Spin

- We will use (a version of) "Spin" model checking system

- Started in 1980 by Gerard Holzmann

- Awarded ACM's Software Systems Award in 2001

Spin

- Spin poses a learning curve

- Erigone is a simplification of Spin designed for teaching

- Erigone uses subset of Spin's *Promela* modeling language

- Promela = (PRO)cess (ME)ta (LA)nguage

- SPIN = (S)imple (P)romela (IN)terpreter

Promela Summary

FEATURE	C	PROMELA
integers	char, short, int, long	byte, short, int, unsigned
floats	float, double	NONE
boolean	use int	bool
strings	char, char*	NONE
arrays	yes	1D & limited
operators	many	mostly same
if	as usual	similar to Erlang
loops	while, for, do	do, similar to if
output	printf	printf
input	scanf	NONE
functions	yes	NO
pointers	yes	NO
enum	enum	mtype
comments	/* */ and //	/* */
cpp	full	1-line #define, #include

Why So Small?

Gerard Holzmann (Spin inventor):

"Why not use full C as the specification language for SPIN? The sobering answer is that we would quickly find that vitrually all properties of interest would become undecidable."

Moti Ben-Ari (Erigone inventor):

"a model with a handful of variables and two dozen statements can give rise to complex behavior that strains the model-checking abilities of SPIN. For that reason PROMELA does not include an extensive set of constructs for structuring the program and its data; in particular you will not find constructs like functions and classes"

If Syntax

- "Guarded commands" wrapped inside "`if ... fi`"

- Example:

```
disc = b*b - 4*a*c;
if
:: disc < 0 ->
   printf("no real roots\n")
:: disc == 0 ->
   printf("duplicate real roots\n")
:: disc > 0 ->
   printf("two real roots\n")
fi
```

- Notes:

  - Each guarded command starts with double colon
  - Guard precedes "->" arrow
  - Command(s) follows arrow

If Semantics

- First: evaluate all guards
- Then:
  - If no guard true: statement blocks until at least one guard becomes true (which could happen due to action of some concurrent process)
  - If one guard true: execute its command(s)
  - If more than one guard true: execute command(s) of randomly chosen guard

Else

- Guard consisting of "`else`" keyword is true if all other guards are false

- Example:

```
disc = b*b - 4*a*c;
if
:: disc < 0 ->
   printf("no real roots\n")
:: disc == 0 ->
   printf("duplicate real roots\n")
:: else ->
   printf("two real roots\n")
fi
```

Do Syntax

- Similar to `if` statement

- Example: compute GCD by repeated subtraction

```
/*  assume x and y are initialized  */
int a = x, b = y;
do
:: a > b   ->   a = a - b
:: b > a   ->   b = b - a
:: a == b  ->   break
od
printf("GCD(%d, %d) = %d\n", x, y, a);
```

- Notes:

  - No loop test; only way out is via `break`
  - Body consists of guarded commands
  - Some true guard is chosen at random
  - Block if no true guard

Do Semantics

- Promela has no other type of loop

- Most common loop has only 2 guarded commands:

```
do
:: [exit test] -> break
:: else        -> [body statements]
od
```

- This structure provides deterministic operation

Complete Program

```
proctype P() {
    int x = 15, y = 20;
    int a = x, b = y;

    do
    :: a > b   ->   a = a - b
    :: b > a   ->   b = b - a
    :: a == b  ->   break
    od
    printf("GCD(%d, %d) = %d\n", x, y, a);
}
```