

Planificación Continua mediante PDDL

Germán Braun

Facultad de Informática - Universidad Nacional del Comahue

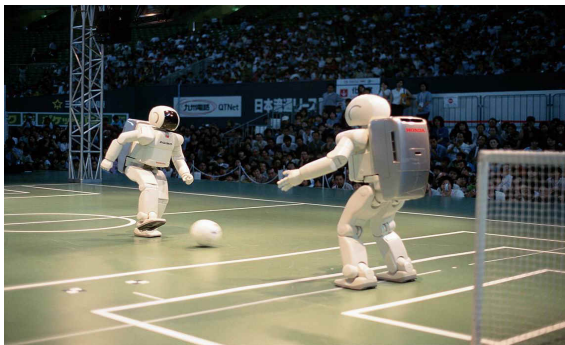
30 de Octubre de 2012

Agenda

- 1 Motivación
- 2 Objetivos
- 3 Traductor en Ciao Prolog
- 4 Demostración
- 5 Traducción de Requerimientos
- 6 Conclusiones

- 1 Motivación
- 2 Objetivos
- 3 Traductor en Ciao Prolog
- 4 Demostración
- 5 Traducción de Requerimientos
- 6 Conclusiones

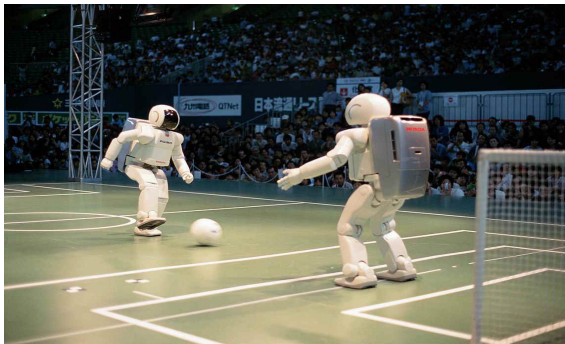
- Planificación Continua



<http://asimo.honda.com/>

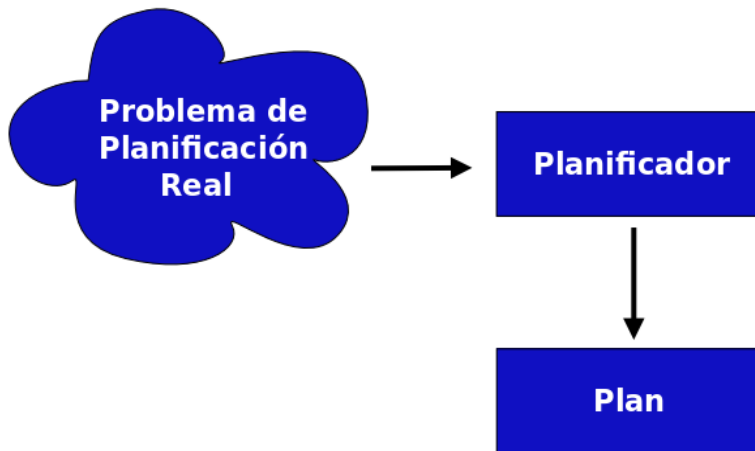
Motivación

- Planificación Continua
- Lenguaje de Definición de Dominios de Planificación (PDDL)

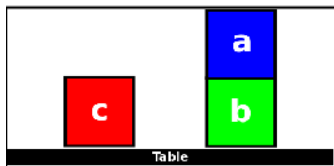
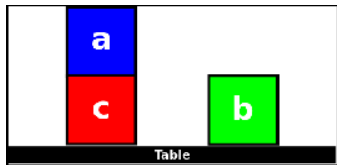


<http://asimo.honda.com/>

Planificación (1)



Planificación (2) - Ejemplo

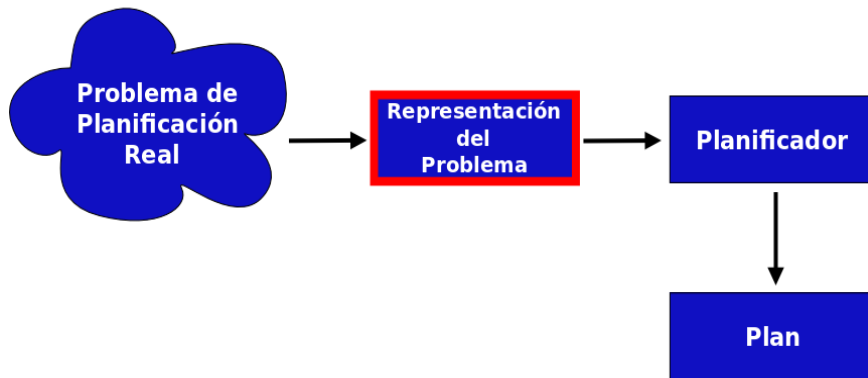


Planificador

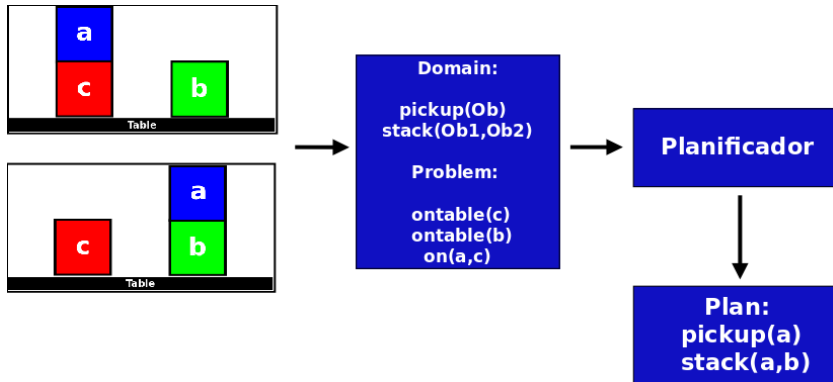


Plan:
pickup(a)
stack(a,b)

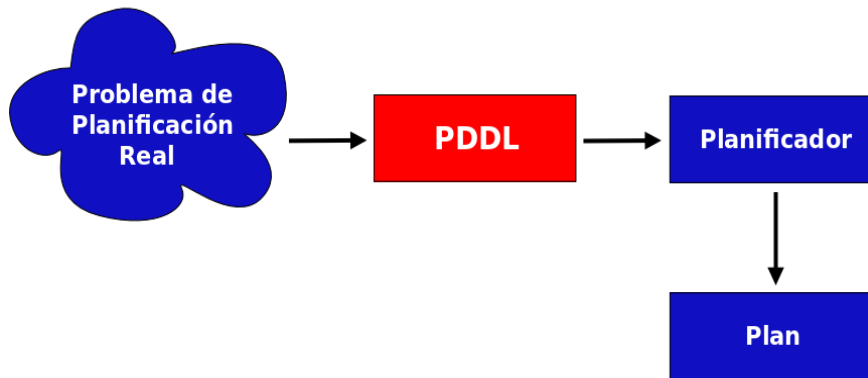
Planificación (3)



Planificación (4) - Ejemplo



Planificación (5)



- Lenguaje estándar

- **Lenguaje estándar**
 - *Ampliamente aceptado por la comunidad IA*

Planificación (6) - PDDL

- **Lenguaje estándar**
 - *Ampliamente aceptado por la comunidad IA*
- **Requerimientos**

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

- **Action-centered**

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

- **Action-centered**

- *Listas de precondiciones y efectos*

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

- **Action-centered**

- *Listas de precondiciones y efectos*

- **Dominio**

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

- **Action-centered**

- *Listas de precondiciones y efectos*

- **Dominio**

- *Acciones parametrizadas, Predicados, Constantes...*

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

- **Action-centered**

- *Listas de precondiciones y efectos*

- **Dominio**

- *Acciones parametrizadas, Predicados, Constantes...*

- **Problema**

Planificación (6) - PDDL

- **Lenguaje estándar**

- *Ampliamente aceptado por la comunidad IA*

- **Requerimientos**

- *strips (“núcleo”), igualdad, efectos condicionales, precondiciones disyuntivas, precondiciones universales...*

- **Action-centered**

- *Listas de precondiciones y efectos*

- **Dominio**

- *Acciones parametrizadas, Predicados, Constantes...*

- **Problema**

- *Objetos, metas y condiciones iniciales...*

Planificación (7) - Dominio PDDL

Ejemplo: *Dominio PDDL*

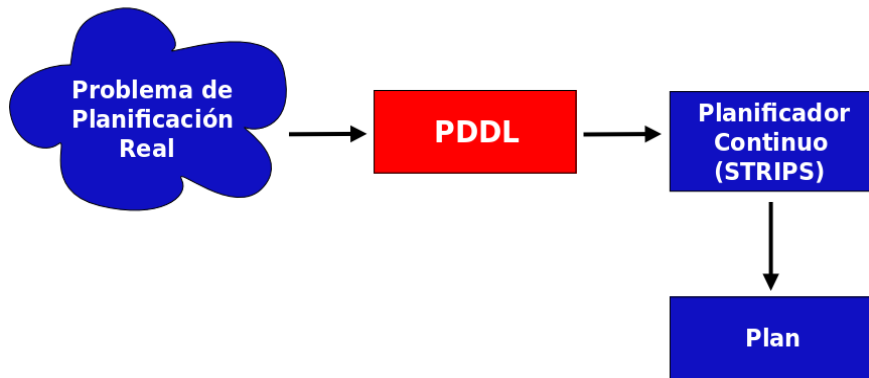
```
(define (domain bkwl)
  (:requirements :strips)
  (:predicates (clear ?x) (ontable ?x) (armempty)
               (holding ?x) (on ?x ?y))

  (:action stack
    :parameters (?ob ?underob)
    :precondition (and (clear ?underob) (holding ?ob))
    :effect (and (clear ?ob) (on ?ob ?underob) (armempty)
                 (not(clear ?underob))
                 (not(holding ?ob)))))
```

Ejemplo: *Problema PDDL*

```
(define (problem pb1)
  (:domain bkw)
  (:objects a b)
  (:goal (on a b))
  (:init (ontable c) (ontable b)
         (on a c) (clear a) (clear b) (armempty))
)
```

Planificación (9)



Motivación Principal

Motivación

Dotar al Planificador Continuo^a de un módulo traductor del lenguaje PDDL, permitiendo que el sistema de creencias de un agente soporte percepciones y acciones especificadas en este lenguaje.

^aTesis Mario Moya: “Control de Agentes Basado en Planificación Continua”.



SAP speaks PDDL



Planning aplicado a
e-learning

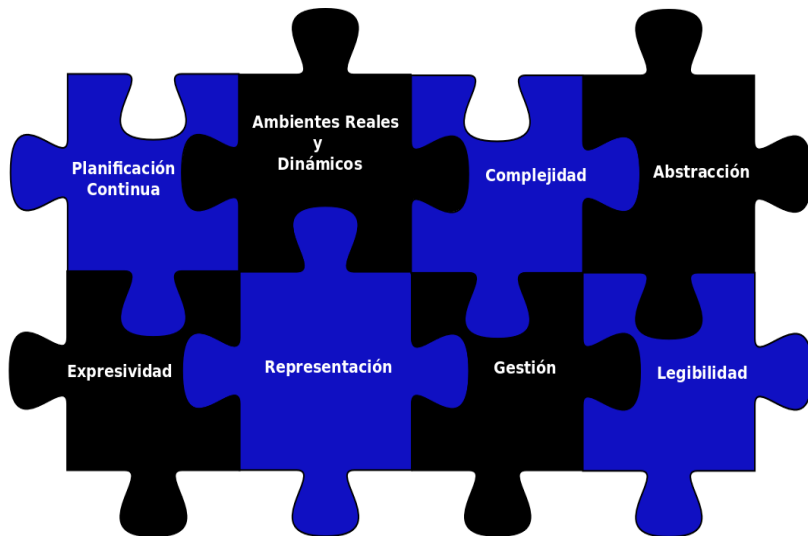
- 1 Motivación
- 2 Objetivos**
- 3 Traductor en Ciao Prolog
- 4 Demostración
- 5 Traducción de Requerimientos
- 6 Conclusiones

Objetivos (1)

Objetivo

Implementar un módulo capaz de procesar problemas de *planificación* en un subconjunto de *PDDL* y generar una especificación equivalente en el lenguaje *STRIPS* del *Planificador Continuo*.

Objetivos (2)



- 1 Motivación
- 2 Objetivos
- 3 Traductor en Ciao Prolog**
- 4 Demostración
- 5 Traducción de Requerimientos
- 6 Conclusiones

- **Lenguaje Fuente**

- **Lenguaje Fuente**
 - *Subconjunto de PDDL*

- **Lenguaje Fuente**

- *Subconjunto de PDDL*
- *Formalismos como variantes de PDDL*

- **Lenguaje Fuente**

- *Subconjunto de PDDL*
- *Formalismos como variantes de PDDL*
 - División por requerimientos.

- **Lenguaje Fuente**

- *Subconjunto de PDDL*
- *Formalismos como variantes de PDDL*
 - División por requerimientos.
 - **:strips** es incluido por defecto.

- `:strips -> PDDLSTRIPS`

Ejemplo: *PDDL*_{STRIPS}

```
(:action stack
:parameters  (?ob ?underob)
:precondition (and (clear ?underob) (holding ?ob))
:effect (and (clear ?ob) (on ?ob ?underob) (armempty)
            (not (clear ?underob))
            (not (holding ?ob))))
```

- :equality \rightarrow PDDL_L

Ejemplo: PDDL_L

```
(:action stack
  :parameters (?X ?Y)
  :precondition (and (clear table) (= ?Y table))
  :effect (and (on ?X table) (not (clear table))))

(:action stack1
  :parameters (?X ?Y)
  :precondition (and (clear ?Y) (not (= ?Y table)))
  :effect (and (on ?X ?Y) (not (clear ?Y))))
```

- :conditional-effect \rightarrow PDDL_C

Ejemplo: PDDL_C

```
(:action stack
  :parameters (?X ?Y ?Z)
  :precondition (and (clear ?X) (clear ?Z) (on ?X ?Y))
  :effects (and (on ?X ?Z) (clear ?Y) (not (on ?X ?Y))
                (when (not (= table ?Z))
                  (not (clear ?Z)))))
```

Lenguaje Fuente - Precondiciones Disyuntivas

- :disjunctive-preconditions \rightarrow PDDL_D

Ejemplo: PDDL_D

```
(:action stack
  :parameters (?X ?Y ?Z)
  :precondition (and (or (istable ?Z) (clear ?Z))
                    (clear ?X) (on ?X ?Y))
  :effect (and (on ?X ?Z) (clear ?Y)
              (not (clear ?Y))
              (not (on ?X ?Y))))
```

- :universal-preconditions \rightarrow $PDDL_u$

Ejemplo: $PDDL_u$

```
(:action stack
:parameters  (?ob ?underob)
:precondition (and (forall (?block) (ontable ?block))
                  (clear ?underob) (holding ?ob))
:effect (and (clear ?ob) (on ?ob ?underob) (armempty)
            (not (clear ?underob))
            (not (holding ?ob))))
```

- **Lenguaje Destino**

- **Lenguaje Destino**
 - *Representación Genérica (Prolog-like)*

- **Lenguaje Destino**
 - *Representación Genérica (Prolog-like)*
 - *Independiente del Planificador destino*

- **Lenguaje Destino**

- *Representación Genérica (Prolog-like)*
 - *Independiente del Planificador destino*
 - *Permite adaptar el traductor a otros Planificadores basados en STRIPS*

- **Lenguaje Destino**

- *Representación Genérica (Prolog-like)*
 - *Independiente del Planificador destino*
 - *Permite adaptar el traductor a otros Planificadores basados en STRIPS*
 - **Nuestra Implementación:** *STRIPS-like -> Es el lenguaje del Planificador Continuo y se obtiene a partir de la representación genérica anterior*

Definición: *Dominio*

```
preconditions(action_name_i(parameters),  
              [predicate_j(parameters_k),...]).
```

```
achieves(action_name_i(parameters),  
          [predicate_j(parameters_k),...]).
```

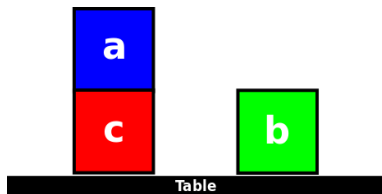
```
deletes(action_name_i(parameters),  
         [predicate_j(parameters_k),...]).
```

Definición: *Problema*

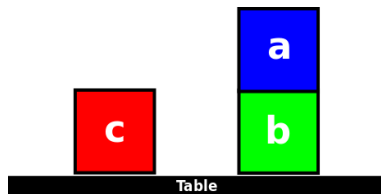
```
(domain(domain_name),  
  objects(obj_1,obj_2,...,obj_N),  
  goal(fact_g),  
  init(fact_1,fact_2,...,fact_N)).
```

- 1 Motivación
- 2 Objetivos
- 3 Traductor en Ciao Prolog
- 4 Demostración**
- 5 Traducción de Requerimientos
- 6 Conclusiones

DEMO



Estado Inicial



Estado Final

- 1 Motivación
- 2 Objetivos
- 3 Traductor en Ciao Prolog
- 4 Demostración
- 5 Traducción de Requerimientos**
- 6 Conclusiones

Traducción ¿Cómo es?

- Conceptos

Traducción ¿Cómo es?

- Conceptos
 - *Esquemas de Compilación*

Traducción ¿Cómo es?

- **Conceptos**

- *Esquemas de Compilación*
- *Compilabilidad*

Traducción - Esquemas de Compilación

- Básicamente, un **esquema de compilación** es un mapeo entre dos formalismos de planificación X e Y.

- Básicamente, un **esquema de compilación** es un mapeo entre dos formalismos de planificación X e Y.

Definición: *Esquemas de Compilación*

$F(\Pi) = \langle f_\xi(\Xi), I \cup f_i(\Xi), G \cup f_g(\Xi) \rangle$, donde Ξ es un dominio, I es el estado inicial y G es un conjunto de metas.

Traducción - Esquemas de Compilación

- Básicamente, un **esquema de compilación** es un mapeo entre dos formalismos de planificación X e Y.

Definición: *Esquemas de Compilación*

$F(\Pi) = \langle f_\xi(\Xi), I \cup f_i(\Xi), G \cup f_g(\Xi) \rangle$, donde Ξ es un dominio, I es el estado inicial y G es un conjunto de metas.

Condición Importante

Existe un plan para Π , si y solo si, existe un plan para $F(\Pi)$, donde Π es la definición del dominio en el formalismo X y $F(\Pi)$ es la definición del dominio en el formalismo Y.

Traducción - Compilabilidad (1)

- Los esquemas de compilación permiten definir una relación entre formalismos, llamada *Compilabilidad (Compilability)*.

- Los esquemas de compilación permiten definir una relación entre formalismos, llamada *Compilabilidad (Compilability)*.

Definición: *Compilabilidad*

Un formalismo de planificación X es **compilable** al formalismo Y , expresado como $X \preceq^x Y$, si y sólo si, existe un esquema de compilación de X a Y .

Compilabilidad (2)

- Si $\mathbf{X} \preceq^1 \mathbf{Y}$, entonces el tamaño del plan es preservado exactamente.

Compilabilidad (2)

- Si $\mathbf{X} \preceq^1 \mathbf{Y}$, entonces el tamaño del plan es preservado exactamente.
- Si $\mathbf{X} \preceq^c \mathbf{Y}$, entonces el tamaño del plan es preservado linealmente (en $||\Delta||$), donde $||\Delta||$ es el tamaño del plan obtenido en \mathbf{X} .

Compilabilidad (2)

- Si $X \preceq^1 Y$, entonces el tamaño del plan es preservado exactamente.
- Si $X \preceq^c Y$, entonces el tamaño del plan es preservado linealmente (en $||\Delta||$), donde $||\Delta||$ es el tamaño del plan obtenido en X .
- Si $X \preceq^p Y$, entonces el tamaño del plan es preservado polinomialmente (en $||\Delta||$ y $||\Pi||$), donde $||\Pi||$ es el número de acciones en X .

Compilabilidad (2)

- Si $X \preceq^1 Y$, entonces el tamaño del plan es preservado exactamente.
- Si $X \preceq^c Y$, entonces el tamaño del plan es preservado linealmente (en $||\Delta||$), donde $||\Delta||$ es el tamaño del plan obtenido en X .
- Si $X \preceq^p Y$, entonces el tamaño del plan es preservado polinomialmente (en $||\Delta||$ y $||\Pi||$), donde $||\Pi||$ es el número de acciones en X .
- Si $X \preceq^x_p Y$, entonces la compilación es en tiempo polinomial y el tamaño del plan es preservado polinomialmente (en $||\Delta||$ y $||\Pi||$).

Compilabilidad (3)

- Entonces, considerando el lenguaje fuente y destino de nuestra implementación, definimos las siguientes relaciones:

Compilabilidad (3)

- Entonces, considerando el lenguaje fuente y destino de nuestra implementación, definimos las siguientes relaciones:
 - $\text{PDDL}_{\text{STRIPS}} \preceq^1 \text{STRIPS}$

Compilabilidad (3)

- Entonces, considerando el lenguaje fuente y destino de nuestra implementación, definimos las siguientes relaciones:
 - $\text{PDDL}_{\text{STRIPS}} \preceq^1 \text{STRIPS}$
 - $\text{PDDL}_L \preceq_p^1 \text{STRIPS}$

Compilabilidad (3)

- Entonces, considerando el lenguaje fuente y destino de nuestra implementación, definimos las siguientes relaciones:
 - $\text{PDDL}_{\text{STRIPS}} \preceq^1 \text{STRIPS}$
 - $\text{PDDL}_L \preceq^1_p \text{STRIPS}$
 - $\text{PDDL}_C \preceq^x_p \text{STRIPS}$

Compilabilidad (3)

- Entonces, considerando el lenguaje fuente y destino de nuestra implementación, definimos las siguientes relaciones:
 - $\text{PDDL}_{\text{STRIPS}} \preceq^1 \text{STRIPS}$
 - $\text{PDDL}_L \preceq^1_p \text{STRIPS}$
 - $\text{PDDL}_C \preceq^x_p \text{STRIPS}$
 - $\text{PDDL}_D \preceq^1_p \text{STRIPS}$

Compilabilidad (3)

- Entonces, considerando el lenguaje fuente y destino de nuestra implementación, definimos las siguientes relaciones:
 - $\text{PDDL}_{\text{STRIPS}} \preceq^1 \text{STRIPS}$
 - $\text{PDDL}_L \preceq^1_p \text{STRIPS}$
 - $\text{PDDL}_C \preceq^x_p \text{STRIPS}$
 - $\text{PDDL}_D \preceq^1_p \text{STRIPS}$
 - $\text{PDDL}_u \preceq^1_p \text{STRIPS}$

Compilabilidad - Ejemplo (1)

- $PDDL_u \preceq_p^1 STRIPS$

Ejemplo: *Problema*

```
(define (problem pb1)
  (:domain bkwup)
  (:objects a b c)
  (:goal (on a b))
  (:init (ontable c) (ontable b) (ontable a)
         (on a c) (clear a) (clear b) (armempty))
)
```

Ejemplo: *Dominio*

```
(:action stack
  :parameters  (?ob ?underob)
  :precondition (and (forall (?block) (ontable ?block))
                    (clear ?underob) (holding ?ob))
  :effect (and (clear ?ob) (on ?ob ?underob) (armempty)
              (not (clear ?underob))
              (not (holding ?ob))))
```

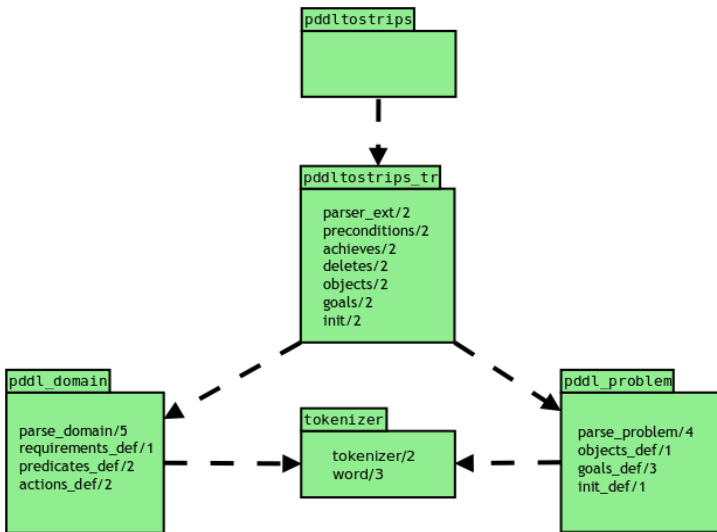
Compilabilidad - Ejemplo (3)

Ejemplo: *STRIPS*

```
% stack(X,Y)
preconditions(stack(X,Y),[ontable(a),ontable(b),
                        ontable(c),
                        clear(Y),holding(X)]).

deletes(stack(X,Y),clear(Y)).
deletes(stack(X,Y),holding(X)).
achieves(stack(X,Y),clear(X)).
achieves(stack(X,Y),on(X,Y)).
achieves(stack(X,Y),armempty).
```

Arquitectura Modular



- 1 Motivación
- 2 Objetivos
- 3 Traductor en Ciao Prolog
- 4 Demostración
- 5 Traducción de Requerimientos
- 6 Conclusiones**

Resultados (1)

Teorema

$$STRIPS_u \preceq_p^1 STRIPS$$

Resultados (1)

Teorema

$$STRIPS_u \preceq_p^1 STRIPS$$

Corolario

$$PDDL_u \preceq_p^1 STRIPS$$

Resultados (2)

- **PDDL_{STRIPS}**

Resultados (2)

- **PDDL_{STRIPS}**
 - Requerimiento :strips

Resultados (2)

- $\text{PDDL}_{\text{STRIPS}}$
 - Requerimiento :strips
- PDDL_L

Resultados (2)

- **PDDL_{STRIPS}**
 - Requerimiento :strips
- **PDDL_L**
 - Requerimiento :equality

Resultados (2)

- $\text{PDDL}_{\text{STRIPS}}$
 - Requerimiento :strips
- PDDL_L
 - Requerimiento :equality
- PDDL_C

Resultados (2)

- **PDDL_{STRIPS}**
 - Requerimiento :strips
- **PDDL_L**
 - Requerimiento :equality
- **PDDL_C**
 - Requerimiento: conditional-effect

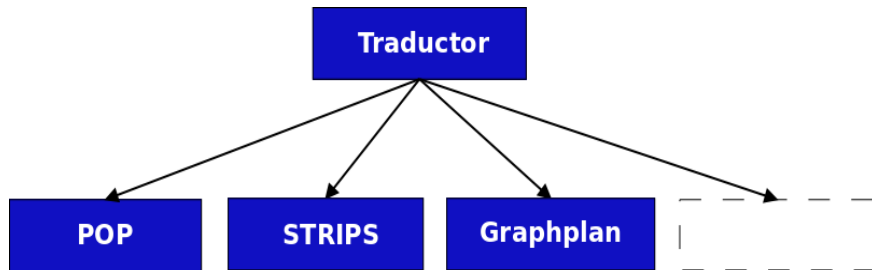
Resultados (2)

- $\text{PDDL}_{\text{STRIPS}}$
 - Requerimiento :strips
- PDDL_L
 - Requerimiento :equality
- PDDL_C
 - Requerimiento: conditional-effect
- PDDL_D

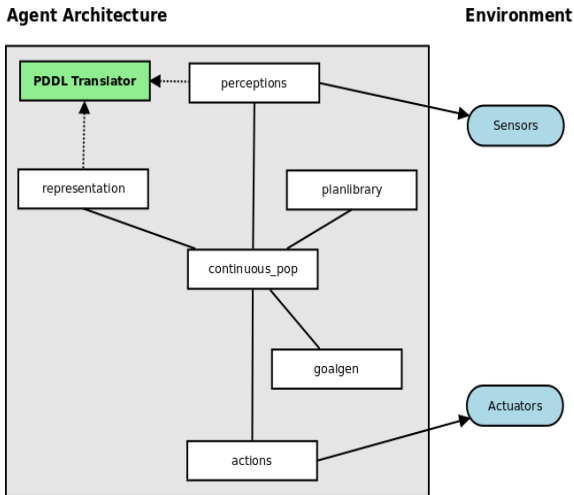
Resultados (2)

- **PDDL_{STRIPS}**
 - Requerimiento :strips
- **PDDL_L**
 - Requerimiento :equality
- **PDDL_C**
 - Requerimiento: conditional-effect
- **PDDL_D**
 - Requerimiento: disjunctive-preconditions

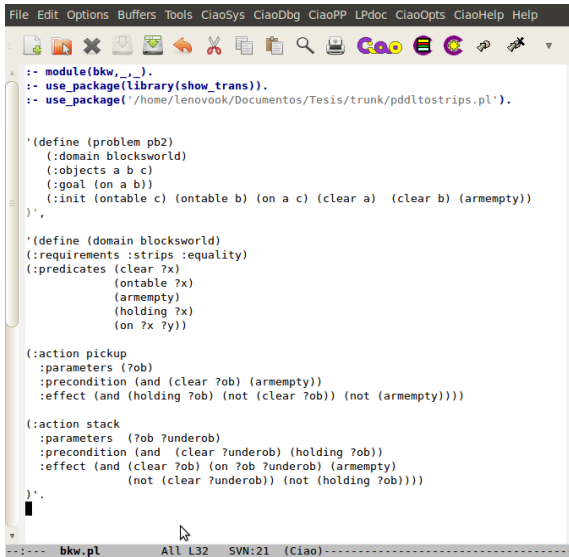
Adaptación a otros Planificadores



Integración con el Framework de Planificación Continua



Expansión Sintáctica para Ciao Prolog



```
-- module(bkw,_,_).
-- use_package(library(show_trans)).
-- use_package('/home/lenovook/Documentos/Tesis/trunk/pddltostrips.pl').

'(define (problem pb2)
  (:domain blocksworld)
  (:objects a b c)
  (:goal (on a b))
  (:init (ontable c) (ontable b) (on a c) (clear a) (clear b) (armempty)))',

'(define (domain blocksworld)
  (:requirements :strips :equality)
  (:predicates (clear ?x)
                (ontable ?x)
                (armempty)
                (holding ?x)
                (on ?x ?y)))

(:action pickup
  :parameters (?ob)
  :precondition (and (clear ?ob) (armempty))
  :effect (and (holding ?ob) (not (clear ?ob)) (not (armempty))))

(:action stack
  :parameters (?ob ?underob)
  :precondition (and (clear ?underob) (holding ?ob))
  :effect (and (clear ?ob) (on ?ob ?underob) (armempty)
              (not (clear ?underob)) (not (holding ?ob))))

)'.

--:---  bkwp.pl          All L32   SVN:21   (Ciao)-----
```

- **Ampliar el Lenguaje Fuente del Traductor**

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**
 - *Pruebas sobre dominios reales.*

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**
 - *Pruebas sobre dominios reales.*
 - *Traducción de percepciones en tiempo real.*

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**
 - *Pruebas sobre dominios reales.*
 - *Traducción de percepciones en tiempo real.*
- **Aplicar conceptos de Compiladores e Intérpretes**

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**
 - *Pruebas sobre dominios reales.*
 - *Traducción de percepciones en tiempo real.*
- **Aplicar conceptos de Compiladores e Intérpretes**
 - *Manipulación y Recuperación de Errores.*

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**
 - *Pruebas sobre dominios reales.*
 - *Traducción de percepciones en tiempo real.*
- **Aplicar conceptos de Compiladores e Intérpretes**
 - *Manipulación y Recuperación de Errores.*
- **Combinar el Traductor con otros Planificadores**

- **Ampliar el Lenguaje Fuente del Traductor**
 - *Definir esquemas de compilación asociados.*
- **Análisis exhaustivo de la complejidad del Traductor**
 - *Disminuir el impacto de la traducción en la planificación.*
- **Interface para el Framework de Planificación Continua**
 - *Pruebas sobre dominios reales.*
 - *Traducción de percepciones en tiempo real.*
- **Aplicar conceptos de Compiladores e Intérpretes**
 - *Manipulación y Recuperación de Errores.*
- **Combinar el Traductor con otros Planificadores**
 - *Realizar comparaciones de performance.*

¿Preguntas?

¡Gracias!

<http://code.google.com/p/my-pddl-to-strips-tesis/>



Anexo (1) - Implementaciones Existentes

- Analizador en SWI-Prolog

Anexo (1) - Implementaciones Existentes

- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*

Anexo (1) - Implementaciones Existentes

- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*
 - *No genera STRIPS*

Anexo (1) - Implementaciones Existentes

- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*
 - *No genera STRIPS*
 - *No incluye alguno de los requerimientos presentados aquí*

Anexo (1) - Implementaciones Existentes

- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*
 - *No genera STRIPS*
 - *No incluye alguno de los requerimientos presentados aquí*
- **Gramática ANTLR para PDDL**

Anexo (1) - Implementaciones Existentes

- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*
 - *No genera STRIPS*
 - *No incluye alguno de los requerimientos presentados aquí*
- **Gramática ANTLR para PDDL**
 - *No genera código Prolog ni STRIPS*

Anexo (1) - Implementaciones Existentes

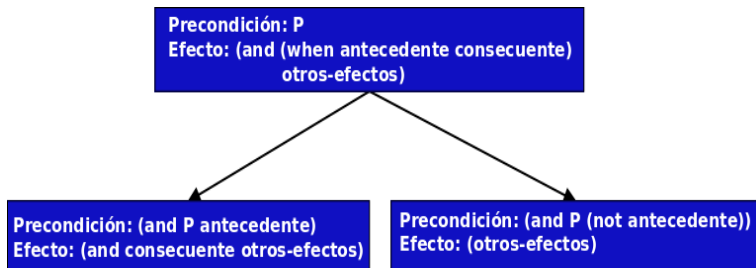
- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*
 - *No genera STRIPS*
 - *No incluye alguno de los requerimientos presentados aquí*
- **Gramática ANTLR para PDDL**
 - *No genera código Prolog ni STRIPS*
- **Librería PDDL4J**

Anexo (1) - Implementaciones Existentes

- **Analizador en SWI-Prolog**
 - *Otra implementación Prolog*
 - *No genera STRIPS*
 - *No incluye alguno de los requerimientos presentados aquí*
- **Gramática ANTLR para PDDL**
 - *No genera código Prolog ni STRIPS*
- **Librería PDDL4J**
 - *JAVA*

Anexo (2) - Esquemas de Traducción

- PDDL_C



Anexo (3) - Diagrama de Traducción

