Promela -> SPIN

Cours GLSD

Partie 1

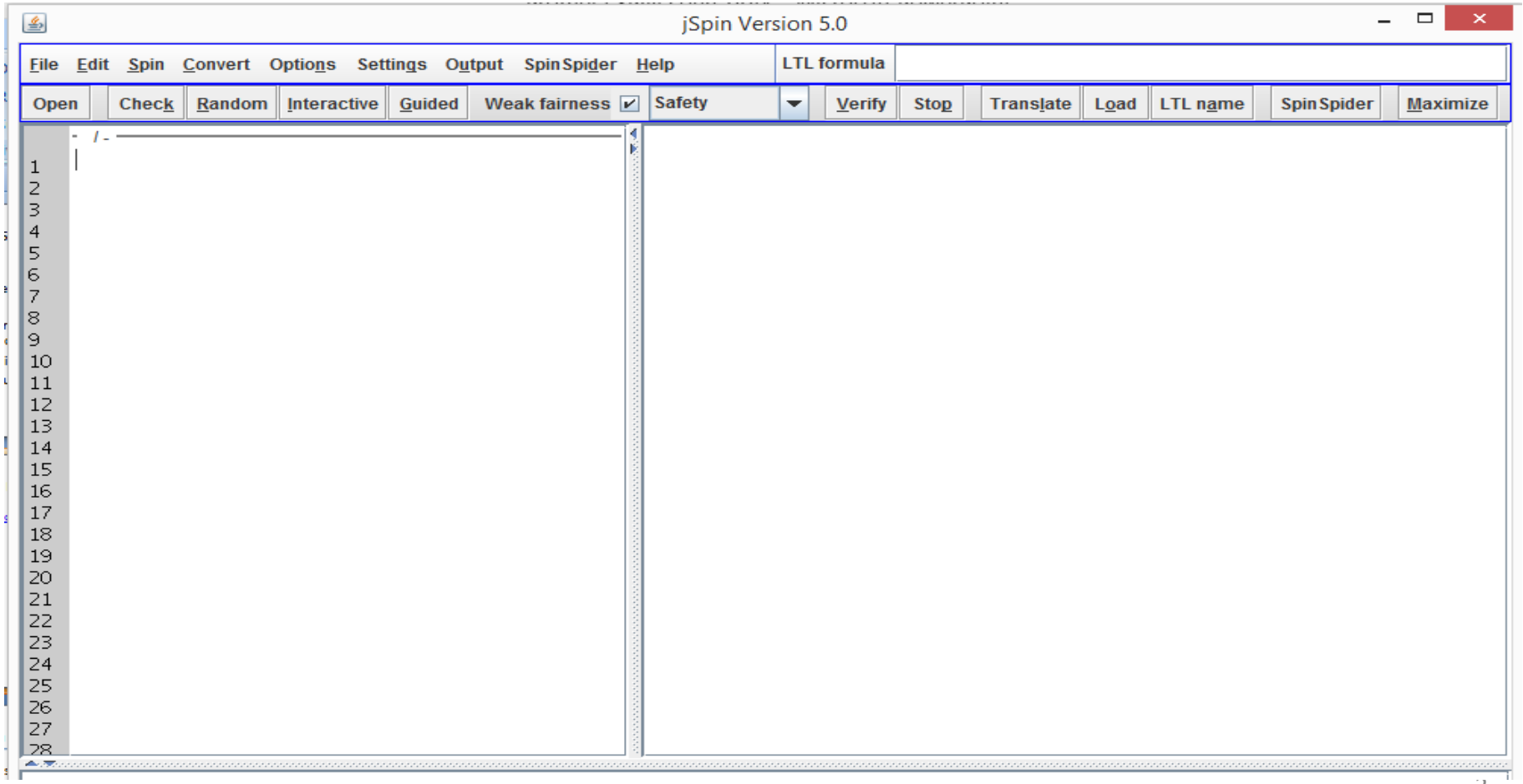
2015-2016

Plan du cours

- Introduction: Promela et SPIN
- Programme Promela: Types de donnée, Operations, Processus
- Processus : structures de contrôle utilisées, communication, synchronisation
- Vérification: insertion d'Assertions logiques
- SPIN: objectifs, options d'exécution de SPIN, exemple

Avant de commencer

- Télécharger le Jspin: http://code.google.com/p/jspin/



Télécharger

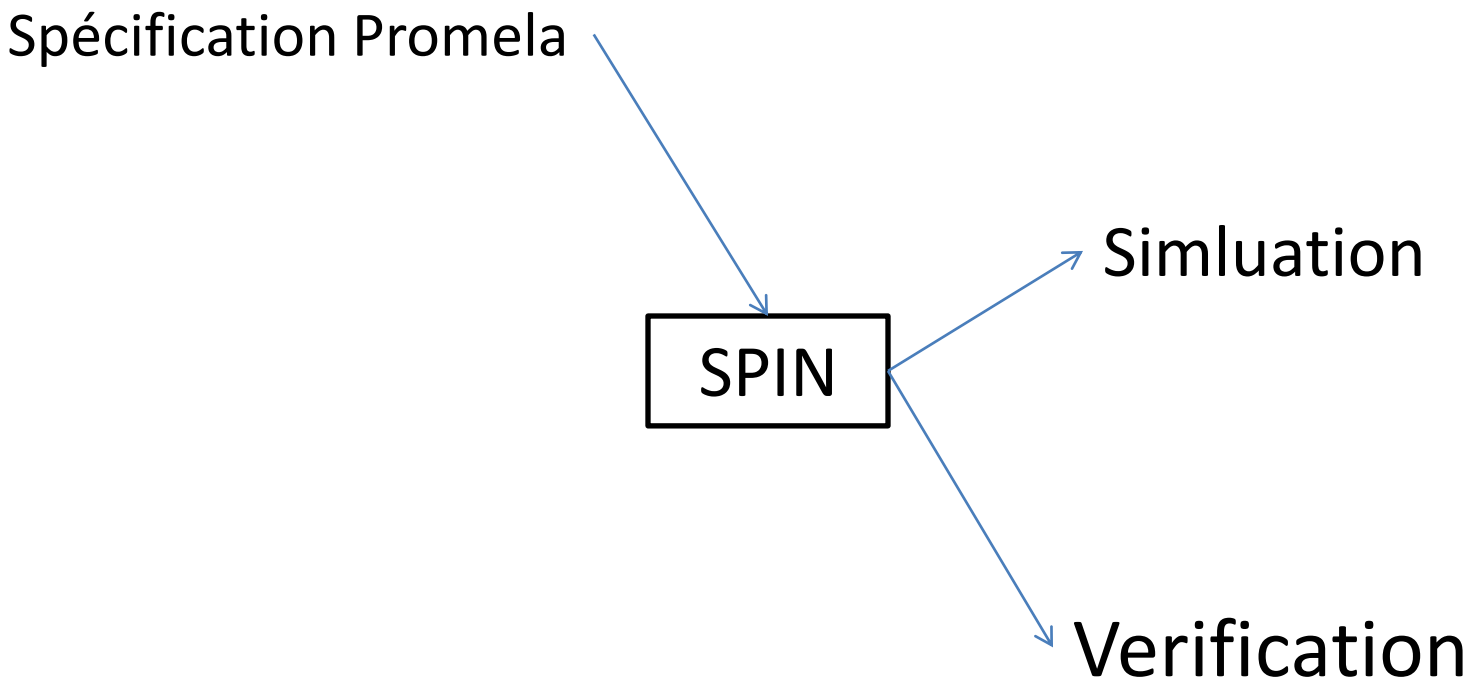
eui-1-7.zip	EUI zip file Type-Archive Deprecated	Jun 3, 2012	443.84KB
eui-1-7.exe	EUI Windows installer Type-Installer Deprecated	Jun 3, 2012	930.84KB
eui-quick.pdf	Quick start guide Type-Docs	Jun 3, 2012	143.99KB
eui-1-6.zip	EUI zip file Type-Archive Deprecated	Nov 8, 2011	457.23KB
eui-1-6.exe	EUI Windows installer Type-Installer Deprecated	Nov 8, 2011	955.59KB
jspin-bin-5-0a.zip	Binaries for Spin (6.0.1) and dot Type-Archive Deprecated	Dec 19, 2010	1.18MB
jspin-5-0.zip	jSpin for non-Windows or custom installation Type-Archive	Dec 15, 2010	530.8KB

Introduction

- **Promela**= **Pro**tocols (**pro**cesses) **me**ta-**la**nguage

- **SPIN**= **S**imple **P**romela **In**terpreter

Promela et SPIN



Promela

- Permet de **créer dynamiquement** des processus **concurrents**,
- Communication **Synchrone** ou **Asynchrone** entre processus,
- Un langage de spécification (inspiré **du langage C**),
- Avec des **types de données**

Programme Promela

- Un **ensemble de processus** qui s'exécutent en concurrence,

- Un programme Promela contient une suite de déclaration: **données**, **canaux**, **processus** ,

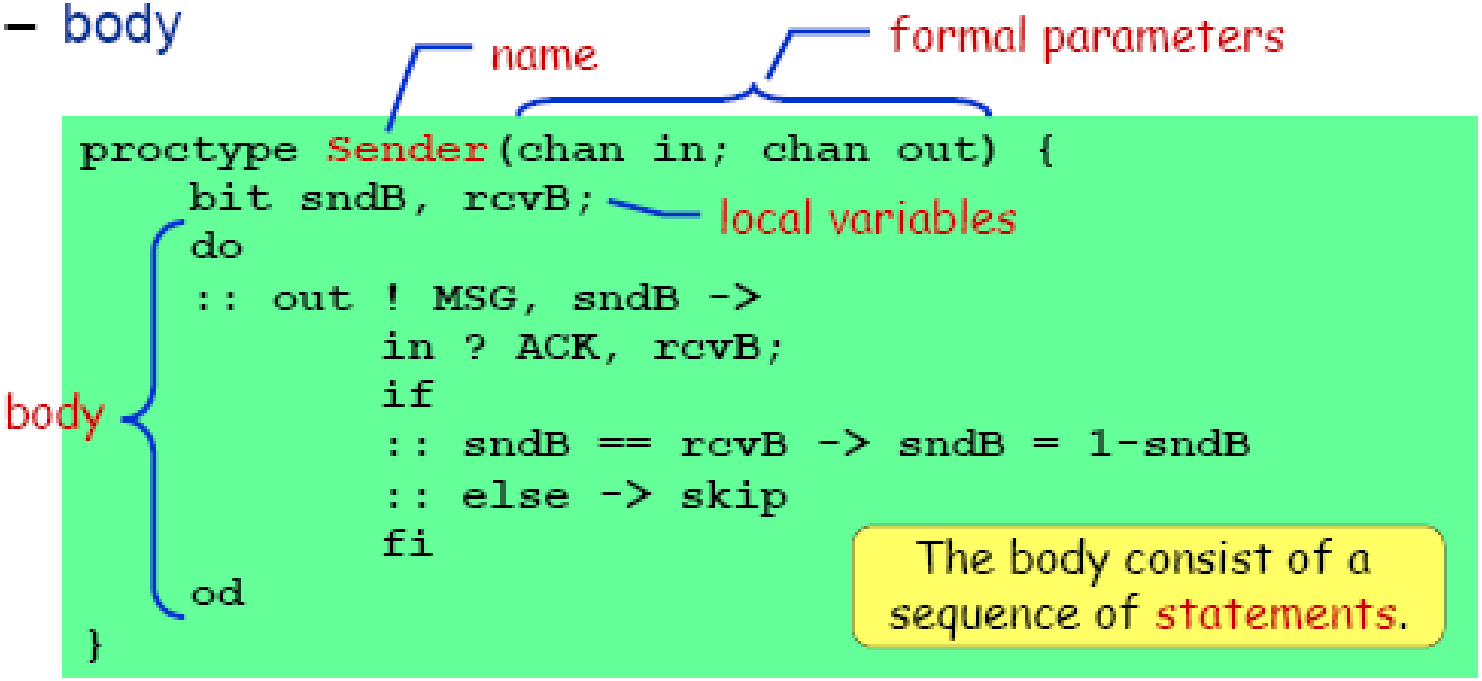
- Chaque processus doit être **défini**, ensuite **instancié** (créé) dans un programme principal: **init()**

Processus

- Chaque processus est **défini** avec le mot clé: **proctype**,
- Chaque processus **s'exécute en concurrence** avec les autres processus,
- Les processus **se communiquent** via des **variables partagées** et des **canaux**.
- Les processus seront créés soit dans un programme principal **init** avec le mot clé **run**, ou en précédant leur déclarations par le mot clé **active[nombre de copies]**

Processus (corps)

- body



Exemple (1)

- Exemple 1:

```
proctype A()
{   byte state;
    state = 3
}
```

- Exemple 2:

```
byte state = 2;
proctype A()
   {        (state == 1) -> state = 3
   }
proctype B()
   {        state = state - 1
   }
```

***Voir**: exp_p11 et exp_p11_2*

Processus (création)

```
proctype Foo(byte x) {
   ...
}

init {
   int pid2 = run Foo(2);
   run Foo(27);
}

active[3] proctype Bar() {
   ...
}
```

number of procs. (opt.)

parameters will be initialised to 0

Processus (Instanciation)

On peut faire la création de processus:

- soit dans un processus principal nommé **init**

- Exemples:

exemple 1: **init** { **skip** }

exemple 2: **init** {**run** A(); **run** B()}

exemple 3:

```
proctype A(byte state; short foo) {
                (state == 1) -> state = foo
                }
init {
        run A(1, 3)
        }
```

- Ou en utilisant le mot clé: **active** juste avant la déclaration d'un processus

***Voir**: exp_p13, exp_p13_2 sous SPIN*

Types de données

Basic types

```
bit    turn=1;        [0..1]
bool   flag;          [0..1]
byte   counter;       [0..255]
short  s;             [-2^16-1.. 2^16 -1]
int    msg;           [-2^32-1.. 2^32 -1]
```

Arrays

```
byte a[27];
bit  flags[4];
```

array indicing start at 0

Typedef (records)

```
typedef Record {
   short f1;
   byte  f2;
}
Record rr;
rr.f1 = ..
```

variable declaration

Opérations

```
int ii;
bit bb;

bb=1;
ii=2;

short s=-1;

typedef Foo {
   bit bb;
   int ii;
};
Foo f;
f.bb = 0;
f.ii = -2;

ii*s+27 == 23;
printf("value: %d", s*s);
```

assignment =

declaration + initialisation

equal test ==

Les opérateurs ==, !=

- Ces opérateurs en Promela servent comme mécanismes de **synchronisation**:

x!=1;

est équivalente à :

while (x==1) **wait**;

Le processus **reste bloqué** tant que cette condition (x!=1) n'est pas satisfaite.

- X==9->y=1; le processus **se bloque** jusqu'à avoir x=9 ensuite il exécute y=1

- ***Voir**: exp_p16*

Structures utilisées dans un processus

- Structure conditionnelle: *if*
- Structure conditionnelle: *if … else*
- Structure itérative: *do*
- Structure de choix multiple
- Structure de saut inconditionnelle: *goto*
- Séquence atomique: *atomic*

Sémantique de: if

```
if
:: choice1 -> stat1.1; stat1.2; stat1.3; …
:: choice2 -> stat2.1; stat2.2; stat2.3; …
:: …
:: choicen -> statn.1; statn.2; statn.3; …
fi;
```

• Si l'une des condition est satisfaite: SPIN **choisit** de **manière aléatoire** l'un des **cas**,

•Si aucune **condition n'est satisfaite**: le **processus est bloqué**,

•Le symbole **flèche ->** peut être remplacé par le **point virgule ;**

•Voir exp_p18 sous SPIN

Sémantique de: if…else

```
if
:: (n % 2 != 0)  -> n=1
:: (n >= 0)      -> n=n-2
:: (n % 3 == 0)  -> n=3
:: else          -> skip
fi
```

- Si aucune condition n'est satisfaite, le **else** est exécuté
- Le **skip** ici oblige le processus de **sortir** du bloc ***if***
- **Voir exp_p19 sous SPIN**

Sémantique de: *do*

```
do
:: choice_1 -> stat_1.1; stat_1.2; stat_1.3; …
:: choice_2 -> stat_2.1; stat_2.2; stat_2.3; …
:: …
:: choice_n -> stat_n.1; stat_n.2; stat_n.3; …
od;
```

- Même effet que *if*, mais **avec itération**

- **Voir exp_p20 sous SPIN**

Exemple (1)

```
mtype = { RED, YELLOW, GREEN } ;

active proctype TrafficLight() {

byte state = GREEN;

do
:: (state == GREEN) -> state = YELLOW;
:: (state == YELLOW) -> state = RED;
:: (state == RED) -> state = GREEN;
od;

}
```

Voir exp_p21

Exemple (2)

```
active proctype Hello() {
    printf("Hello process, my pid is: %d\n", _pid);
}

init {
    int lastpid;
    printf("init process, my pid is: %d\n", _pid);
    lastpid = run Hello();
    printf("last pid was: %d\n", lastpid);
}
```

Exemple (3)

```
byte state = 1;
    proctype A()
    {       byte tmp;
            (state==1) -> tmp = state;
            tmp = tmp+1;
            state = tmp
    }

    proctype B()
    {       byte tmp;
            (state==1) -> tmp = state;
             tmp = tmp-1;
            state = tmp
    }

    init { run A(); run B() }
```

Exemple (4)

```
1.   #define true   1
2.   #define false  0
3.   #define Aturn  false
4.   #define Bturn  true
5.
6.   bool x, y, t;
7.   proctype A()
8.   {x = true;
9.   t = Bturn;
10.  (y == false || t == Aturn);
11.          /* critical section */
12.          x = false
13.  }
14.  proctype B()
15.  {y = true;
16.  t = Aturn;
17.  (x == false || t == Bturn);
18.          /* critical section */
19.          y = false
20.  }
21.
22.  init
23.  {       run A(); run B()
24.  }
```

structure de contrôle: GOTO

```
1.  proctype Euclid(int x, y)
2.      {
3.          do
4.          :: (x >  y) -> x = x - y
5.          :: (x <  y) -> y = y - x
6.          :: (x == y) -> goto done
7.          od;
8.      done:
9.          skip
10.     }
```

Séquence atomique

```
1.   byte state = 1;
2.
3.       proctype A()
4.       {           atomic {
5.                     (state==1) -> state = state+1
6.                   }
7.       }
8.
9.       proctype B()
10.      {           atomic {
11.                    (state==1) -> state = state-1
12.                  }
13.      }
14.
15.      init
16.      {           run A(); run B()
17.      }
```