Université de Bretagne Occidentale

Master Informatique

2019/2020

Rapport de Projet

Compilation 2

Analyse d'un langage de gestion de scenarii domotiques (DOMUS) et production de code pour le simulateur SiDo

Auteur :
William PENSEC

Auteur :
Alan LE BOUDEC

13 décembre 2019

I Présentation

Dans cette archive, vous trouverez ce rapport PDF expliquant le travail réalisé entièrement ou partiellement et également des captures d'écrans des exécutions SiDo des fichiers de tests (simple, moyen et projet_domus). Vous y trouverez également une explication du code et de l'implémentation du projet.

II Travail réalisé

Analyse Lexicale

Ce point a été traité entièrement. Le fichier LEX est complet et fonctionne très bien avec le fichier CUP.

Analyse syntaxique

Cette partie fonctionne bien. Nous repérons les erreurs lexicales et syntaxiques mais nous ne détectons pas davantage d'erreur.

Production de code

Cette partie fonctionne quasiment entièrement sauf pour le scenario départ du fichier exemple_projet_domus car il y a l'alarme à allumer or elle ne s'allume pas lorsque l'on lance le scénario car il manque 2 lignes dans le HabitatSpecific mais impossible pour nous d'en déterminer la raison malgré énormément de test.

Analyse sémantique

Nous n'avons pas eu le temps de traiter cette partie du tout par manque de temps.

Génération de résumé

Cette partie a été faite et tout fonctionne en affichant à chaque fois les bons résultats.

III Choix d'implémentation

III.1 Fichier LEX

Nous avons choisi de mettre beaucoup de terminaux pour être le plus précis possible lors de l'analyse avec le fichier CUP. Cela nous a aidé à différencier beaucoup de cas.

III.2 Fichier CUP

Le fichier commence par la déclaration des symboles terminaux et non terminaux. Pour davantage de facilité, nous avons mis tout les symboles en String y compris les entiers car pour les enregistrer ensuite dans les Hashmap c'était plus simple plutot que de créer une HashMap pour des entiers exclusivement.

```
prog    ::= liste
            |  {: System.out.println("FINPROG "+parser.pline()+","+parser.pcolumn()); :} ;

com ::= COMMENTAIRE com
        | ;

liste ::= com DEBPROG  com blocappareil  com blocinterface  com blocscenarii  com bloccommande
```

Ici nous déclarons la lecture du bloc appareil :

```
//============================BLOC APPAREIL=======================
//================================================================

blocappareil ::= DEBAPP listeligne FINAPP ;

listeligne ::= ligne
             |ligne listeligne
             |error ;

ligne ::=      TYPE:t varAp:i POINT
             |AUTAPP PARO AUTTYPE:at PARF varAp:i POINT
             |DEF IDENT:id EQUL ACOO var:i ACOF POINT
             |COMMENTAIRE ;

varAp ::=   IDENT:i
            |IDENT:i VIRG varAp
            |error_virg ;

error_virg ::= IDENT:i varAp {: System.out.println(" Erreur ligne "+parser.pline()+" colonne "+parser.pcolumn()+" : => "

var ::=     IDENT:i
            |IDENT:i VIRG var
            |error_virg ;
```

Puis ensuite, nous lisons le bloc interface :

```
//============================BLOC INTERFACE =======================
//================================================================
blocinterface ::= DEBINTER listeinter FININTER ;

listeinter ::= inter
             | inter listeinter
             | error ;

inter ::= INTER:a varInt POINT
          | COMMENTAIRE ;


varInt ::=  IDENT:i
            |IDENT:i VIRG varInt
            |error_virg;
```

Puis les blocs scenarii décomposés en plusieurs parties notamment avec les pourtout et les si ... sinon ... :

```
//============================BLOC SCENARII SCENARIO==============
//================================================================

blocscenarii ::= DEBSCENAR listeblocscenario FINSCENAR ;

listeblocscenario ::=   blocscenario
                      | blocscenario listeblocscenario ;

blocscenario ::= SCENAD IDENT:i CHEVD listesuite SCENAF IDENT CHEVD
               | COMMENTAIRE
               | error ;

listesuite ::= suite
               | suite listesuite ;

suite ::=  IDENT:i POINT ACTION:a POINTV
    | for
    | message
    | if
    | EXEC IDENT:i POINTV
    | COMMENTAIRE ;
```

```
//==========================boucle pourtout et message==============
//==================================================================

for ::= POURTT IDENT:i DEUXP variable:v FAIRE listesuite FAIT POINTV ;


variable ::= TYPE:t
           |AUTAPP PARO AUTTYPE:at PARF
           |AUTAPP:a
           |IDENT:i ;

message ::= MESSD PARO listeinterieurmess PARF POINTV;

listeinterieurmess ::= interieurmess
                     |interieurmess  VIRG listeinterieurmess ;

interieurmess ::= CHAINE:c
                |IDENT:i
                |IDENT:i POINT ETAT
                |error ;


truc::= varif:i1 DEQUAL varif:i2;

varif ::= IDENT:i POINT ETAT
        |IDENT:i
        |ACTETAT:a ;

if ::= SI PARO truc  PARF ALORS  listesuite FSI POINTV
      |SI PARO truc PARF  ALORS listesuite sinon ;


sinon ::= SINON listesuite FSI POINTV ;
```

Et enfin, nous finissons avec le bloc commande :

```
//====================DECLARATION COMMANDE ========================

bloccommande ::= COMMO listelignecommande COMMF ;

listelignecommande ::= lignecommande
                     | lignecommande listelignecommande ;

lignecommande ::= ASSOC IDENT:id EQUL commande POINT
                | PROG IDENT:i EQUL suiteprog POINT
                | COMMENTAIRE ;

commande ::= IDENT:i
           | ACOO res ACOF ;

res ::= IDENT:i
      | IDENT:i VIRG res ;

suiteprog ::= date:d
            | date:d VIRG suiteprog
            | ACOO date:d VIRG suiteprog ACOF ;

date ::= PARO choix:c1 VIRG choix:c2 VIRG choix:c3 VIRG choix:c4 VIRG choix:c5 PARF

choix ::= ENTIER:nb
    | UND ;
```

III.3 Simulateur SiDO

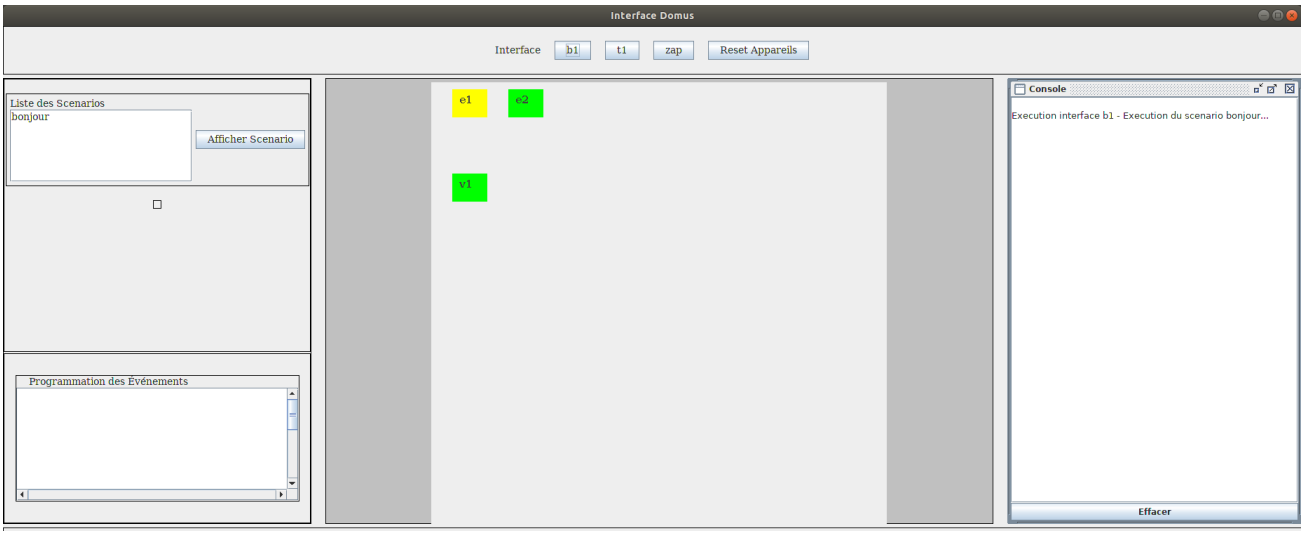

FIGURE 1 – Exemple mini

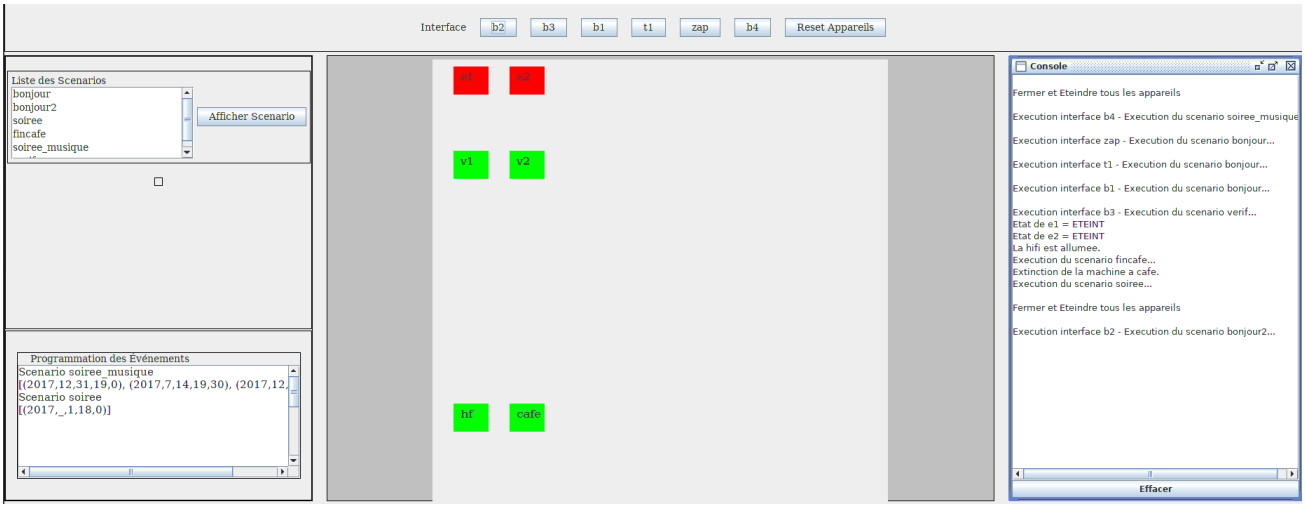

FIGURE 2 – Exemple simple

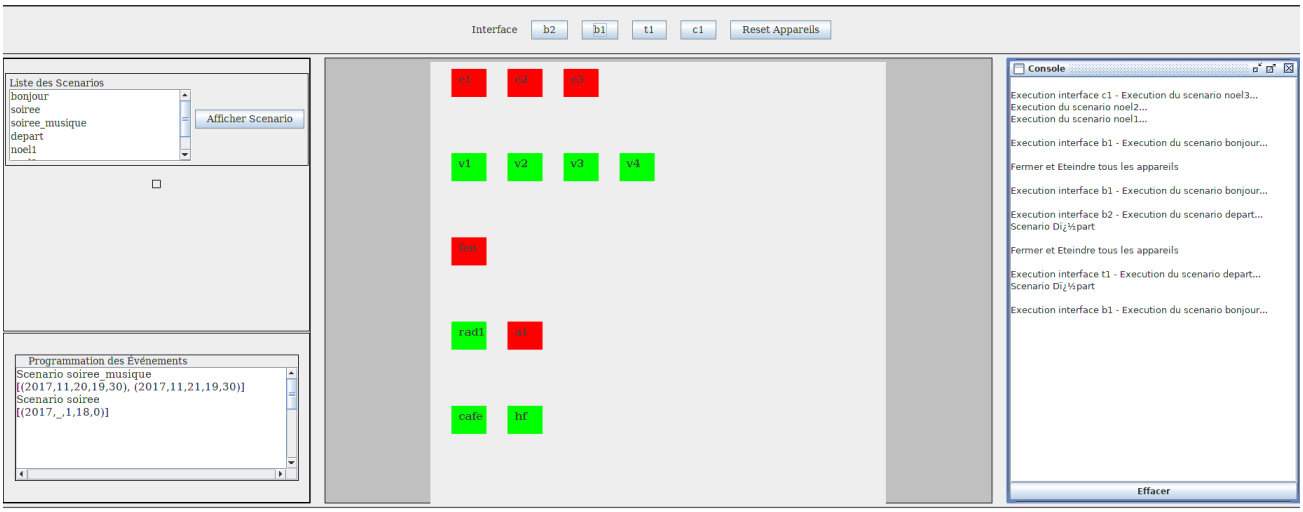

FIGURE 3 – Exemple projet_domus