OPL, un lenguaje de programación de restricciones.

Nacho Castiñeiras

Grupo de programación declarativa

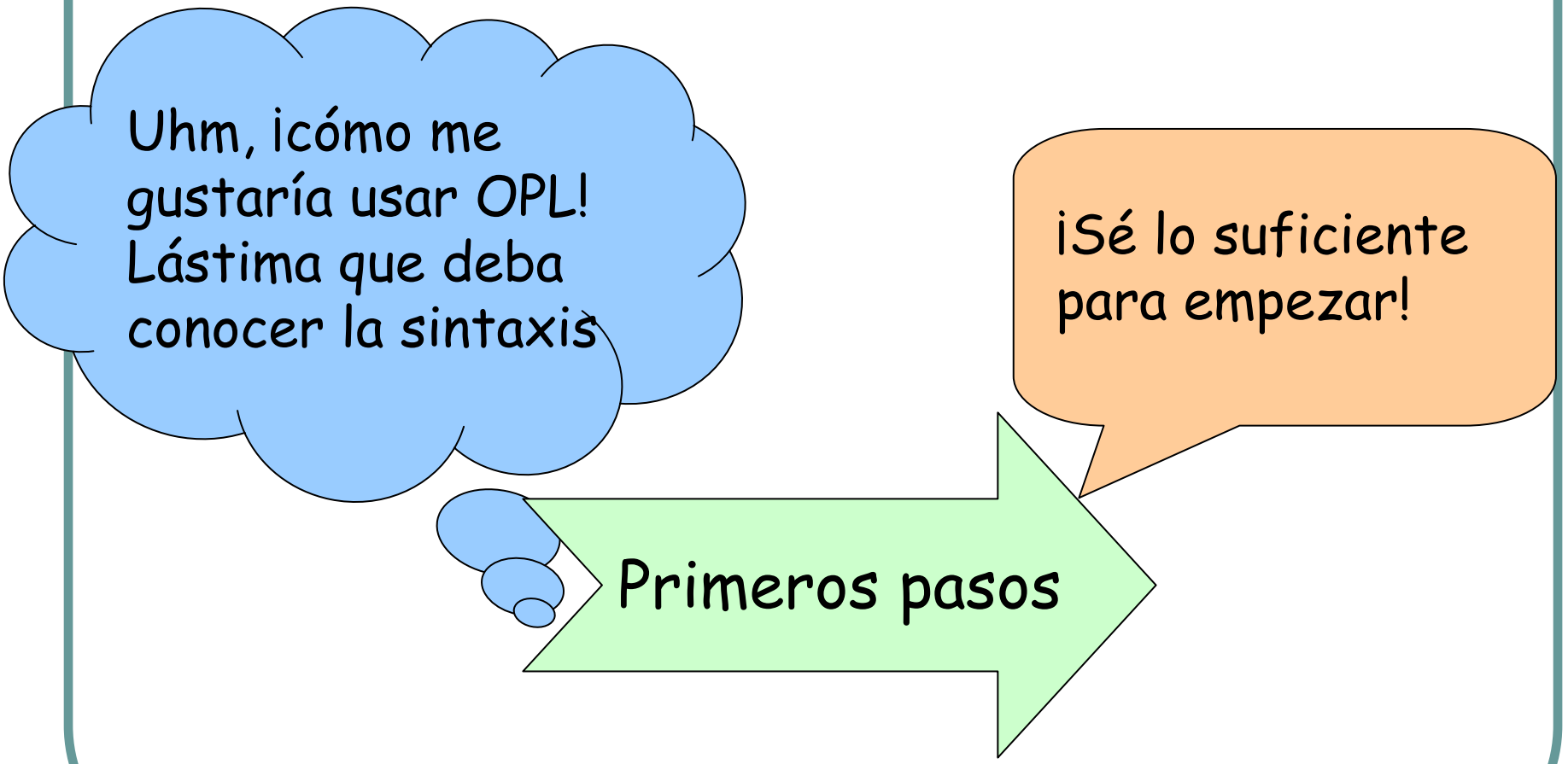
Índice

- Primeros pasos
- ILOG CPLEX
- ILOG SOLVER
- ILOG SOLVER SCHEDULER

Primeros pasos.

Comprender el entorno y empezar a trabajar en él.

Objetivo de esta fase previa.



Primeros pasos

- Motivación
- Mapa de componentes.
- Sintaxis mínima OPL

Motivación

¿Qué queremos?

Resolver problemas de :

- Satisfacción de restricciones.
- Optimización de restricciones.
  - Búsqueda del valor óptimo.

Motivación

¿Cómo resolverlos?

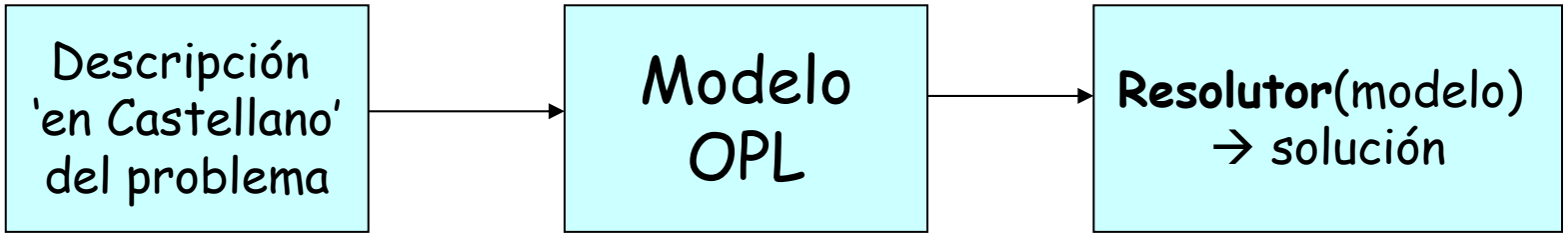
Ciclo de vida

Mapa de componentes

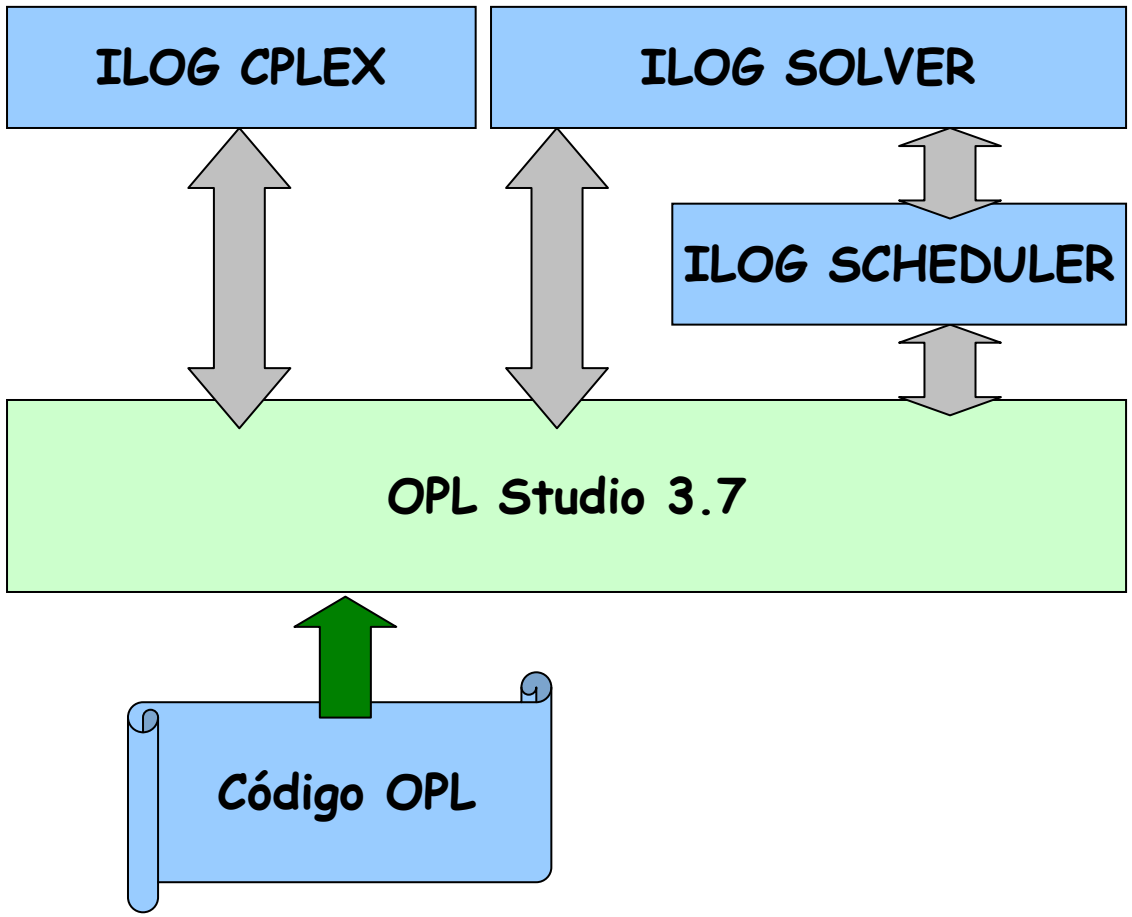
¿Qué nos ofrece ILOG?

- Resolutores especializados.
- Un lenguaje de modelado: OPL
  - Orientado a naturaleza de estos problemas.
  - Reduce salto modelo-algoritmo.

Mapa de componentes



Mapa de componentes



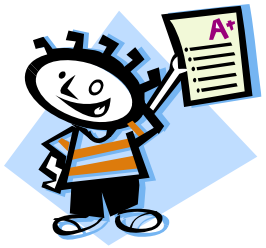
OPL Studio 3.7

IDE.

¿Cómo…

Abrir/Crear 

Modelar 

Resolver/Ver resultados 

?

Sintaxis mínima OPL

Estructura de un modelo OPL.

1. Datos
2. Variables
3. Función de óptimo
4. Restricciones
5. Procedimiento búsqueda

Sintaxis mínima OPL.

Datos

- Tipos básicos.

  int, float, enum

- Tipos estructurados.

  range, array, struct, set

Datos. Tipos básicos

int

- int numero1 = 2;
- int numero2 = numero1 * numero1 + 1;
- int numero3 = ...;
- int dimeTu << "Pasame un entero";

Datos. Tipos básicos

float

- float real = -3.2;
- float+ real_positivo = 4.27;

Datos. Tipos básicos.

enum

- enum Color { Azul, Grana, Naranja };
- enum Days { Lunes, Miercoles, Viernes, Domingo};

- //Color mi_color = Azul;

Datos. Tipos estructurados

range

- range coches 1..6;
- range numeros [2*numero1..2*numero2];

Datos. Tipos estructurados

arrays

Dos aspectos a estudiar

- Indexación
- Inicialización

Datos. Tipos estructurados

arrays

```
int mi_array[2..5] = [10, 20, 30, 40];
//int a1 = mi_array[3];

Color concesionario[coches] = [Azul,
Grana, Naranja, Naranja, Azul, Azul];

//int me_gusta[Days] = [0,3,6,3];
```

Datos. Tipos estructurados

arrays

```
int camisetas[i in 1..3] = i+1;

int feliz[Days,1..3] = [ [1, 2, 3], [4, 5, 6],
                         [7, 8, 9], [10, 11, 12] ];

int muy_elegante[Days, Color] = ...;
```

Datos. Tipos estructurados

- int muy_elegante[Days, Color] = ...;

```
//Instancia
muy_elegante = #[
            Lunes : #[
                    Azul : 4,
                    Naranja : 2,
                    Grana : 3
                    ]#,
            Miercoles : #[
                    Azul : 3,
                    Naranja : 3,
                    Grana : 4
                    ]#
... ]#;
```

Datos. Tipos estructurados

struct

- struct Conexion {
  int origen;
  int destino;
  };
  Conexion c = <2,3>;
  //int orig = c.origen;

Datos. Tipos estructurados

```
sets
{int} pares_hasta4 = {0,2,4};

{Conexion} conexiones = {<1,2>, <1,3>, <2,3>};

//Array indexado por los elementos de un
   conjunto. Estos elementos son estructuras.
   int valores[conexiones] = [3,4,5];
```

Sintaxis mínima OPL

Estructura de un modelo OPL.

1. Datos
2. Variables
3. Función de óptimo
4. Restricciones
5. Procedimiento búsqueda

Sintaxis mínima OPL

Variables de decisión

- int
- float
- enum

Variables de decisión

int

- var int entero1 in 0..5;
- var int nuevo_concesionario1[coches] in 0..1;

enum

- var Color nuevo_concesionario2[coches];

Variables de decisión

float

- var float x3;
- var float+ x4;
- var float+ real1[1..3];

Sintaxis mínima OPL

assert

Restricciones dependientes únicamente de datos de entrada.

```
assert
   num_reinas >= 3;
```

Sintaxis mínima OPL

Estructura de un modelo OPL.

1. Datos
2. Variables
3. Función de óptimo
4. Restricciones
5. Procedimiento búsqueda

Sintaxis mínima OPL

Función de óptimo

- minimize

  2*x3 + 3*x4

- maximize

  3*x3 + 2*x4

Sintaxis mínima OPL

Estructura de un modelo OPL.

1. Datos
2. Variables
3. Función de óptimo
4. Restricciones
5. Procedimiento búsqueda

Sintaxis mínima OPL

Restricciones

- Lineales
- No lineales

Restricciones

Lineales

$3*x3 + x4 <= 11;$

$-x3 + 2*x4 <= 5;$

Restricciones

No lineales

$x1 + x2 < 12;$

$entero1 <> 3;$

Sintaxis mínima OPL

Estructura de un modelo OPL.

1. Datos
2. Variables
3. Función de óptimo
4. Restricciones
5. Procedimiento búsqueda

Sintaxis mínima OPL

Procedimientos de búsqueda

```
search{
   tryall(j in 0..5 ordered by decreasing j)
        entero1 = j;
};
```

Sintaxis mínima OPL

Operadores.

Sobre grupo elementos. **sum, prod, max, min**

Relacionales. =, <>, <, <=, >, >=

Lógicos. \/, **&, not,** =>, <=>

Binarios. **+, -, *, /, mod**

Sobre conjuntos. **in, not in, union, diff, inter**

Sintaxis mínima OPL

Condiciones

- //filtrado de un conjunto

  {Conexion} nuevas_conexiones = { p | p in conexiones : p.origen + p.destino = 4};

- {int} modulos[i in 3..4] = { e | e in 1..10 : e mod i = 0 };

  modulos[3] = {3,6,9}

  modulos[4] = {4,8}

Sintaxis mínima OPL

```
forall
forall(i in 1..8){
   reinas[i] >= 1;
   reinas[i] <= 8;
}
sum
sum(i in 1..8) reinas[i] = (8*(8+1))/2
```

Sintaxis mínima OPL

Ideas extra sobre restricciones

- //Restricciones de orden superior

```
forall(i in Range)
  s[i] = sum(j in Range) (s[j] = i);
```

- //Array indexado con posición de otro array

```
forall(m in Men)
  husband[wife[m]] = m;
```

- //Restricciones globales

```
alldifferent(entero2);
```

Fin