Un procesador de expresiones epistémicas en programas lógicos

Javier Garea Cidre

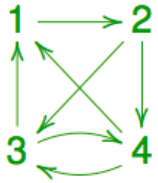
Director: Pedro Cabalar
Grado en Ingeniería Informática
Mención en Computación

Motivación

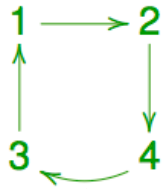
Programación lógica ...

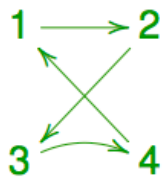
- **Answer Set Programming (ASP)**: paradigma Resolución de Problemas muy extendido en Representación del Conocimiento.
- Aplicaciones: configuración de productos, asignación y optimización de recursos y horarios, diagnóstico en *space shuttle*, composición musical, **planificación**, robótica, ...
- **Resolución de problemas** en ASP:
  - descripción problema = programa lógico (conjunto de reglas)
  - 1 solución al problema = 1 **answer set** (modelo del programa)
  - complejidad = $\mathbf{\Sigma}_2^P = \mathbf{NP}^{\mathbf{NP}}$

"Real world" (combinatorial) problem



solutions





ENCODING

DECODING

Problem instance (EDB)

```
vtx(1). vtx(2). vtx(3). vtx(4).
edge(1,2). edge(2,3). edge(2,4).
edge(3,1). edge(3,4). edge(4,3).
```

Problem specif. (KB)

```
{in(X,Y)} 1:- edge(X,Y).
:- in(X,Y), in(X,Z), Y!=Z.
:- in(X,Z), in(Y,Z), X!=Y.
reached(X) :- in(1,X).
reached(Y) :- reached(X), in(X,Y).
:- vtx(X), not reached(X).
```

```
$ clingo 0
mygraph.gph
hamilt.txt
```

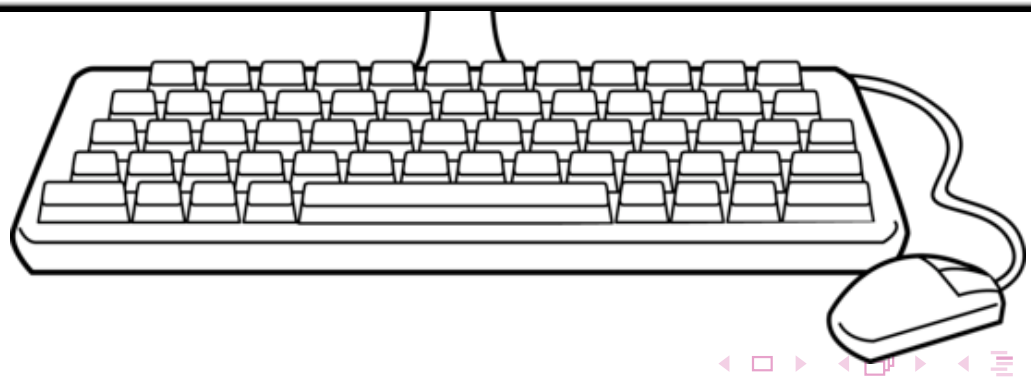
```
% Answer 1
in(4,3).
in(3,1).
in(2,4).
in(1,2).
```

```
% Answer 2
in(4,1).
in(3,4).
in(2,3).
in(1,2).
```

...

answer sets

ASP as a problem solving paradigm



Especificaciones epistémicas

- Algunos escenarios requieren ampliar ASP (investigación actual):
  - Operadores temporales (dominios dinámicos)
  - Funciones agregadas y restricciones numéricas
  - Optimización y preferencias
  - **Especificaciones epistémicas**

Especificaciones epistémicas

- Algunos escenarios requieren ampliar ASP (investigación actual):
  - Operadores temporales (dominios dinámicos)
  - Funciones agregadas y restricciones numéricas
  - Optimización y preferencias
  - **Especificaciones epistémicas**

- ASP + **operador modal** `&k`

  - la arista (X, Y) es crítica si está en **todas** las soluciones

```
critical(X,Y) :- edge(X,Y), &k{in(X,Y)}.
```

  - la arista (X, Y) es relevante si está en **alguna** solución

```
relevant(X,Y) :- edge(X,Y), not &k{not in(X,Y)}.
```

Especificaciones epistémicas

- Algunos escenarios requieren ampliar ASP (investigación actual):
  - Operadores temporales (dominios dinámicos)
  - Funciones agregadas y restricciones numéricas
  - Optimización y preferencias
  - **Especificaciones epistémicas**

- ASP + **operador modal** `&k`

  - la arista (X, Y) es crítica si está en **todas** las soluciones

```
critical(X,Y) :- edge(X,Y), &k{in(X,Y)}.
```

  - la arista (X, Y) es relevante si está en **alguna** solución

```
relevant(X,Y) :- edge(X,Y), not &k{not in(X,Y)}.
```

- Aplicaciones: **múltiples interpretaciones**, **información incompleta**.

- **Complejidad computacional**: aumenta a $\mathbf{\Sigma_3^P = NP^{\Sigma_2^P} = NP^{NP^{NP}}}$

- Pocas implementaciones (EP-ASP).

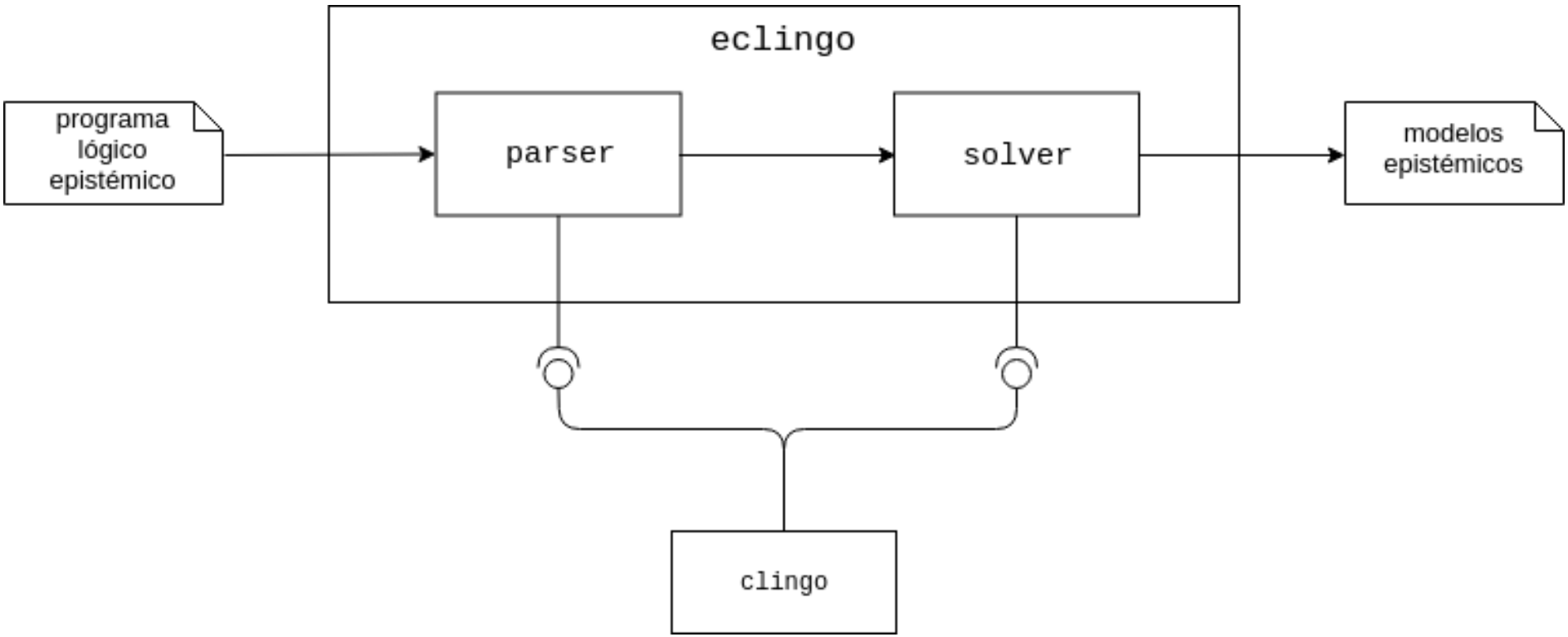
Índice

Demostración

Índice

eclingo

- Calcula **modelos** de un programa lógico epistémico.
- Escrito sobre el lenguaje de programación `Python` (versión 3.6.9).



Semántica G91

Definición no constructiva

1. Suponer un posible conjunto $\boldsymbol{S} = \{S_1, \ldots, S_n\}$ de posibles soluciones (answer sets).

Definición no constructiva

1. Suponer un posible conjunto $\boldsymbol{S} = \{S_1, \ldots, S_n\}$ de posibles soluciones (answer sets).
2. Simplificar $\&k\{p\}$ y $\&k\{not\ p\}$ en el programa usando $\boldsymbol{S}$:
   - $\&k\{p\} = true$, si $p \in \bigcap_{i=1}^{n} S_i$
     $\&k\{p\} = false$, en caso contrario

Definición no constructiva

1. Suponer un posible conjunto $\boldsymbol{S} = \{S_1, \ldots, S_n\}$ de posibles soluciones (answer sets).
2. Simplificar $\&k\{p\}$ y $\&k\{not\ p\}$ en el programa usando $\boldsymbol{S}$:
   - $\&k\{p\} = true$, si $p \in \bigcap_{i=1}^{n} S_i$
     $\&k\{p\} = false$, en caso contrario
   - $\&k\{not\ p\} = true$, si $p \notin \bigcup_{i=1}^{n} S_i$
     $\&k\{not\ p\} = false$, en caso contrario

   El programa resultante no contiene $\&k$'s.

Definición no constructiva

1. Suponer un posible conjunto $\boldsymbol{S} = \{S_1, \ldots, S_n\}$ de posibles soluciones (answer sets).

2. Simplificar $\&k\{p\}$ y $\&k\{not\ p\}$ en el programa usando $\boldsymbol{S}$:
   - $\&k\{p\} = true$, si $p \in \bigcap_{i=1}^{n} S_i$
     $\&k\{p\} = false$, en caso contrario
   - $\&k\{not\ p\} = true$, si $p \notin \bigcup_{i=1}^{n} S_i$
     $\&k\{not\ p\} = false$, en caso contrario

   El programa resultante no contiene $\&k$'s.

3. Comprobar si los *answer sets* del programa resultante son igual a $\boldsymbol{S}$.

Semántica G91

Problema

Usar la definición directamente no es factible para el cálculo.

- Si hay 10 átomos, tenemos 2^{10} posibles interpretaciones (no todas son soluciones) y $2^{(2^{10})}$ posibles candidatos **S**.

Semántica G91

Problema

Usar la definición directamente no es factible para el cálculo.

- Si hay 10 átomos, tenemos 2^{10} posibles interpretaciones (no todas son soluciones) y $2^{(2^{10})}$ posibles candidatos **S**.

Aproximación de `eclingo`

Variar cada $\&k\{p\}$ como un átomo auxiliar nuevo.

- Si tenemos 3, $\&k\{p\}$, $\&k\{q\}$, $\&k\{r\}$, tenemos 2^3 posibles combinaciones.

Semántica G91

Problema

Usar la definición directamente no es factible para el cálculo.

- Si hay 10 átomos, tenemos 2^{10} posibles interpretaciones (no todas son soluciones) y $2^{(2^{10})}$ posibles candidatos **S**.

Aproximación de `eclingo`

Variar cada $\&k\{p\}$ como un átomo auxiliar nuevo.

- Si tenemos 3, $\&k\{p\}$, $\&k\{q\}$, $\&k\{r\}$, tenemos 2^3 posibles combinaciones.

Π_{aux} = programa donde reemplazamos cada $\&k$ por un átomo auxiliar

Aproximación `eclingo`

Ejemplo:

1. Generamos candidato $\&k\{p\} = true$, $\&k\{q\} = false$, $\&k\{r\} = true$.
2. Calculamos los answer sets $\{S_1, \ldots, S_n\}$ de Π_{aux}.
3. Comprobamos: p cierto en todos, q falso en alguno, r cierto en todos.

Aproximación `eclingo`

Ejemplo:

1. Generamos candidato $\&k\{p\} = true$, $\&k\{q\} = false$, $\&k\{r\} = true$.
2. Calculamos los answer sets $\{S_1, \ldots, S_n\}$ de Π_{aux}.
3. Comprobamos: p cierto en todos, q falso en alguno, r cierto en todos.

Los pasos 1 y 2 los resuelve una llamada a `clingo` sobre Π_{aux}.

Aproximación `eclingo`

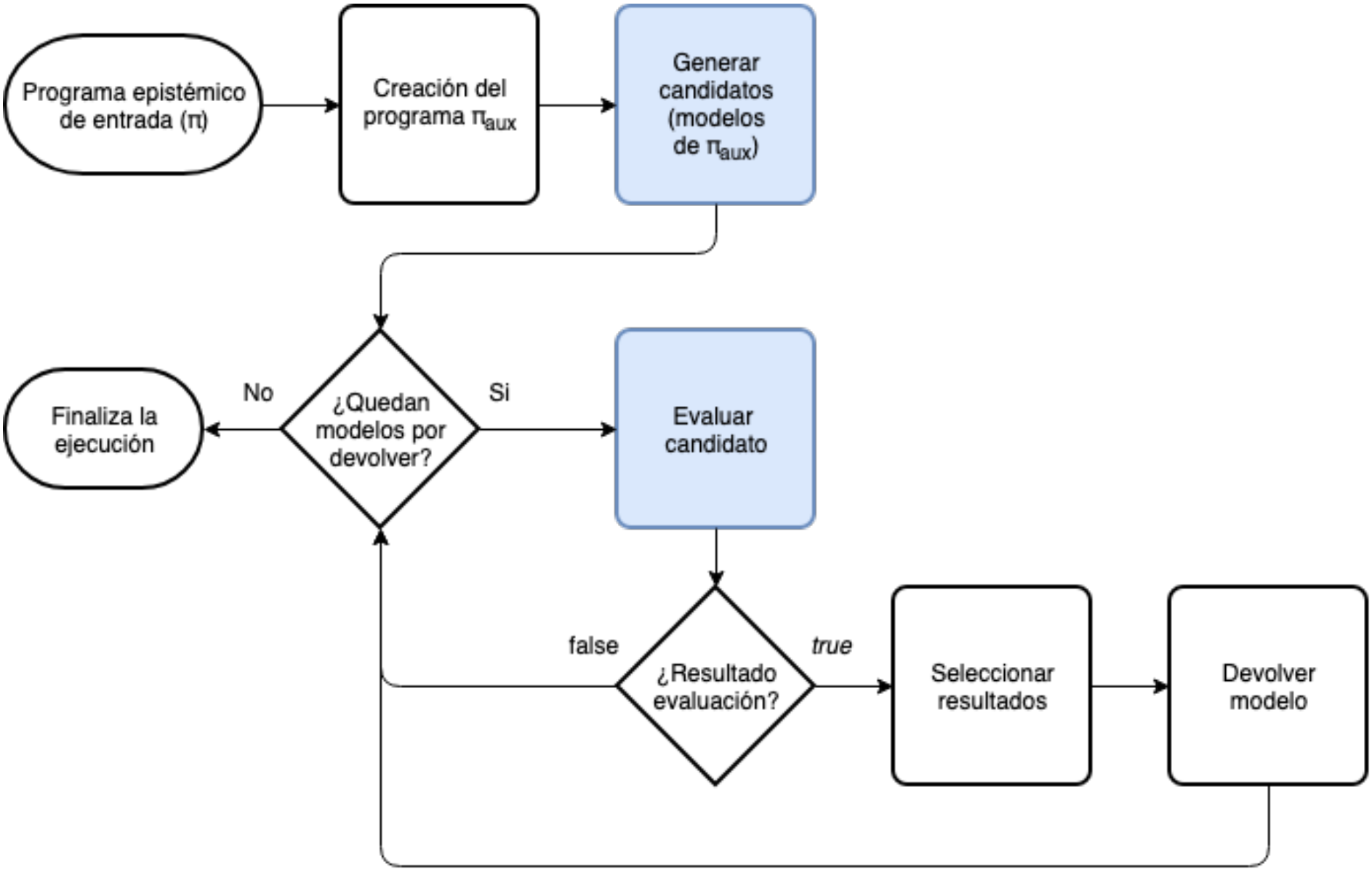
Ejemplo:

1. Generamos candidato $\&k\{p\} = true$, $\&k\{q\} = false$, $\&k\{r\} = true$.
2. Calculamos los answer sets $\{S_1, \ldots, S_n\}$ de Π_{aux}.
3. Comprobamos: p cierto en todos, q falso en alguno, r cierto en todos.

Los pasos 1 y 2 los resuelve una llamada a `clingo` sobre Π_{aux}.

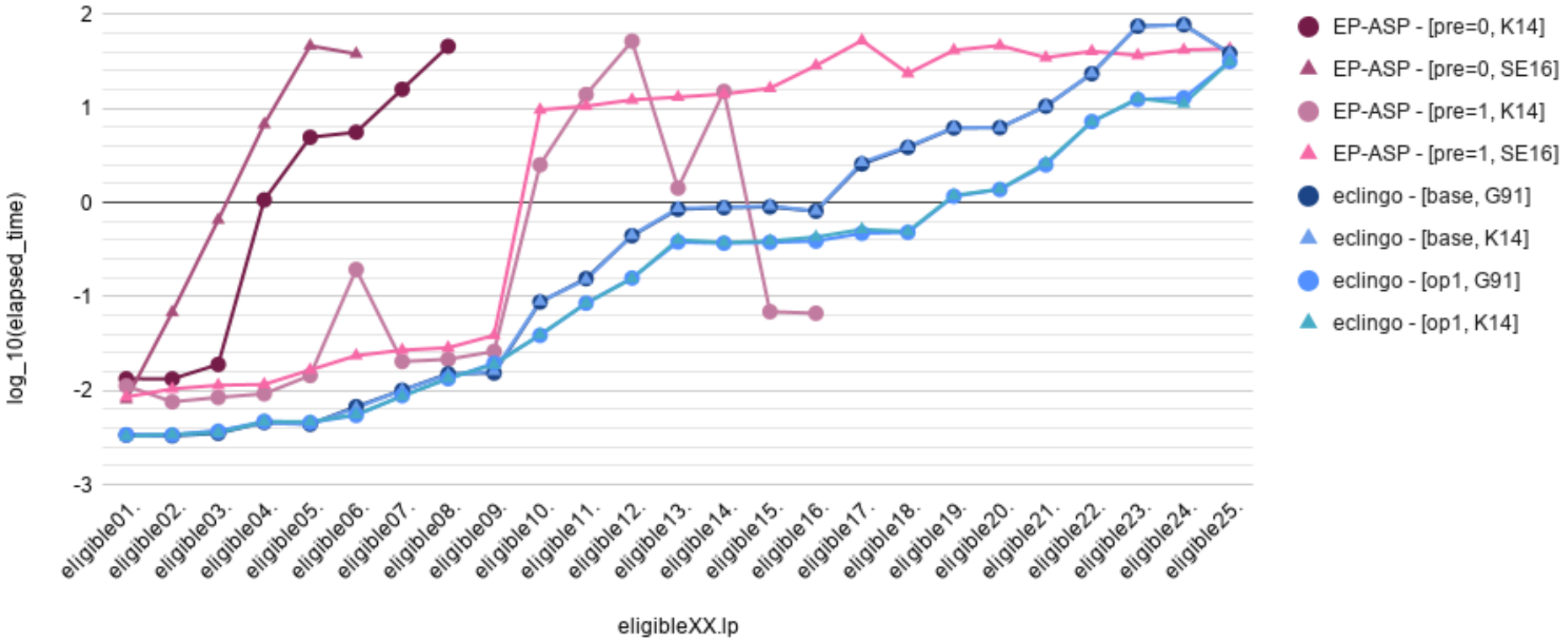
El paso 3 implica una llamada a `clingo` por cada solución de Π_{aux}.

Flujo de ejecución

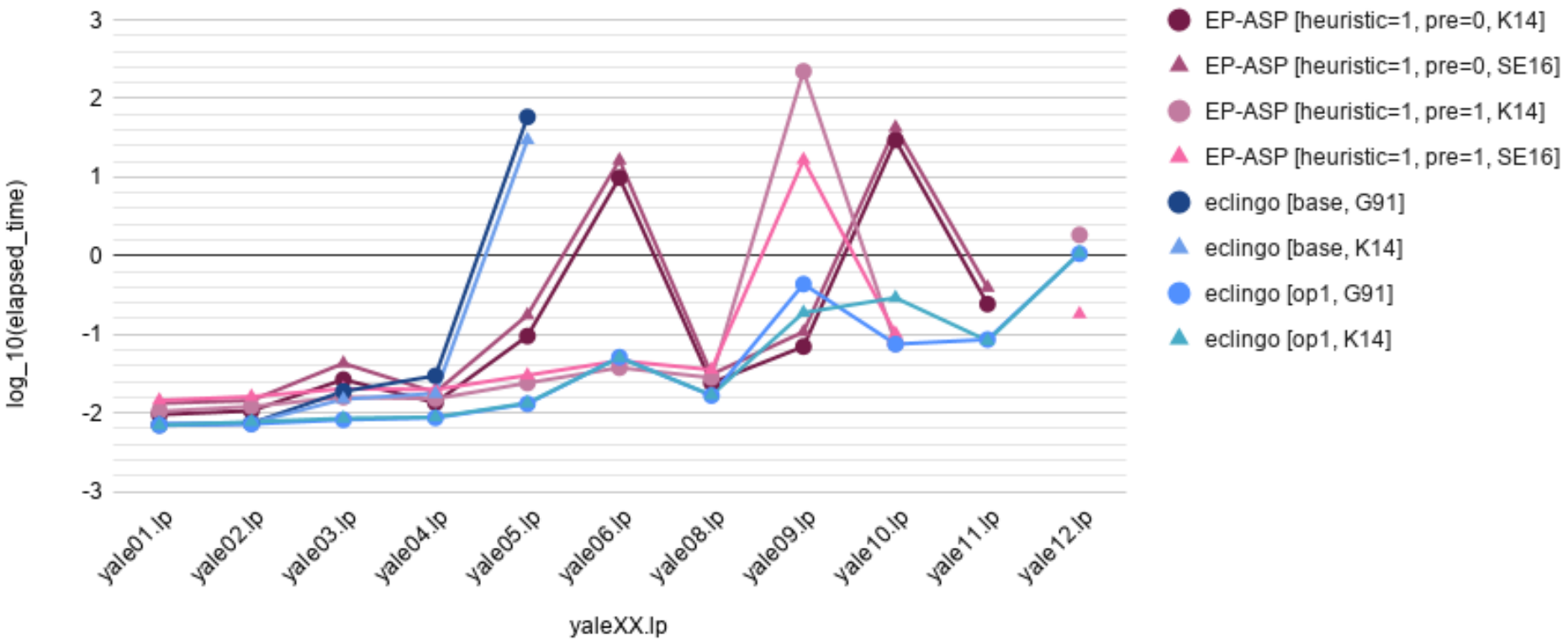


Índice

Scholarship Elegibility Problem



Yale Shooting Problem

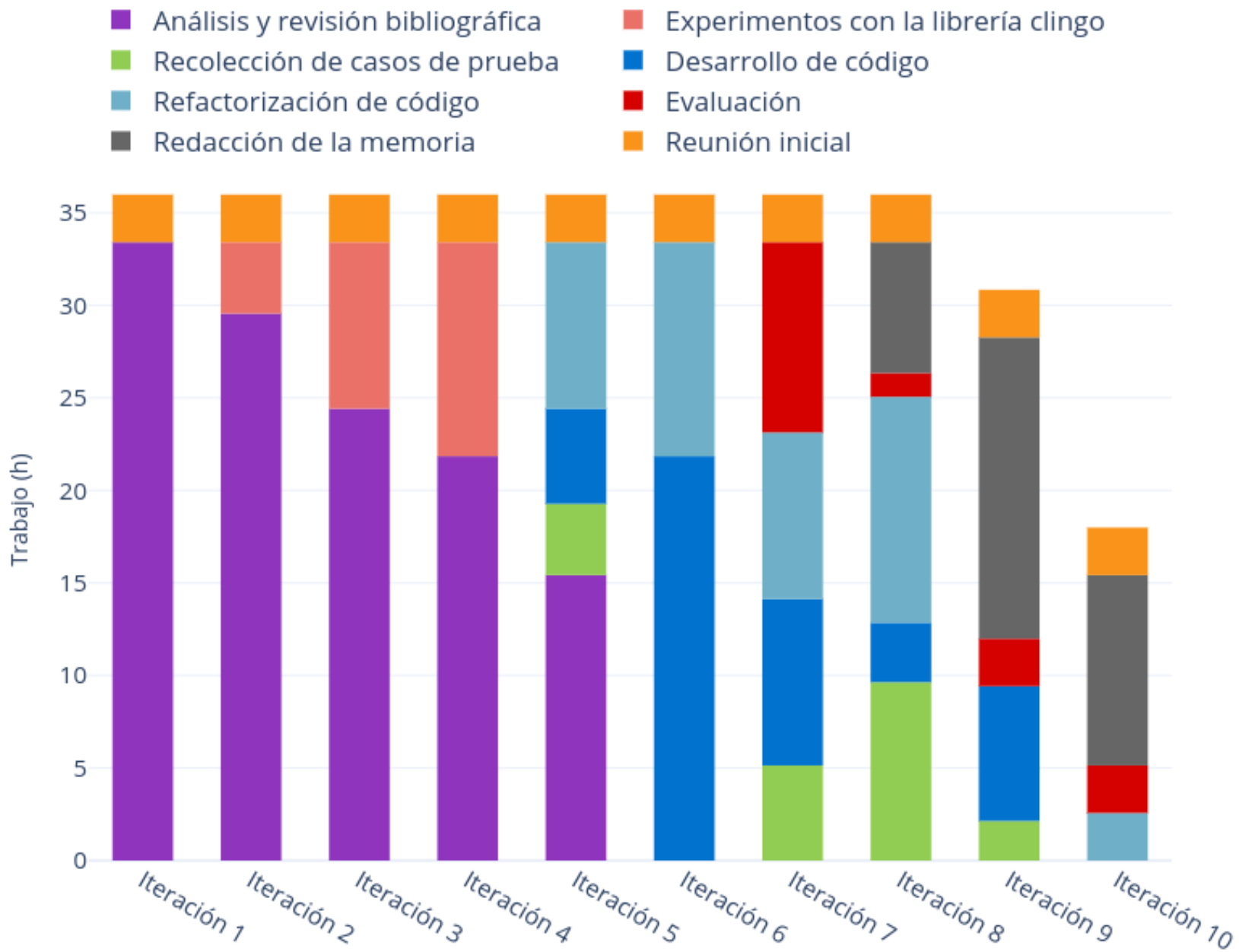


Índice

Metodología

- Aproximación **ágil**, **incremental** e **iterativa**.
  - *Sprints* de corta duración (1-2 semanas)
- Basada en las metodologías **scrum** y **extreme programming**.
  - Scrum
    - ★ Agenda de producto y *sprint*
    - ★ Planificación y revisión de *sprint*
  - Extreme programming
    - ★ Refactorización de código
    - ★ Desarrollo dirigido por pruebas
    - ★ Integración continua
- Pruebas software.
  - De caja blanca o estructurales.
  - De caja negra o funcionales.
- No se han tenido en cuenta las cuestiones relativas a la gestión del grupo de trabajo.

Seguimiento



Coste del proyecto

Recurso	Horas de trabajo (h)	Coste/hora (€/h)	Coste total (€)
Jefe de proyecto	13,5	20	270
Analista-programador	300,85	9	2.707,65
Coste total de recursos (€)			2.967,65

Recurso	Coste total (€)
`clingo`	0
Intérprete de `Python`	0
`pylint`	0
`pytest`	0
`git`	0
`Travis CI`	0
Servidor `AWS` 1	0*
Servidor `AWS` 2	0*
Coste total de recursos (€)	0

*Coste del servicio con el plan AWS Educate.

Índice

Conclusiones

- Se han cumplido los **objetivos propuestos**:
  1. `eclingo` calcula los modelos de un programa lógico epistémico bajo las semánticas G91 y K14.
  2. Sintaxis de entrada: sencilla, similar a la de `clingo`.
  3. Evaluación: resultados muy positivos frente a `EP-ASP` **[IJCAI 17]**.
- No se han detectado defectos o errores inesperados en el comportamiento de la herramienta.
  - **50 escenarios de prueba** agrupados en 4 categorías
  - Desarrollo dirigido por pruebas
  - Integración continua
- Trabajo futuro:
  - Nuevas mejoras.
  - Publicación.
  - Inclusión en el paquete de herramientas `Potassco`.

Un procesador de expresiones epistémicas en programas lógicos

Javier Garea Cidre

Director: Pedro Cabalar
Grado en Ingeniería Informática
Mención en Computación

¡Gracias por su atención!