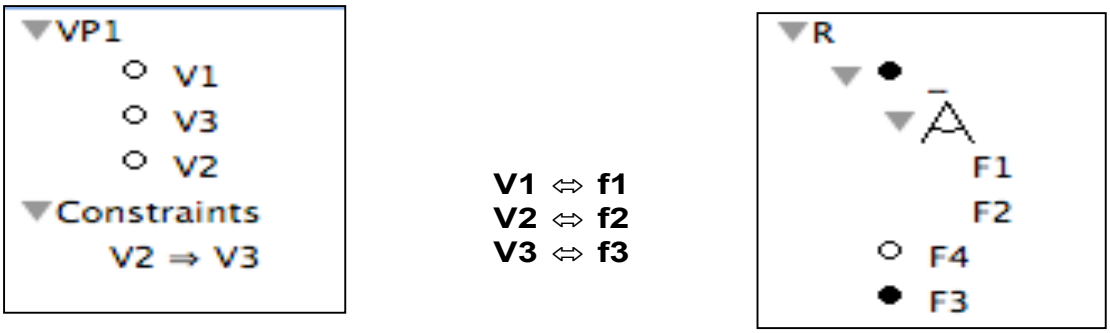


Figure 4

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.

Nous considérons les 2 matrices de comparaison ci-dessous (extraites de https://en.wikipedia.org/wiki/Comparison_of_video_converters)

Overview [edit]

Video converter	Developer	Licensing scheme	Supported platform			Website
			Windows	Mac OS X	Linux	
Any Video Converter	AVCLabs	Freeware	Yes	Yes	No	any-video-converter.com
Audials Tunebite 10 Platinum	Audials	Shareware	Yes	Yes	No	audials.com
Avidemux	Mean,Gruntster,Fahr	Free and open-source	Yes	Yes	Yes	www.avidemux.org
Dr. DivX	DivX, Inc.	Free and open-source	Yes	Yes	No	labs.divx.com/DrDivX
DVDVideoSoft Free Studio	DVDVideoSoft	Freeware (ad supported)	Yes	No	No	dvdvideosoft.com
FFmpeg	FFmpeg project	Free and open-source	Yes	Yes	Yes	ffmpeg.org
FormatFactory	Chen Jun Hao	Freeware (ad supported)	Yes	No	No	formatoz.com
Freemake Video Converter	Freemake	Freeware (ad supported)	Yes	No	No	freemake.com
HandBrake	Handbrake Project	Free and open-source	Yes	Yes	Yes	handbrake.fr
MediaCoder	Stanley Huang	Freeware (ad supported)	Yes	No	No	www.mediacoderhq.com
MEncoder	The MPlayer Project	Free and open-source	Yes	Yes	Yes	mplayerhq.hu
OggConvert	Tristan Brindle	Free and open-source	Experimental	No	Yes	oggconvert.tristanb.net
Prism Video Converter	NCH Software	Freeware (and paid version)	Yes	Yes	No	nchsoftware.com/prism/
SUPER	eRightSoft	Freeware (ad supported)	Yes	No	No	www.erightsoft.com
Transcode	Transcode Team	Free and open-source	No	No	Yes	transcoding.org
VirtualDub	Avery Lee	Free and open-source	Yes	No	No	virtualdubmod.sourceforge.net
XMedia Recode	Sebastian Dörfler	Freeware	Yes	No	No	www.xmedia-recode.de

Input [edit]

	Supported input container formats														
Video converter	3GP	AVI	Blu-ray video	DVD video	FLV	Matroska	MP4	MPEG-PS	Ogg	QuickTime	SVCD	TS	TOD	VCD	WMV
Any Video Converter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Audials Tunebite 10 Platinum	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Avidemux	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FFmpeg	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Freemake Video Converter	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
FormatFactory	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DVDVideoSoft Free Studio	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
HandBrake	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SUPER[1]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Prism Video Converter	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes
XMedia Recode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Question #13 : L'objectif est de produire un feature model qui correspond à ces deux matrices de produit (donc on veut un seul feature model). A toute configuration valide de ce feature model correspondra au moins un produit. Les « features » seront donc issues des les 2 matrices.
Avant d'élaborer le feature model, il faut s'interroger (1) sur la pertinence de certaines caractéristiques (e.g., Website) ; (2) la manière de structurer l'information sous forme de hiérarchie. Attention également car certains produits ne sont que dans une matrice (en première approximation, on s'intéressera uniquement aux produits qui sont dans les deux matrices)

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

Question #14 : Etudier la documentation et localiser la variabilité de ffmpeg (à la compilation ou à l'exécution): e.g., https://www.ffmpeg.org/ffmpeg.html#Main-options

Question #15 : Etudier le code source de ffmeg (http://git.videolan.org/?p=ffmpeg.git;a=tree) et localiser la variabilité de ffmpeg

Question #16 : Choisir une option de ffmpeg dans la documentation et expliquer comment elle est parsée et prise en compte dans le code :
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 #17 : Produire le feature model d'un configurateur de votre choix