UFR 919 Informatique
Master Informatique – Spécialité STL
UE ILP

TME1 — Programmation en ILP1

Jacques Malenfant, Christian Queinnec

`$Revision: 1.21 $` — `$Date$`

1 Environnement de travail

Objectif : se donner un environnement de travail incorporant les outils et les programmes du cours.

Buts :

- Identifier et localiser les outils
- Installer son environnement de travail (à l'UPMC et chez soi)

Les liens

Les répertoires dans lesquels vous pourrez trouver informations et outils pour le cours ILP se trouvent en :

```
file:///Infos/lmd/2008/master/ue/ilp-2008oct/
http://www-master.ufr-info-p6.jussieu.fr/2008/Ext/queinnec/ILP/
```

Les sources du système ILP se trouvent sous :

```
file:///Infos/lmd/2008/master/ue/ilp-2008oct/
```

Par ailleurs, indiquez, si besoin, dans votre navigateur que votre proxy est `proxy.ufr-info-p6.jussieu.fr` port 3128.

1.1 Travail à réaliser :

Lancez Eclipse avec la commande `eclipse`. Cette commande lance la dernière version (3.4) d'Eclipse ; en revanche, le Java par défaut à l'ARI est de version 1.5 alors que 1.6 est nécessaire pour ILP. Java 1.6 est présent en `/usr/java/jdk1.6.0_10`. Pour vous épargner des soucis, mettez à jour votre environnement avec :

```
export JAVA_HOME=/usr/java/jdk1.6.0_10
PATH=$JAVA_HOME/bin:$PATH
```

Eclipse peut être trouvé sur son site en `http ://www.eclipse.org/`.

Une fois lancé, Eclipse veut savoir où ranger son espace de travail (pour *workspace*), dites oui à ce qu'il vous propose. C'est, par défaut, `~/workspace/`, ne changez rien ! Cochez aussi la case lui demandant de mémoriser cette réponse afin qu'il ne vous la pose plus. L'installation à l'ARI semble ne pas poser la question mais le résultat est le même : votre espace de travail est bien le répertoire *workspace*.

Ajustez Eclipse de la manière suivante :

1. Menu *Window*, *Preferences*, *Java*, *Compiler*, réglez *Compliance level* à 1.6.
2. Menu *Window*, *Preferences*, *General*, *Editors*, *Text Editors*, cochez la case *Show Print Margin* qui montre la marge droite située à 80 colonnes : ne dépassez pas cette colonne, nous risquons de ne pas corriger vos programmes si leurs lignes dépassent cette limite.
3. Menu *Window*, *Preferences*, *General*, *Editors*, *Text Editors*, cochez la case *insert spaces for tabs*
4. Menu *Window*, *Preferences*, *General*, *Workspace*, *Text File Encoding* : utf8.
5. Menu *Window*, *Preferences*, *General*, *Workspace*, *text file line delimiter* : unix.
6. (facultatif) : Menu *Window*, *Preferences*, *General*, *Editors*, *Keys*, *Scheme :* emacs.
7. (facultatif) : Menu *Window*, *Preferences*, *General*, *Editors*, *Text editors*, *Spelling*, *User defined dictionary* : vous pourrez indiquer le fichier *liste.de.mots.francais.frgut.txt* qui viendra avec les sources d'ILP.

1.1.1 Installation sources ILP

Pour installer les sources comme un projet dans Eclipse, voici une procédure :

1. Dans l'écran initial d'Eclipse, cliquez sur le cercle *workbench* (tout à droite)
2. dans un shell (vous avez bien lancé Eclipse en arrière plan ?), faites ce qui suit (cela va créer quelques répertoires dont Java, Grammars et C :
   ```
   % cd ~/workspace/
   % tar xzf $( ls -t /Infos/lmd/2008/master/ue/ilp-2008oct/TGZ/ILP* | head -1 )
   ```
3. Créez un nouveau projet de type Java (*File*, *New java project*) ; dans la case nom du projet, tapez ILP1. Au moment où vous finissez de taper 1, Eclipse détecte qu'un tel répertoire existe et vous propose de le récupérer tel quel : cliquez *Finish*. Si ce n'était pas le cas, cliquez sur *create new project from sources* et choisissez le répertoire `workspace/ILP1`, enfin, cliquez *Finish*.
4. Le projet est presque prêt.

Voici ce que l'on peut voir dans l'explorateur de paquetages :

– `Java/src` : les sources d'ILP en Java.
– `*.jar` : les archives des outils utilisés dans le projet.
– `JRE System Library [JVM 1.6]` : l'exécutif Java utilisé par ce projet.
– `C` : des bibliothèques pour la compilation d'ILP vers C.

– `Grammars` : les grammaires des différentes versions ILP* et des programmes dans le sous-répertoire `Sample`.
– `Java` : le répertoire contenant `Java/src`, d'autres utilitaires et la documentation Javadoc du projet.
– et quelques autres fichiers ou répertoires comme le fichier `build.xml` qui est le fichier Ant de construction des binaires et autre produits du projet ou le fichier `LISEZ.MOI` dont le nom indique l'usage qu'il faut en faire (il contient notamment la documentation d'emploi du greffon).

Pour tester un peu l'installation, effectuez ceci :

1. dépliez le projet et clic droit (menu contextuel) sur le fichier `build.xml` pour sélectionner *Run as Ant build* (Prenez bien *Ant build* sans les trois petits points).
2. Pour tester un peu plus, dépliez `Java/src` puis le paquetage `fr.upmc.ilp.ilp1`, clic droit sur le fichier `WholeTestSuite`, *Run as JUnit test*. Normalement, tous les tests doivent passer (ce que montre une barre totalement verte).

Eclipse cherche normalement à compiler tous les sources du projet automatiquement, et donc lors de la création d'un projet à partir de sources. la fenêtre du bas sert à produire les messages de compilation. Si cette fenêtre fait apparaître des erreurs, il se peut que certaines configurations du projet ne soient pas correctes.

Pensez à étudier le tutoriel d'Eclipse intégré dans l'aide en ligne d'Eclipse.

1.1.2 Installation greffon ILP

Pour installer le greffon (pour *plugin*) propre à ILP, il faut effectuer les démarches suivantes.

1. Menu *Help*, item *Software Update*, sous-item *Find and Install*. Cliquez *Search for new features* puis le bouton *Next*.
2. Créer un nouveau site distant, indiquez ILP comme nom et, comme URL,
   `http://www-master.ufr-info-p6.jussieu.fr/2008/Ext/queinnec/ILP/`
3. Cliquez sur le bouton OK, puis sur l'écran précédent où ILP apparaît maintenant, cliquez sur le bouton *Finish*. Eclipse cherche alors le greffon sur le site distant et vous montre ce qu'il contient avec les numéros de version, etc.
4. Cochez ILP pour demander son installation puis cliquez sur le bouton *Next*
5. Acceptez la licence d'utilisation et passez, avec le bouton *Next*, à l'écran suivant.
6. Indiquez que vous voulez sauver les greffons dans un répertoire avec le bouton *Change Location* qui ouvre une sous-fenêtre dans laquelle vous pourrez ajouter un nouvel endroit (*Add location*) que vous choisirez être le répertoire `workspace`.
7. Quelque *Valider*, *OK* et *Finish* plus tard, le greffon s'installe. Il faut juste confirmer que je ne l'ai pas signé et que vous me faites confiance en cliquant sur *Install all*.
8. Eclipse se relance alors et le greffon est normalement installé.

À propos, si vous souhaitez installer d'autres greffons dans Eclipse, vous aurez besoin d'indiquer à Eclipse d'utiliser un relais (pour *proxy*) pour accéder à Internet. C'est dans les préférences d'Eclipse, menu *Window*, *Preferences...*, item *install/update*, *enable proxy connection*. Indiquez alors les coordonnées du proxy conseillé par l'ARI, fort probablement `proxy.ufr-info-p6.jussieu.fr` port 3128.

2 Cycle d'exécution d'un programme ILP

Objectif : apprendre à réaliser toutes les étapes permettant d'exécuter un programme ILP et vérifier la bonne installation de l'environnement de travail.

Buts :

- Comprendre les exemples de programmes ILP1.
- Comprendre comment vérifier la syntaxe d'un programme en validant le document XML correspondant par rapport à sa grammaire.
- Comprendre comment exécuter un programme.

Les liens :

XML	`http://www.w3.org/XML/Core/`
RelaxNG	`http://www.oasis-open.org/committees/relax-ng/`

Documents sur RelaxNG :

Tutoriel	`http://www.oasis-open.org/committees/relax-ng/tutorial-20011203.html`
Syntaxe compacte	`http://www.oasis-open.org/committees/relax-ng/compact-20021121.html`
Livre	`http://books.xmlschemata.org/relaxng/`

Outils spécifiques :

Jing	`http://www.thaiopensource.com/`

2.1 Comment valider un document avec Jing

Jing est le validateur de document XML par rapport aux schémas RelaNG que nous utilisons pour définir nos grammaires de langages. Voici les principales étapes permettant de le mettre en œuvre :

- On peut, dans Eclipse, grâce au greffon ILP, utiliser le menu contextuel sur un fichier XML, menu *ILP* puis *Validate*. Les résultats apparaissent dans l'onglet console (dans la sous-console intitulée *Jing Output*.
- Un greffon spécialisé en XML procure la possibilité de vérifier si un fichier est bien-formé. C'est l'entrée *Validate* du menu contextuel des fichiers XML. La validation par le greffon ILP vérifie en plus la conformité à une grammaire ILP.
- en termes de commande, pour valider un fichier XML par rapport à une grammaire RelaxNG, il faut faire (après avoir défini les variables d'environnement précédentes) :

```
java -jar ${JING} <schema>.rng <fichier>.xml
```

2.2 Comment exécuter un programme ILP1

Pour exécuter un programme ILP1 avec l'interprète, nous vous fournissons une classe appelée `EASTFileTest.java` disponible dans le paquetage `fr.upmc.ilp.ilp1.eval`. Pour exécuter l'échantillon des programmes ILP1 du répertoire `Grammars/Samples`, il faut cliquez sur le nom de la classe (`EASTFileTest.java`) avec le bouton droit et sélectionner `Run as` puis `JUnit Test`. L'exécution de cette classe de test produit une trace dans la fenêtre du bas où vous pourrez voir le nom de chaque programme exécuté, le résultat et les impressions produites ainsi que ce qui était attendu.

Les programmes ILP1 du répertoire `Samples` ont tous un nom de la forme `u<d>+-1.xml` où `<d>+` représente une suite de chiffres. La manière la plus simple de faire exécuter une programme ILP1 consiste donc à l'écrire dans un fichier d'un nom de cette forme du répertoire `Samples` et de relancer les tests de la classe `EASTFileTest.java`.

Nous passons effectivement par des tests unitaires écrits avec l'outil JUnit pour exécuter les programmes. Les liens vous permettront de vous documenter sur JUnit (versions 3 et 4) :

JUnit `http://www.junit.org/`
JUnit `http://junit.sourceforge.net/`

Une meilleure description du traitement d'un programme se trouve dans la classe `fr.upmc.ilp.ilp1.Process`. L'ensemble des tests d'ILP1 est déclenchable avec la classe `fr.upmc.ilp.ilp1.WholeTestSuite`. Il est d'ailleurs bon de lire les classes de tests car elles montrent comment synthétiser, lire, plus généralement traiter des programmes ILP.

La classe `EASTFileTest.java` contient également un point d'entrée permettant de la considérer comme une application Java entière. On peut lancer une application Java avec le menu contextuel *Run as*, puis *Java application*. Il faut toutefois, avant de faire cela, paramétrer le lanceur d'application avec *Run as*, puis *Open Run Dialog...* afin d'indiquer que l'argument de l'application est le fichier que vous voulez tester individuellement.

2.3 Travail à réaliser :

– Exécuter les tests d'ILP1 et comparer les résultats obtenus avec ceux que vous attendiez en lisant chacun des programmes exécutés.

3 Programmer en ILP1

Objectif : comprendre toutes les étapes permettant d'écrire, de mettre au point et d'exécuter un programme ILP1.

Buts :

– Écrire un programme ILP1.
– Le mettre au point syntaxiquement par validation.
– Le mettre au point sémantiquement par son exécution.

3.1 Édition avec Eclipse d'un document XML

Créer un fichier XML et ouvrez-le avec le menu contextuel avec un éditeur de texte ou un éditeur structuré.

3.2 Édition sous Emacs documents XML avec nxml-mode et de schémas avec rnc-mode

Les liens :

nxml-mode `http://www.thaiopensource.com/nxml-mode/`
rnc-mode `http://www.pantor.com/`

3.2.1 Mise en place du mode nxml

Pour utiliser nXML sous emacs, placez d'abord les lignes suivants dans votre fichier `.emacs` (attention, si vous copiez-collez ce qui suit depuis le document PDF vers un éditeur de textes, il se peut que vous ayiez à convertir les apostrophes en des simples guillemets (une petite barre verticale)) :

```
(setq load-path
      (append load-path
              '("/Infos/lmd/2008/master/ue/ilp-2008oct/ELISP/nxml-mode-20041004/")))

(load "/Infos/lmd/2008/master/ue/ilp-2008oct/ELISP/nxml-mode-20041004/rng-auto.el")

(setq auto-mode-alist
      (cons '("\\.\\(xml\\|xsl\\|rng\\|xhtml\\)$" . nxml-mode)
            auto-mode-alist))
```

Ceci fait, dès que vous lancerez Emacs sur un fichier de suffixe xml, xsl, rng ou xhtml, ce mode sera automatiquement activé.

3.2.2 Mise en place du mode rnc

Pour utiliser rnc-mode sous emacs, placez d'abord les lignes suivants dans votre fichier `.emacs` :

```
(autoload 'rnc-mode
          "/Infos/lmd/2008/master/ue/ilp-2008oct/ELISP/rnc-mode")
(setq auto-mode-alist
      (cons '("\\.rnc$" . rnc-mode) auto-mode-alist))
```

Ceci fait, dès que vous lancerez Emacs sur un fichier de suffixe rnc, ce mode sera automatiquement activé.

3.2.3 Utilisation des modes

Pour les documents XML, éditez ensuite un nouveau fichier avec l'extension `xml` puis jeter un coup d'œil à la liste des liaisons des clés (*key bindings*) du mode pour voir les principales commandes. Pour compléter, regarder sur le site donné ci-haut dans la partie "*Further information about nXML mode*".

Le mode nXML vous donne quelques outils pour faciliter l'écriture de fichiers XML comme la validation au fur et à mesure des documents et la complétion des noms de balise. Pour que la validation fonctionne, il faut associer un schéma au fichier en cours d'édition. Pour cela, on va dans le menu XML et on choisit "Set schema" puis "File..." et on sélectionne le fichier de grammaire en format rnc. Pour les programmes ILP1, il s'agit du fichier `grammar1.rnc` du répertoire `Grammars`. Quand la validation est activée, un message `nXML-valid` ou `nXML-invalid` apparaît dans la barre d'état d'emacs en bas de la fenêtre.

Les deux commandes les plus utiles sont Ctrl-enter pour compléter le nom d'une balise ou d'un attribut et Ctrl-c Ctrl-f pour générer la balise fermante de l'élément englobant ouvert à l'endroit où se situe le curseur (en fait, j'utilise plutôt `</` qui suffit à fermer un élément).

Pour les schémas RelaxNG (en syntaxe compacte), il suffit d'éditer un fichier avec l'extension `rng`. Le mode rnc est cependant très limité. Il fait une légère coloration syntaxique, introduit les crochets et les accolades électriques et permet de commenter et décommentr rapidement des parties de fichier. Faire un "List key bindings" pour en savoir (un peu) plus.

3.3 Travail à réaliser

Exercices du TD1 Reprendre les exercices du TD1, les programmer en ILP1 avec emacs et les exécuter sous Eclipse.

Exercice. Calcul du discriminant. Écrire un programme ILP1 qui calcule le discriminant d'une équation du second degré étant donnée les coefficients a, b, c. Votre programme doit d'abord lier les variables a, b et c aux valeurs choisies pour exécuter le calcul. Ensuite, le programme calcule le discriminant et doit retourner l'une des chaînes suivantes :

- `"discriminant négatif : aucune racine"`
- `"discriminant positif : deux racines"`
- `"discriminant nul : une seule racine"`

4 Annexes diverses

4.1 Commandes

Si vous souhaitez utiliser la ligne de commande, il peut être utile de définir quelques variables d'environnement.

- Mettre les archives des outils en Java dans le chemin des classes :
  Sous bash :

```
export BASE_ILP=$HOME/workspace/ILP1
export JING=$BASE_ILP/Java/jars/jing.jar ;
```

```
for arch in $BASE_ILP/Java/jars/*.jar ; do
    CLASSPATH=$arch:$CLASSPATH ;
done
export CLASSPATH
```

Sous tcsh :

```
setenv BASE_ILP $HOME/workspace/ILP1
setenv JING ${BASE_ILP}/ILP/Java/jars/jing.jar ;
foreach arch (${BASE_ILP}/Java/jars/*.jar)
    setenv CLASSPATH ${arch}:${CLASSPATH} ;
end
```

4.2 Initialisation automatique de l'interprète de commandes

Les initialisations de variables globales de l'interprète de commandes (*shell*) que vous venez de faire sont valables uniquement dans l'instance d'interprète de commandes dans lequel vous les avez exécutées. Si vous lancez un nouvel interprète de commandes (par exemple, en ouvrant une nouvelle fenêtre terminal), il faut refaire les opérations pour initiliser les variables de cet interprète (modulo les interprètes dépendants qui reçoivent les liaisons de variables exportées).

Pour éviter de refaire ces commandes et plutôt initialiser automatiquement toutes les instances d'interprète lancées, il faut utiliser le fichier d'initialisations `.bashrc` (`.tcshrc` si vous utilisez `tcsh`). Ce fichier doit se trouver dans votre répertoire de base ("*home directory*"). Vous pouvez donc y ajouter les lignes précédentes. Faites cependant attention, il y a probablement déjà un tel fichier qui y a été placé par défaut lors de la création de votre compte. Éditez donc ce fichier plutôt que d'en créer un autre en l'écrasant.

4.3 Chemin de classes explicite en Java

Le fait de mettre les archives Java dans le chemin des classes définis par l'interprète de commandes (*shell*) fait en sorte que lors de l'exécution de la plupart des programmes de la jdk (javac, java, ...) qui vont suivre, ces outils sauront retrouver les classes correspondantes pour les exécuter. Une alternative consiste à passer explicitement ce chemin de classes à Java lors de son appel avec l'option `-cp` ou `-classpath`, sous la forme :

```
java -cp <chemin-de-classes> MaClasseAExecuter
```

Dans certains cas, les archives sont auto-exécutables, ce qui évite de préciser la classe à exécuter. L'inconvénient de cette approche est de devoir repréciser le chemin des classes à chaque appel à Java. On peut circonvenir cela en se définissant soi-même une variable pour contenir le chemin des classes et l'utiliser explicitement :

```
setenv ILPCP <chemin-des-classes-pour-ILP>
java -cp ${ILPCP}
```

Cela peut paraître équivalent à l'utilisation du chemin de classes standard (spécifié par la variable (globale) CLASSPATH), mais l'avantage est de pouvoir gérer indépendamment plusieurs chemins

des classes selon le projet Java sur lequel on travaille. Lorsqu'on fait beaucoup de Java, cela permet d'éviter les inévitables conflits entre les chemins de classes des différents projets, de même que réduire la taille du chemin de classes, ce qui accélère la recherche des classes par Java.