Universidad de Granada

decsai.ugr.es

Tema 3: Aprendizaje en Planificación Automática

Curso en Minería de Procesos (Gestión de Procesos)

Juan Fernández Olivares

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

1. **Introducción**
   1. Process Mining en Adaptive Case Management = Aprendizaje en Planificación Automática
   2. Planificación contínua
   3. Agentes autónomos (no interacción usuarios)
   4. Herramientas inteligentes (interacción usuarios)
2. **¿Por qué aplicar aprendizaje en Planificación?**
3. **¿Dónde y qué aprender?**
4. **Autonomía y aprendizaje por refuerzo en agentes reactivos.**
   1. Para entender aprendizaje en deliberativos.
5. **Mejorar la autonomía del planificador**
   1. Aprender a plantearse objetivos por sí mismo
   2. Aprender su modelo de proceso.
      1. Modelos basados en acciones
      2. Modelos jerárquicos.

Con Process Mining para BPM clásico hemos visto cómo aprender modelos de proceso para trabajo rutinario (procesos estáticos) a partir de logs de eventos, ahora vamos a ver cómo aprender **modelos prescriptivos**: modelos de proceso para knowledge work (procesos dinámicos, orientados a objetivos) en el marco de Adaptive Case Management.

Perspectivas de la planificación automática

- Aspecto 1: es una técnica para implementar agentes autónomos:
  - Robótica.
  - Videojuegos
  - Un agente sólo es completamente autónomo si tiene la capacidad de adaptarse al medio
  - Aprender es mejorar su modelo de comportamiento.
- Aspecto2: técnica de resolución de problemas.
  - Aprender es mejorar el conocimiento para resolver problemas por sí solo.
- Aspecto3: técnica para gestión inteligente de procesos ->
  - Aprender es mejorar un modelo de proceso observando cómo toman los humanos decisiones para ayudarles a gestionar mejor su trabajo
  - Aprender -> Adaptarse
- ACM: Adaptive Case Management:
  - Adaptive: se generan procesos adaptados a una situación.
  - Adaptive: se generan mejores procesos porque el modelo de proceso es adaptativo -> Aprender modelos de proceso.

Experiencia
Normas
Buenas Prácticas

Conocimiento

Bases de datos Externas

Organización

Recursos

Estado actual

Dominio HTN

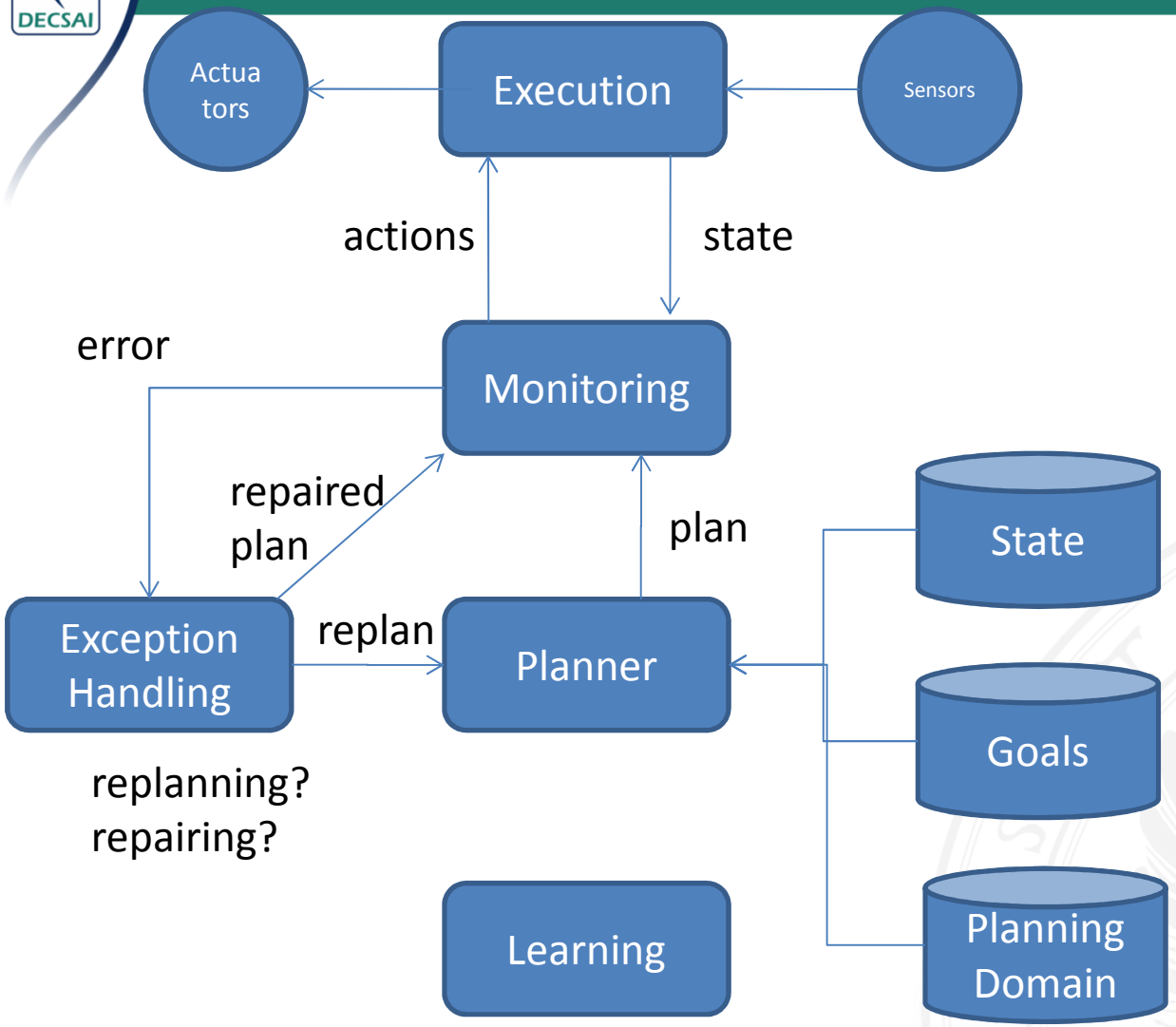
Estado Inicial

Planificador HTN

Decide

Plan de actuación

Objetivos



- In order to obtain a robust autonomous behaviour, since the world is dynamic and uncertain, planning applications have to cope with unexpected events that invalidate the current plan.
- From an engineering p.o.v, the monitoring has to detect errors that are managed by an exception handling process. Basically, it decides wether to replan (build a new plan from the current state) or repair the plan.

M. Fox, A. Gerevini, D. Long, y I. Serina, «Plan stability: Replanning versus plan repair», in *Proceeding of ICAPS06*, 2006, pp. 212-221.
K. L. Myers, «CPEF: A Continuous Planning and Execution Framework», *AI Magazine*, vol. 20, n.º 4, p. 63, dic. 1999.

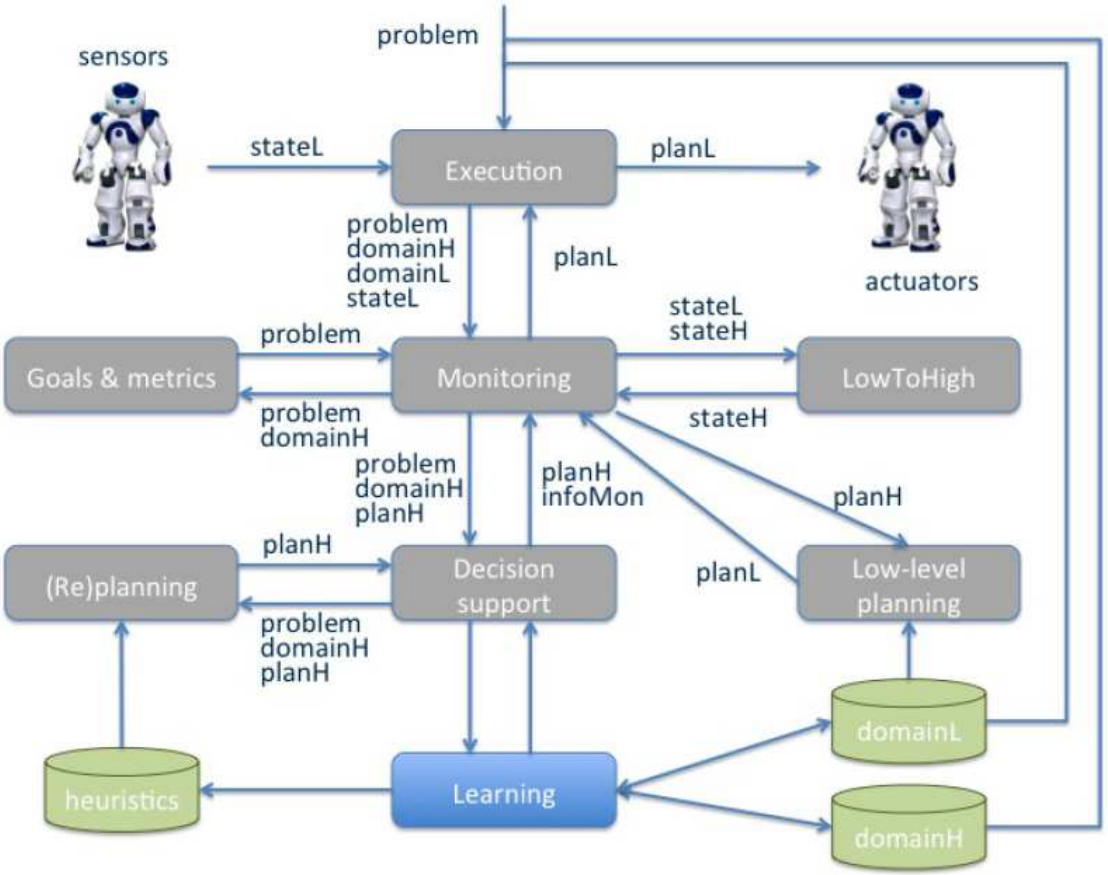
Extracted from SPARK Workshops at ICAPS 2007, 2008, 2009, 2010

http://decsai.ugr.es/~lcv/SPARK/

- Autonomous systems (27/43): aimed at achieving a fully automated, intelligent system
  - Autonomous systems
    - Aerospace and ocean applications (17/43)
      - *Spacecrafts, Aircrafts, submarine AGVs*
    - Robots (2/43)
    - Semantic web (web services) (4/43)
  - Manufacturing and system control
    - Power-plants, nuclear-plants (4/43)
- Intelligent tools (10/43): aimed at achieving support for human decision making
  - Logistics and transport (2/43)
  - Tourism, e-learning, emergencies, medical (5/43)
  - Project management, BPM and workflow (3/43)
- Others (4/43)
  - Experiments in the edge, like generation of natural language sentences via AI planning
  - Testbeds,…

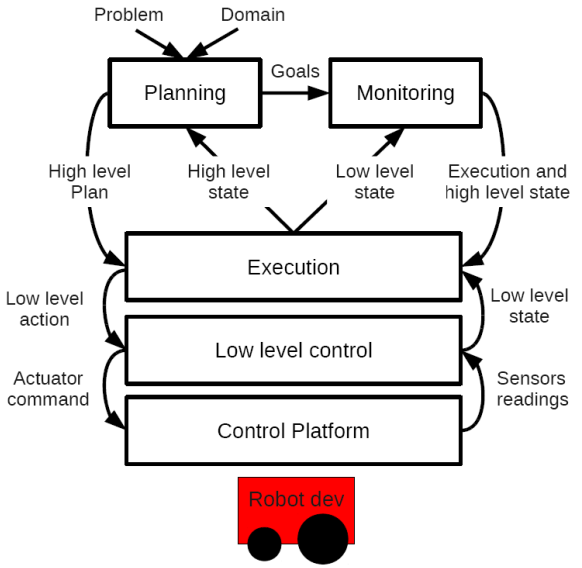
Agente deliberativo: decide qué acciones ejecutar representando y razonando sobre las consecuencias de sus acciones.

- Planificación contínua: base inicial para alcanzar autonomía total.
- Agentes físicos autónomos: robots.
- Agentes virtuales autónomos: Jugadores autónomos o NPCs en videojuegos.



ROS

http://www.ros.org/



http://servergrps.dsic.upv.es/pelea/

Pelea
TIN2008-06701-C03
Planning, Execution and LEarning Architecture

GRPS Group of Reasoning on Planning and Scheduling
PLG Planning & Learning research Group
ISG Intelligent Systems Group

http://stavros.lostre.org/category/planning/

- Non-player characters (NPCs):
  - Move in game-world space (2D/3D)
  - Act in game-world
    - Pick-up objects, attack, hide, etc.
  - Exhibit behavior
    - Follow the player, get scared and runaway, etc.
- How do they think?
  - Navigation: Pathfinding
  - Action-driven behavior:
    Finite State Machines / Behavior Trees / Goal Oriented Action Planning / Utility systems

HTNs in videogames

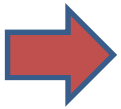
- Overview on aigamedev.com:planning-in-games

STRIPS→ Goal-Oriented Action Planning

- Overview on aigamedev.com:planning-in-games

¿Por qué aplicar aprendizaje en Planificación?

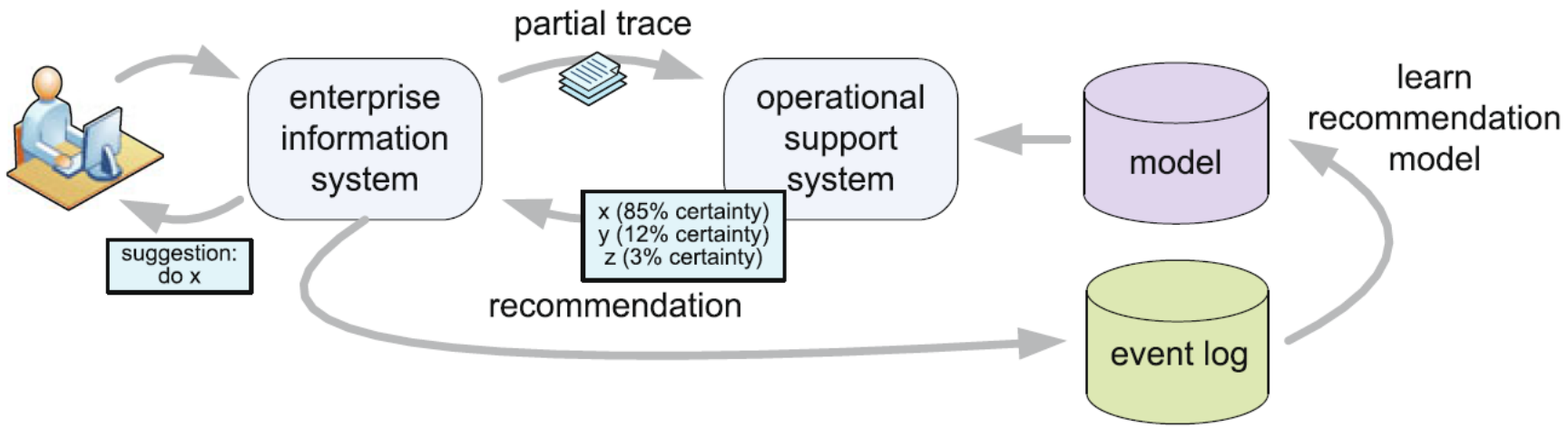
- Mejorar el comportamiento autónomo
  - Para sistemas autónomos la necesidades imponen que los agentes se adapten al entorno, para reducir barreras de adopción por usuarios.
    - Pocos usuarios aceptarían un robot que tienen que programar,
    - Los videojuego con NPCs inteligentes son más atractivos y tienen mayor índice de adopción (más ventas).
- Reducir el esfuerzo de modelado
  - Para herramientas inteligentes (como las usadas en ACM) el modelado es un cuello de botella que dificulta la adopción de este tipo de herramientas.
    - ¿Para qué esforzarme en codificar un conocimiento del que ya dispongo ejemplos en los logs?.



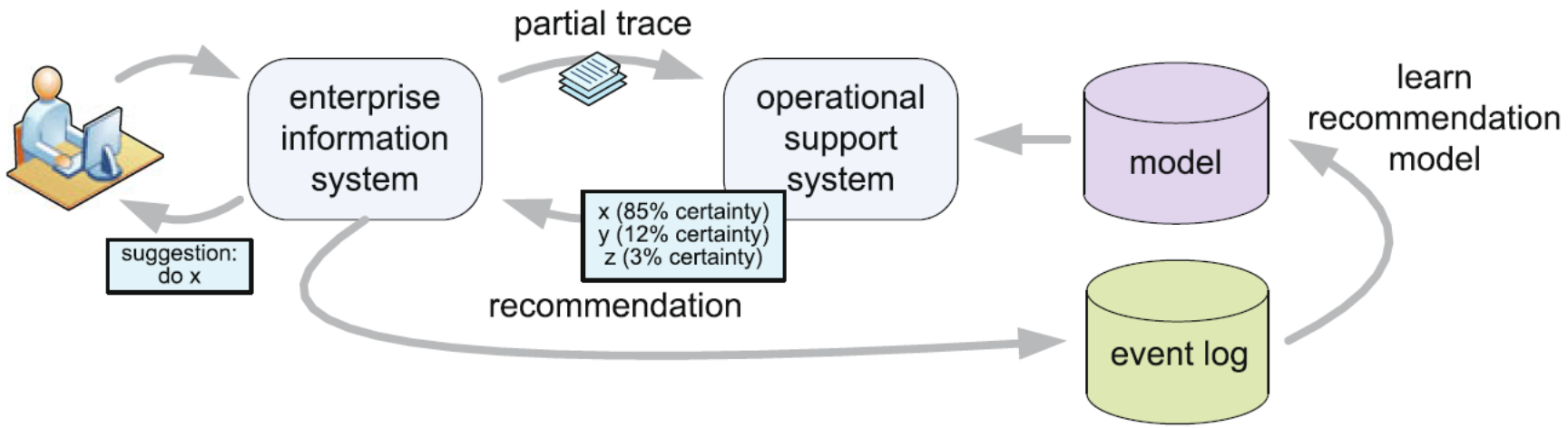
- Ayudar a los usuarios (expertos de un dominio) a tomar mejores decisiones.
  - La interacción con el usuario y la experiencia acumulada en la resolución de episodios permite mejorar el modelo de proceso (dominio de planificación) y afinar mejor en las decisiones.
- Mejorar eficiencia del proceso de planificación
  - La planificación es un proceso de búsqueda, con espacios de búsqueda exponenciales. El aprendizaje a partir de episodios previos ayuda a reducir las alternativas posibles en el proceso de planificación.

Ayuda a la decisión "online"

- Caso de uso: predecir el tiempo restante de un proceso o recomendar acciones para minimizar coste.
- ¿Qué funcionalidades son interesantes para ayuda a la decisión online?
  - Detectar: eventos y modelos se usan para explorar procesos en tiempo de ejecución. Los casos actuales en ejecución pueden compararse con casos similares previos.
  - Predecir: Combinando información de casos en ejecución con modelos es posible hacer predicciones sobre el futuro (¿cuánto tiempo resta para finalizar, qué probabilidad de éxito tiene el proceso?
  - Recomendar: la información usada para predecir puede también ser usada para **recomendar** posibles acciones orientadas a objetivos (minimizar el coste o el tiempo …)

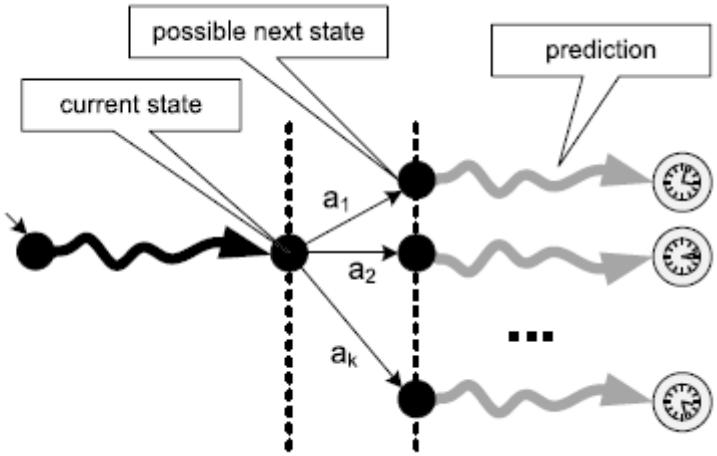


- Recomendación/prescripción: ¿qué se puede hacer a continuación?. Elementos:
  - Aprender un **modelo prescriptivo** a partir de event logs.
  - El Sistema de Soporte Operativo (motor BPM, p. ej) necesita conocer el **espacio de decisión** (cual es el conjunto de posibles acciones a escoger)
  - Las acciones se ordenan en el tiempo de acuerdo al modelo prescriptivo
  - Ejemplo: el sistema recomienda “hacer x con un 85% de certidumbre”
  - No siempre es posible una recomendación óptima
  - La mejor elección para la siguiente acción (o etapa) depende de eventos exógenos en el futuro.



- Una prescripción/recomendación se hace siempre respecto a un *objetivo/meta específico:*
  - Minimizar el tiempo restante
  - Minimizar los costes totales
  - Maximizar la fracción de casos gestionados en 4 semanas
  - Maximizar la fracción de casos
- Los objetivos pueden agregarse/combinarse: por ejemplo, balancear reducción de coste y de tiempo restante.
- Para hacer operativos objetivos, hay que definir indicadores de rendimiento (KPI: key performance indicators). P.e: en los logs tiene que aparecer información sobre tiempo o coste de actividades.

Fig. 10.12
Recommendations can be based on predictions. For every possible choice, simply predict the performance indicator of interest. Then, recommend the best one(s)



- Las recomendaciones no solo se refieren al orden en que se ejecutan las acciones, se consideran otras perspectivas: recursos, tiempo, etc.
- En el estado actual
  - ¿cual es la mejor acción candidata dado el objetivo seleccionado.?
  - ¿cual es el mejor recurso para ejecutar una acción.?
  - Por ejemplo asignar a1 a Mike para minimizar el tiempo restante
- Si no disponemos de modelo prescriptivo, podemos usar un modelo predictivo para simular la evolución de un indicador a partir de cada alternativa.

¿Dónde y qué aprender?

- Aprender comportamiento de agente autónomo
- Aprender a plantearse objetivos.
- Aprender modelo de proceso (dominio de planificación)
- Aprender a mejorar el propio proceso de planificación

[1]
S. Jiménez, T. De la Rosa, S. Fernández, F. Fernández, y D. Borrajo, «A review of machine learning for automated planning», *The Knowledge Engineering Review*, vol. 27, n.º 04, pp. 433–467, 2012.

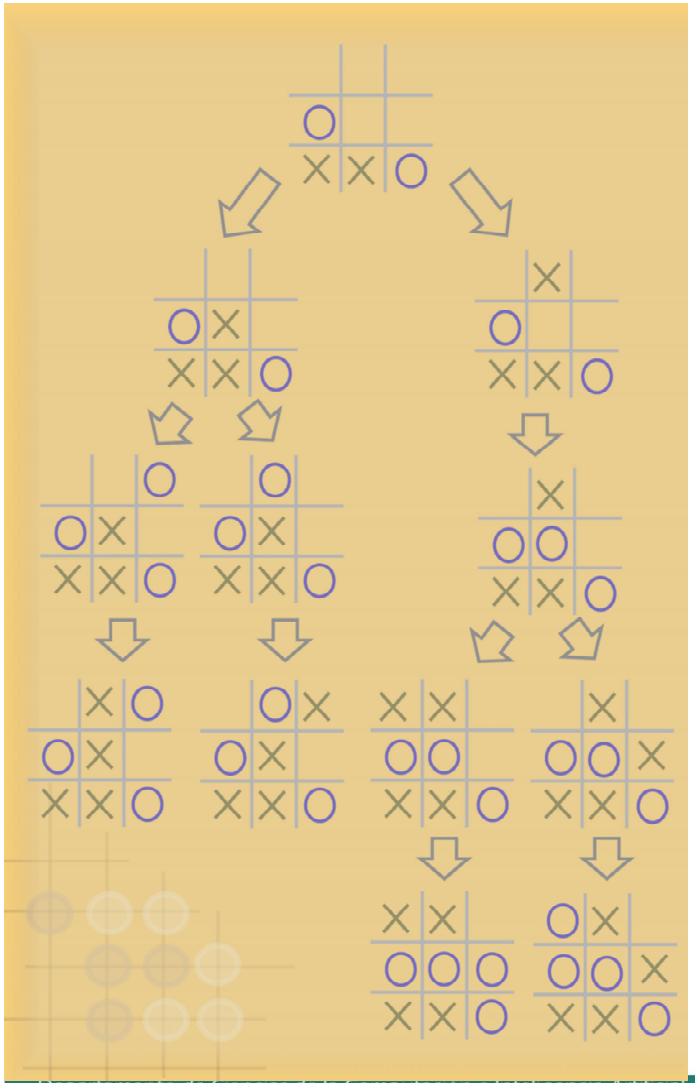
http://www.nature.com/nature/journal/v518/n7540/full/nature14236.html

- Vamos a ver un ejemplo de agente reactivo que mejora su comportamiento.
- Atención: es un ejemplo motivador para ver cómo puede aprender un agente reactivo, con habilidades cognitivas de bajo nivel (reacciona ante situaciones, no delibera).
- El tema se centra en agentes deliberativos que establecen políticas de actuación (planes) para alcanzar objetivos a largo plazo.

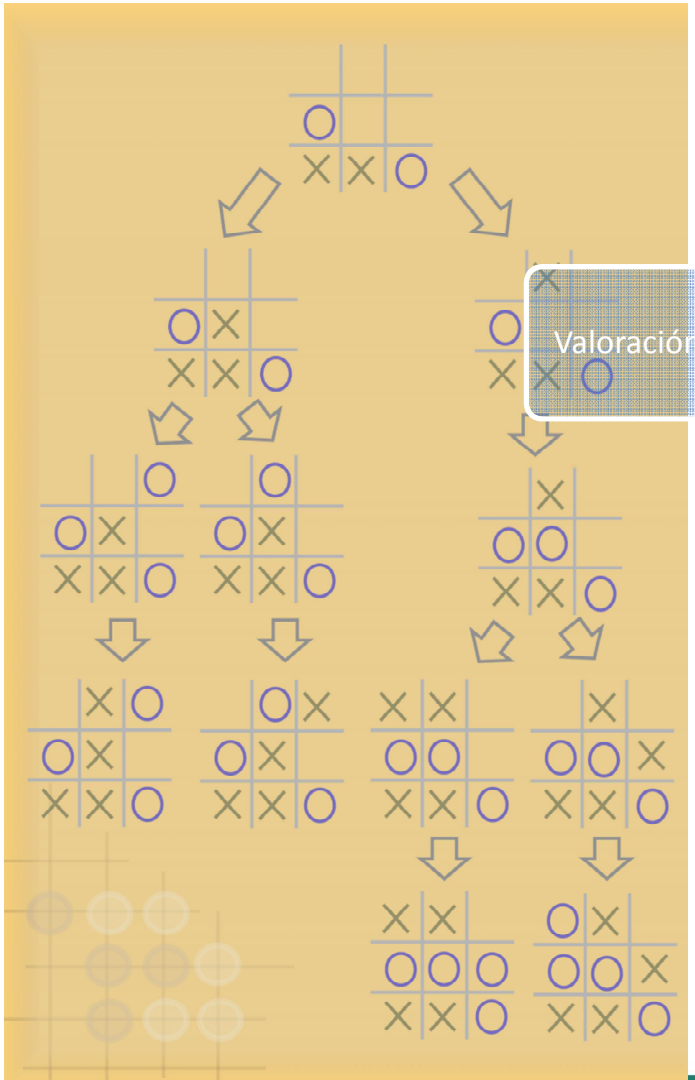
Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, et al. “Human-Level Control through Deep Reinforcement Learning.” *Nature* 518, no. 7540 (February 26, 2015): 529–33. doi:10.1038/nature14236.

- Objetivo: aprendizaje del modelo de comportamiento, mostrando rendimiento de nivel humano.
  - Empresa DeepMind, adquirida recientemente por Google.
- Inteligencia Artificial como ciencia experimental.
  - Psicología cognitiva, neurociencia, inteligencia artificial -> mejora del comportamiento animal -> estudio del comportamiento humano a nivel cognitivo.
- Pero … agente reactivo, puede afinar al máximo su reacción a partir de su experiencia previa, pero no puede planificar, anticiparse a lo que puede ocurrir.
- Nosotros nos centramos en agentes deliberativos, pero vamos a ver en un poco más detalle aprendizaje por refuerzo porque nos servirá para entender cómo un agente deliberativo aprende a auto-proponerse objetivos.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. «Mastering the Game of Go with Deep Neural Networks and Tree Search». *Nature* 529, n.º 7587 (28 de enero de 2016): 484-89. doi:10.1038/nature16961.



Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. «Mastering the Game of Go with Deep Neural Networks and Tree Search». *Nature* 529, n.º 7587 (28 de enero de 2016): 484-89. doi:10.1038/nature16961.



Valoración Estado

Búsqueda en profundidad b^d

Impracticable en ajedrez … $b \approx 35, d \approx 80$

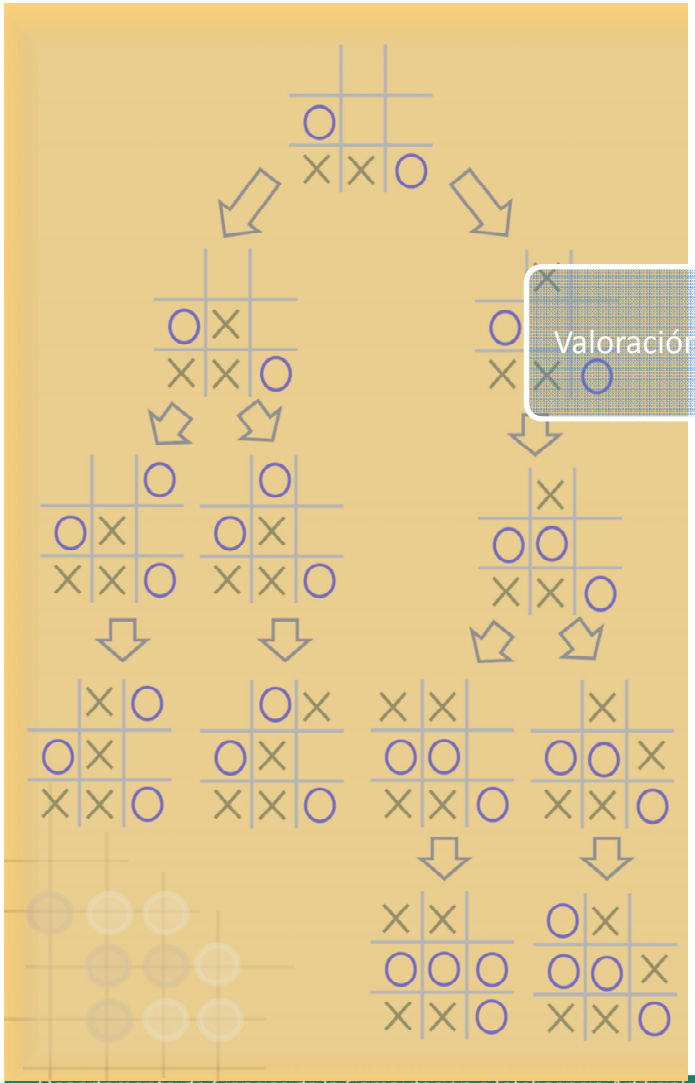
…y en Go $b \approx 250, d \approx 150$

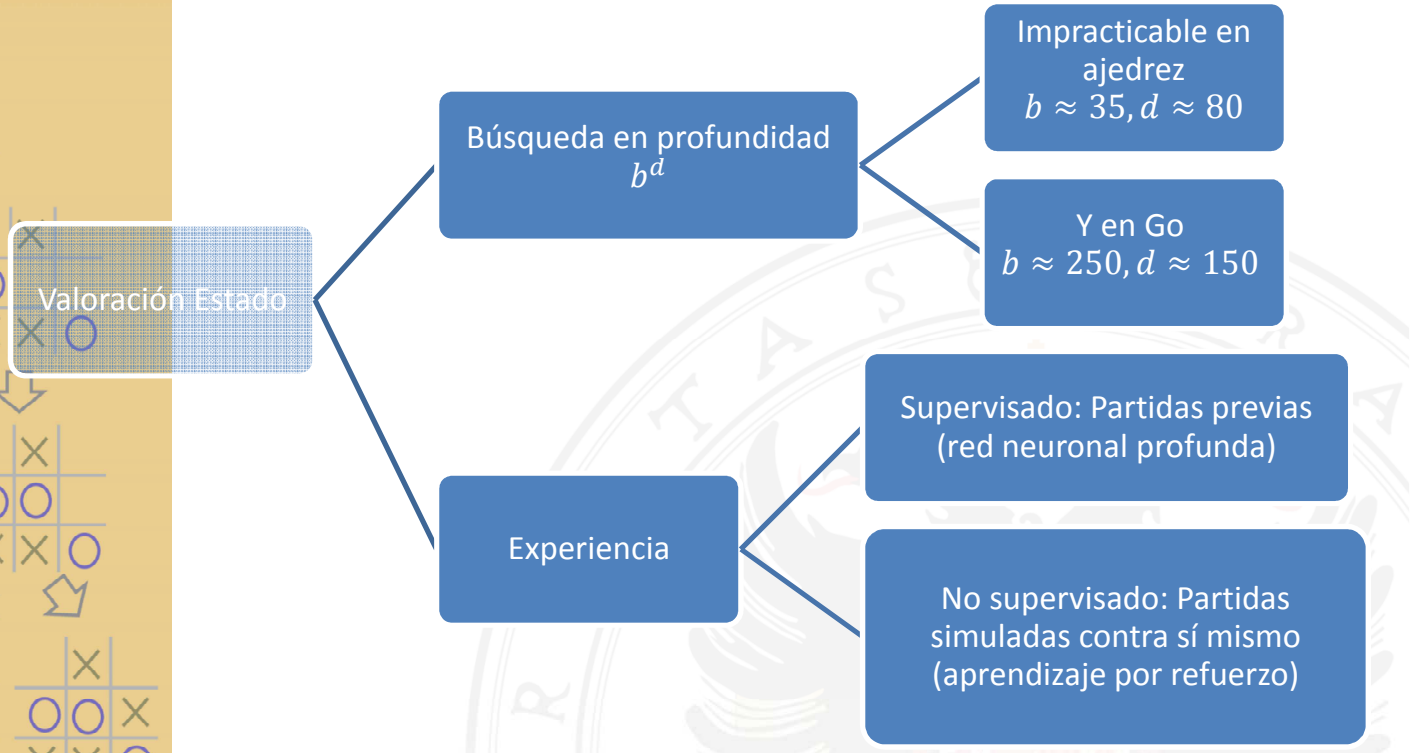
Experiencia

Supervisado: Partidas previas (red neuronal profunda)

No supervisado: Partidas simuladas contra sí mismo (aprendizaje por refuerzo)

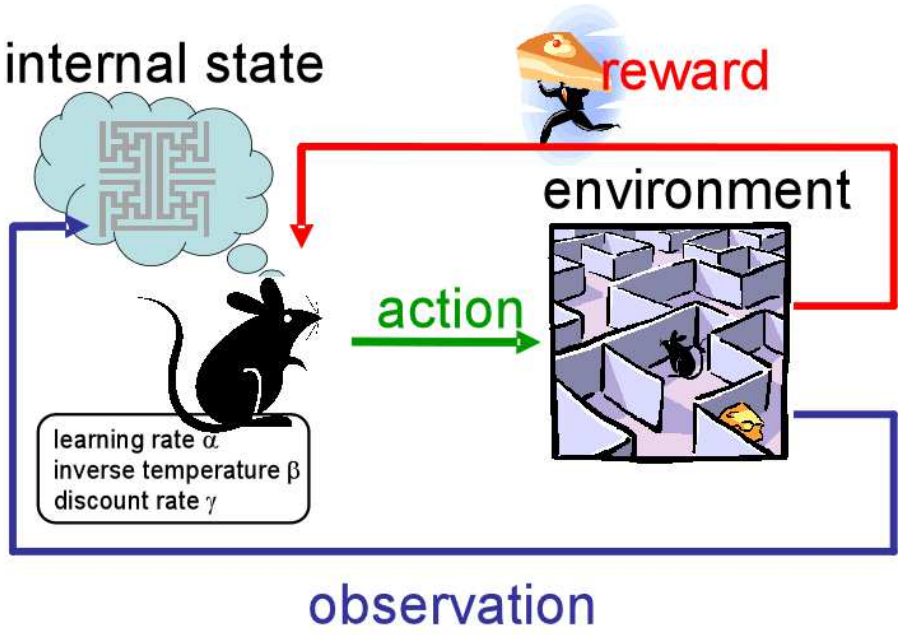
Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, et al. «Mastering the Game of Go with Deep Neural Networks and Tree Search». *Nature* 529, n.º 7587 (28 de enero de 2016): 484-89. doi:10.1038/nature16961.

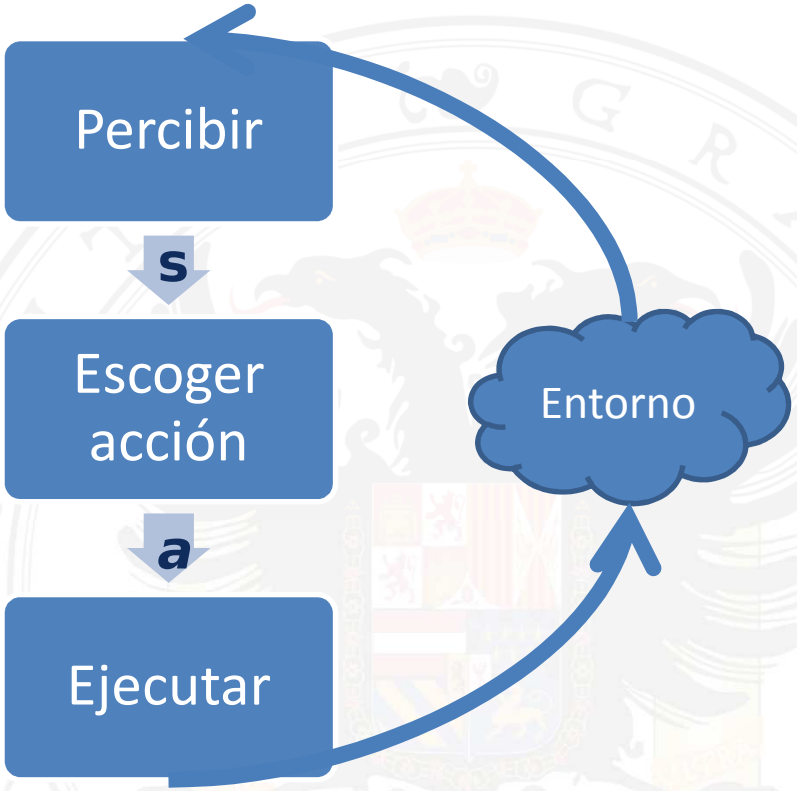




Go es modélico: toma de decisiones deliberativa,
espacio de búsqueda intratable,
solución óptima aparentemente inviable

Aprendizaje por refuerzo para un agente reactivo: determinar cuál es la acción más adecuada para cada estado posible en el mundo, recibiendo recompensas (premio o castigo) por el comportamiento realizado.





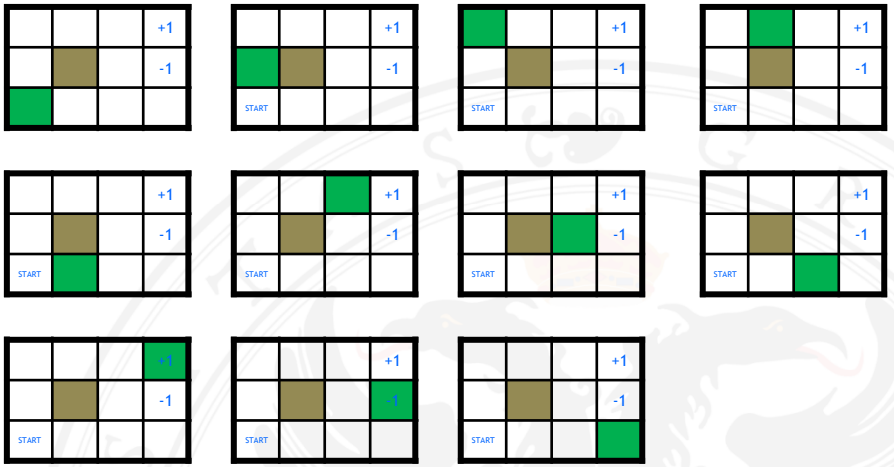
Ejemplo: un robot en mundo cuadriculado que sólo puede ejecutar 4 acciones. Sólo se pueden percibir estados y **no existe relación conocida entre estado y acción**. Pero sí se conocen las recompensas.

Recompensas

-0.04	-0.04	-0.04	Premio +1
-0.04	Muro	-0.04	Castigo -1
START	-0.04	-0.04	-0.04

Premio si llega a [4,3], Castigo si llega a [4,2] y pequeña penalización por cada paso

Estados

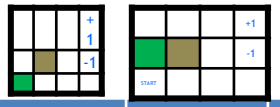


Acciones: UP, DOWN, LEFT, RIGHT

[1]
R. S. Sutton y A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Mass: A Bradford Book, 1998

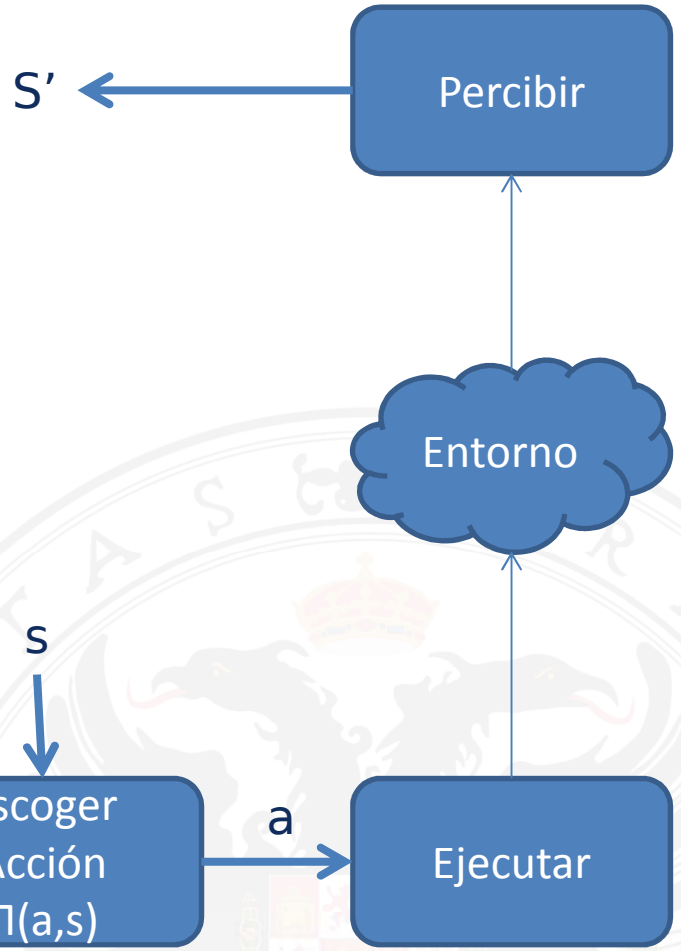
2ª edición descargable en pdf (incompleta)

- Objetivo:
  - Determinar una **política de actuación->** ¿qué acción selecciono en cada instante para ejecutar?
  - Política de actuación: una función **Π(s):S->A** que asocia cada estado con una acción (representada por la tabla de abajo)
  - **Utilidad:** la aplicabilidad de una acción *a* en un estado *s* tiene asociado un valor de utilidad Q(a,s).
  - Se trata de encontrar la política que maximiza la utilidad.



	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
UP	Q(1,1)	Q(1,2)	Q(1,3)								
RIGHT	Q(2,1)										
LEFT	Q(3,1)										
DOWN	Q(4,1)										

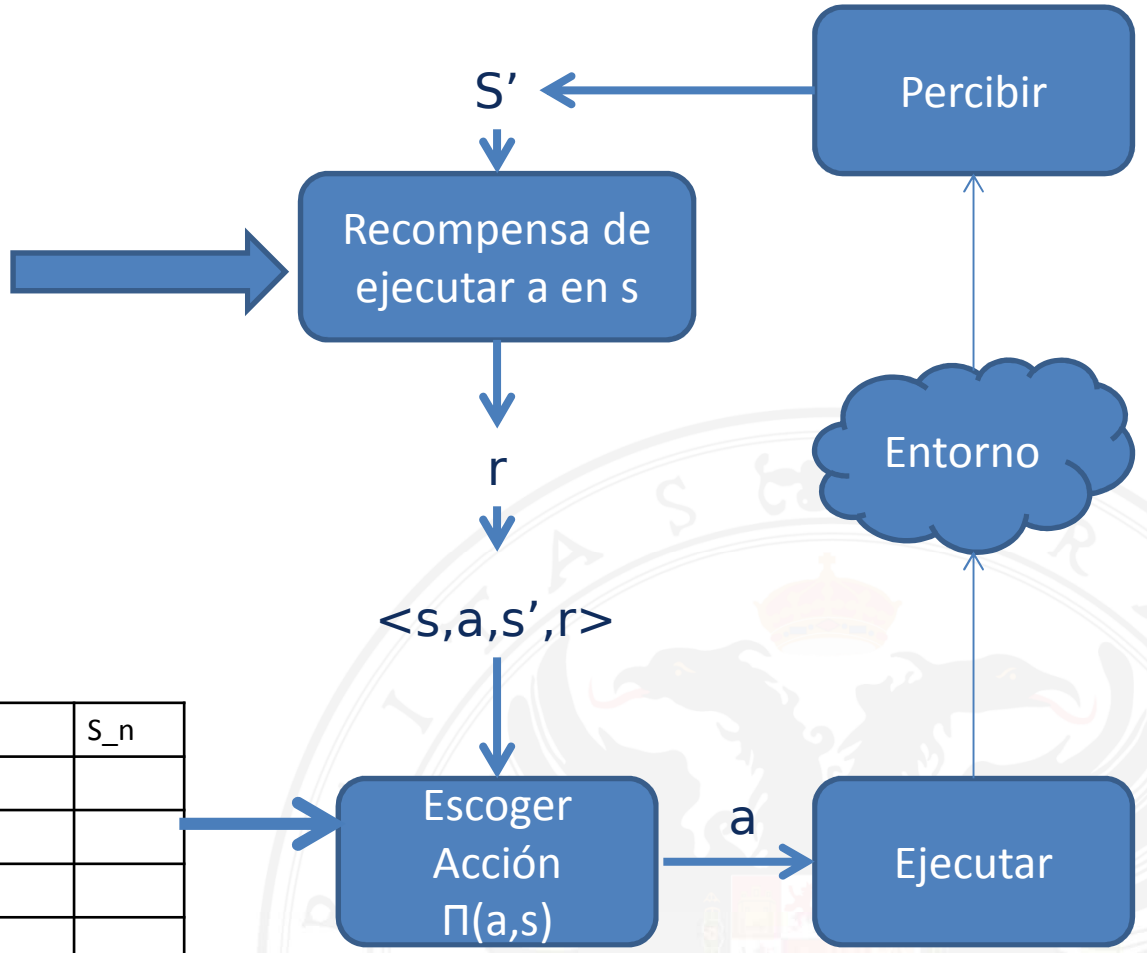
El aprendizaje por refuerzo **sólo es aplicable si se apoya en la ejecución** en el mundo real o en la ejecución en una simulación para obtener el estado resultante de ejecutar acciones.



Política Π(a,s)

	S1	…	S	…	S'	…	S_n
A1							
…							
A			Q(a,s)				
…							
A_n							

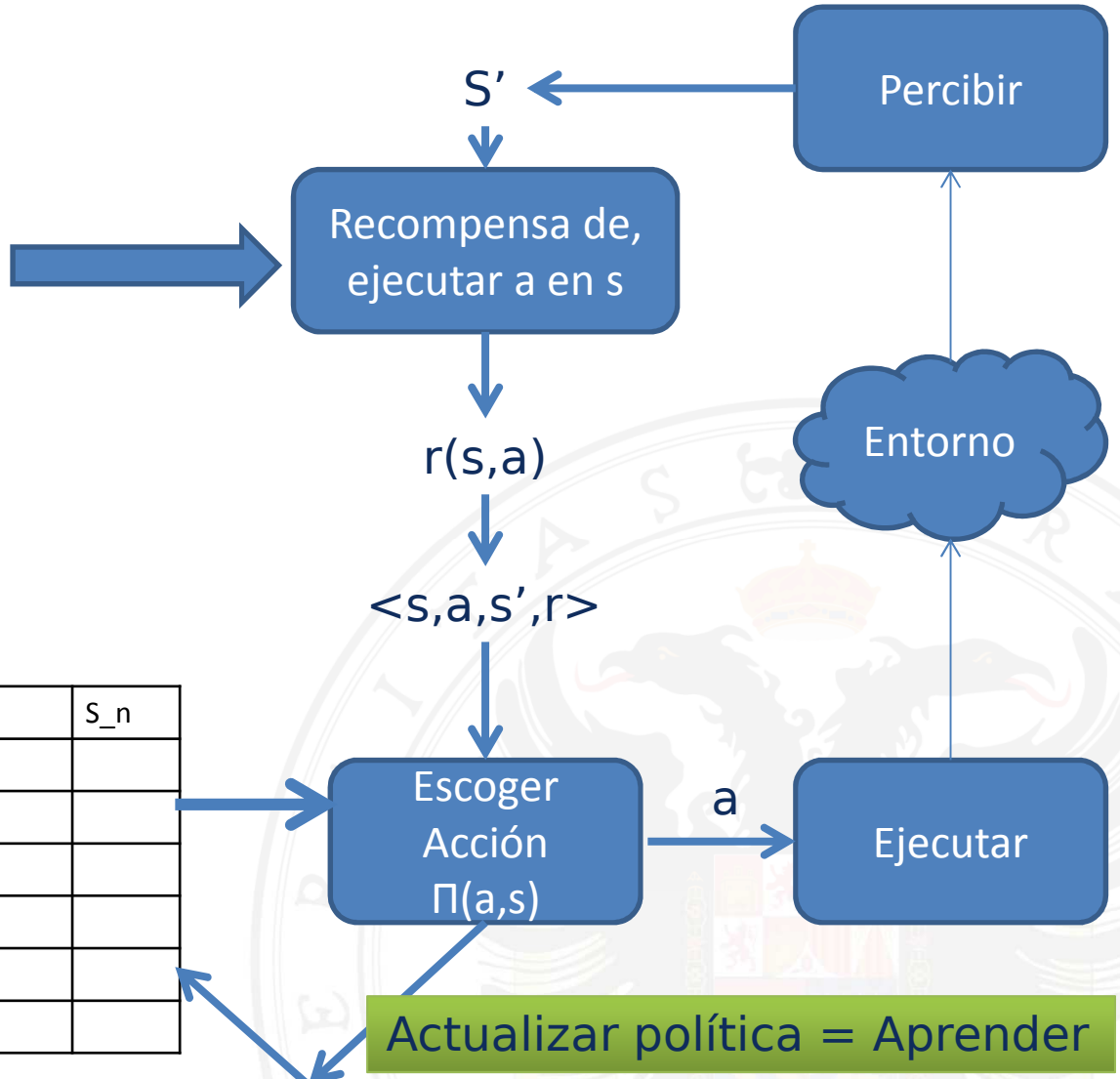
-0.04	-0.04	-0.04	+1
-0.04		-0.04	-1
START	-0.04	-0.04	-0.04



Política Π(a,s)

	S1	…	S	…	S'	…	S_n
A1							
…							
A			Q(a,s)				
…							
A_n							

-0.04	-0.04	-0.04	+1
-0.04		-0.04	-1
START	-0.04	-0.04	-0.04

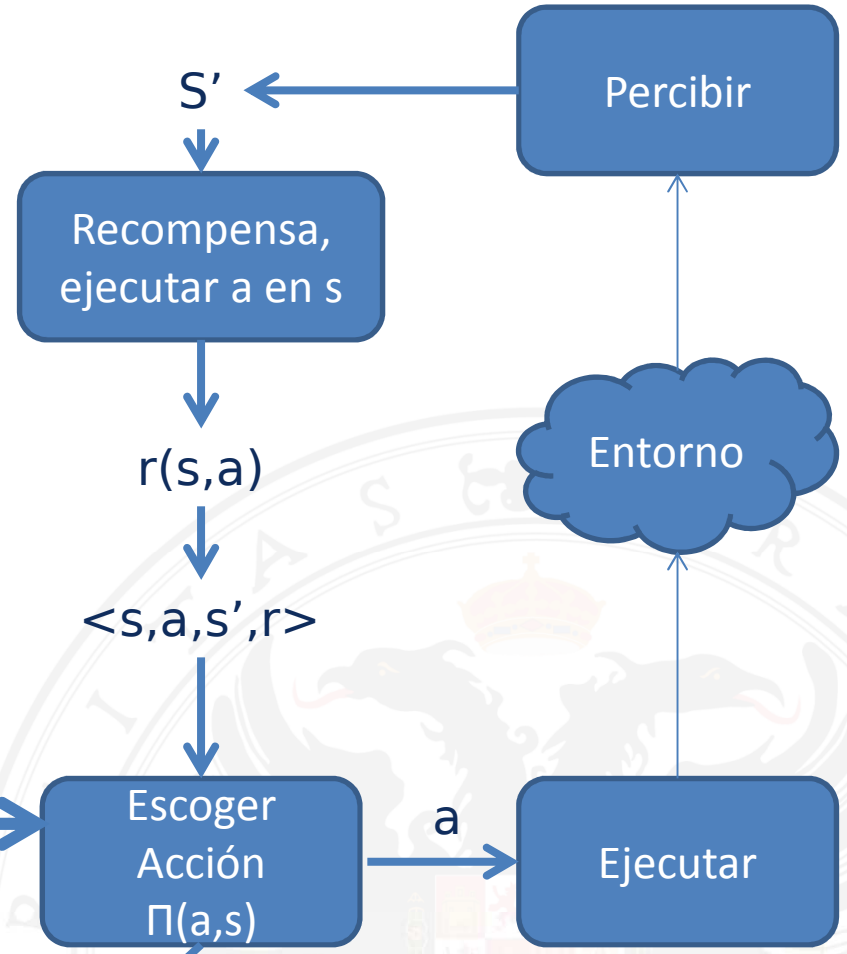


Política Π(a,s)

	S1	…	S	…	S'	…	S_n
A1					Q'1		
…					Q'2		
A			Q				
…					Q'n-1		
A_n					Q'n		

$$Q_{t+1}(s,a) := Q_t(s,a) + \alpha(r(s,a) + \gamma \max_{a'} Q(s',a') - Q_t(s,a))$$

- Alpha: Tasa de aprendizaje
  - Alpha = 0, no aprende nada
  - Alpha = 1, sólo más reciente información
- Gamma: factor de descuento (importancia del futuro)
  - Gamma = 0, greedy: solo considera recompensas locales
  - Gamma = 1, se esmera por recompensas a largo plazo

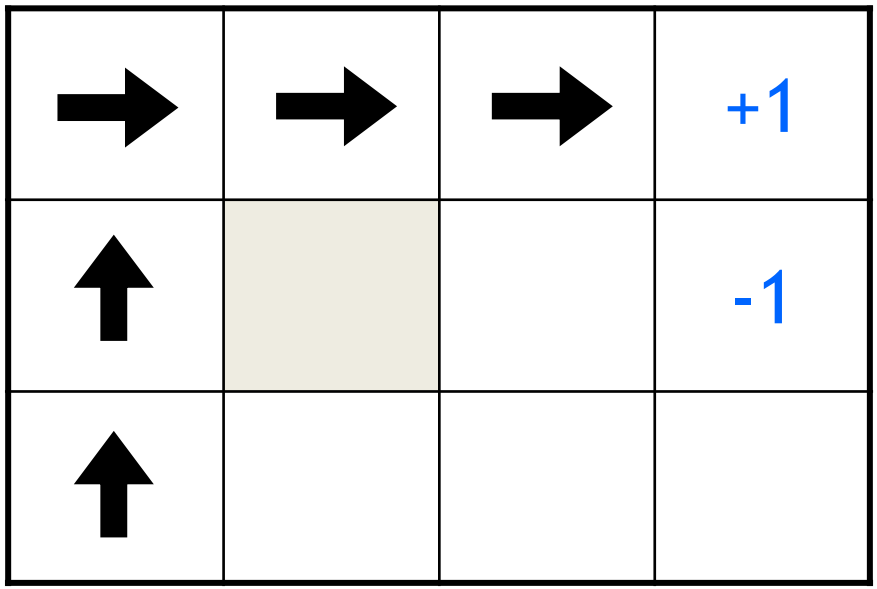


Política Π(a,s)

	S1	…	S	…	S'	…	S_n
A1					Q'1		
…					Q'2		
A			Q				
…					Q'n-1		
A_n					Q'n		

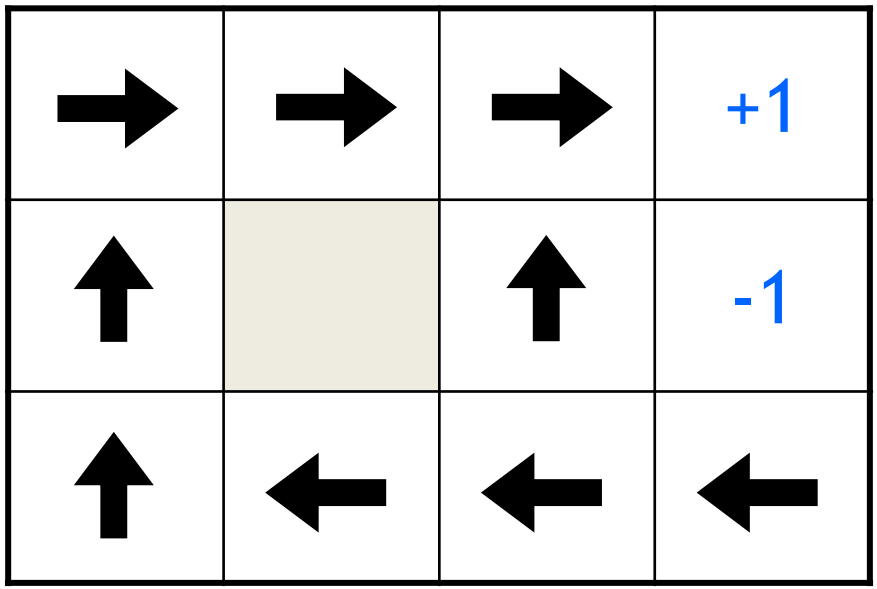
$$Q_{t+1}(s,a) := Q_t(s,a) + \alpha(r(s,a) + \gamma \max_{a'} Q(s',a') - Q_t(s,a))$$

(... de presentación de Peter Bodik, y las 6 siguientes)

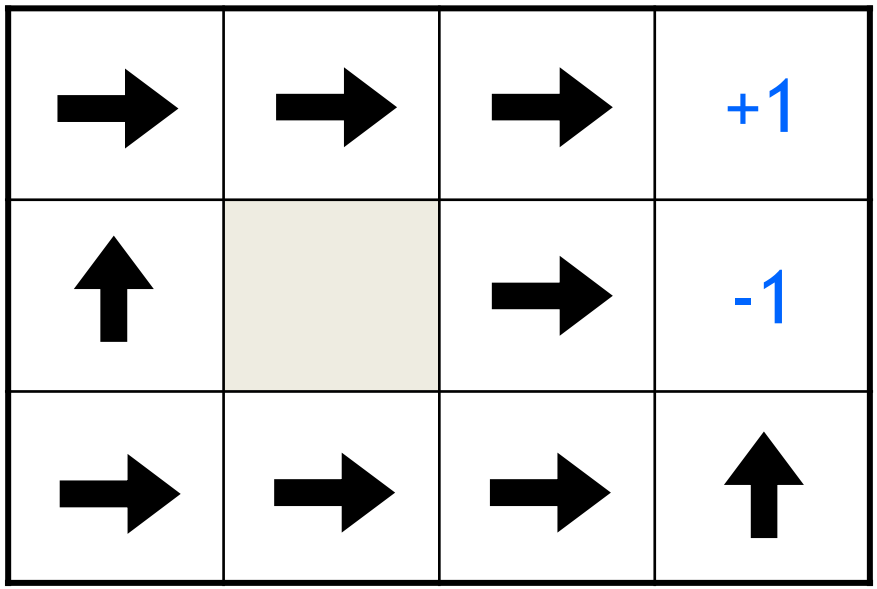


- only if actions deterministic
  - not in this case (actions are stochastic)
- solution/policy
  - mapping from each state to an action

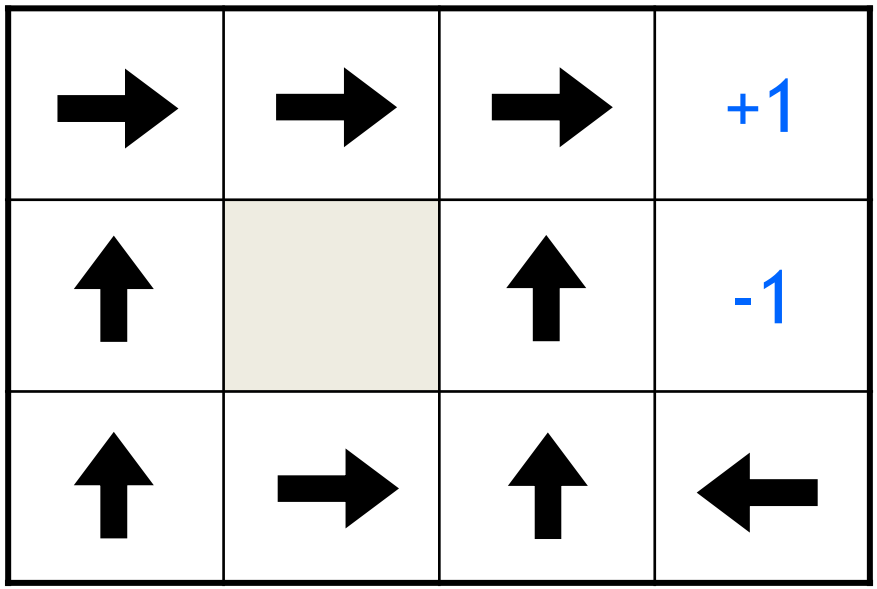
Optimal policy



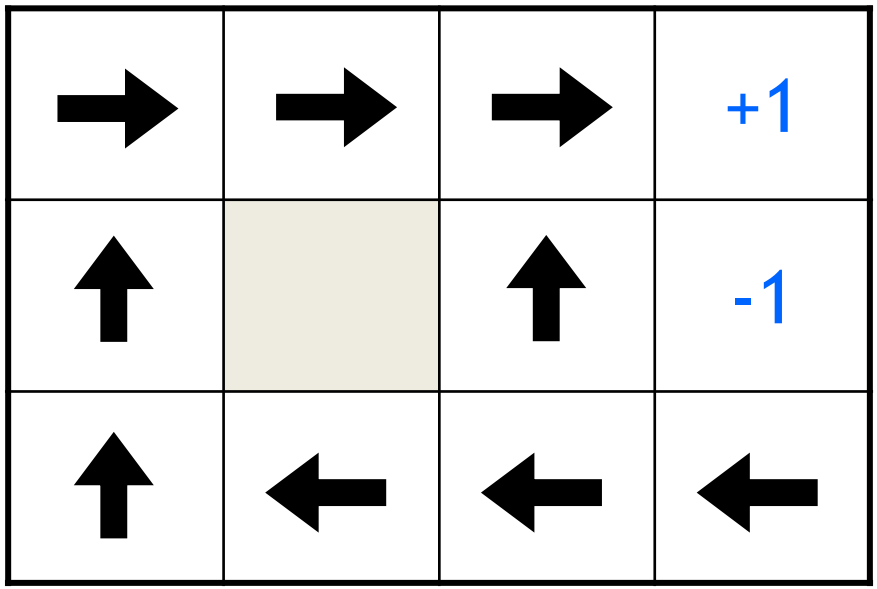
Reward for each step -2



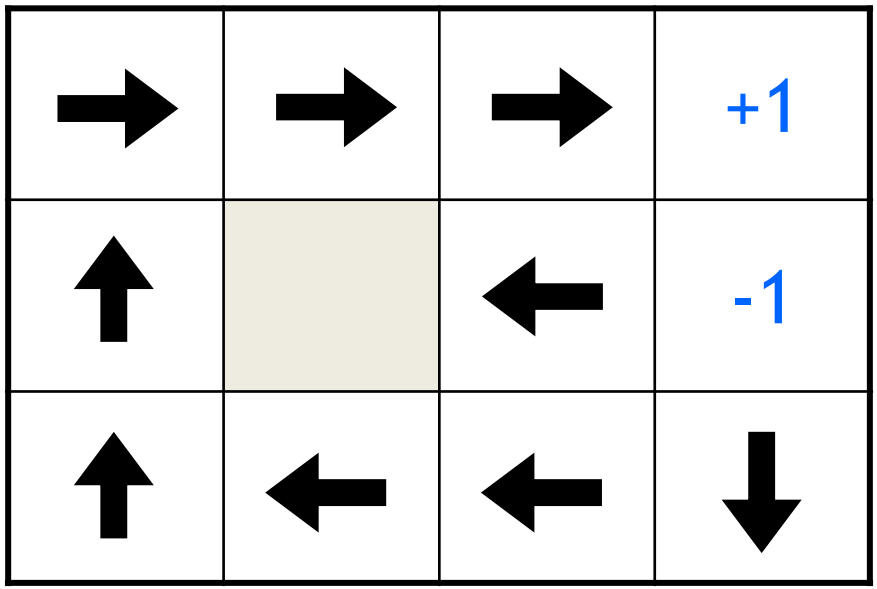
Reward for each step: -0.1

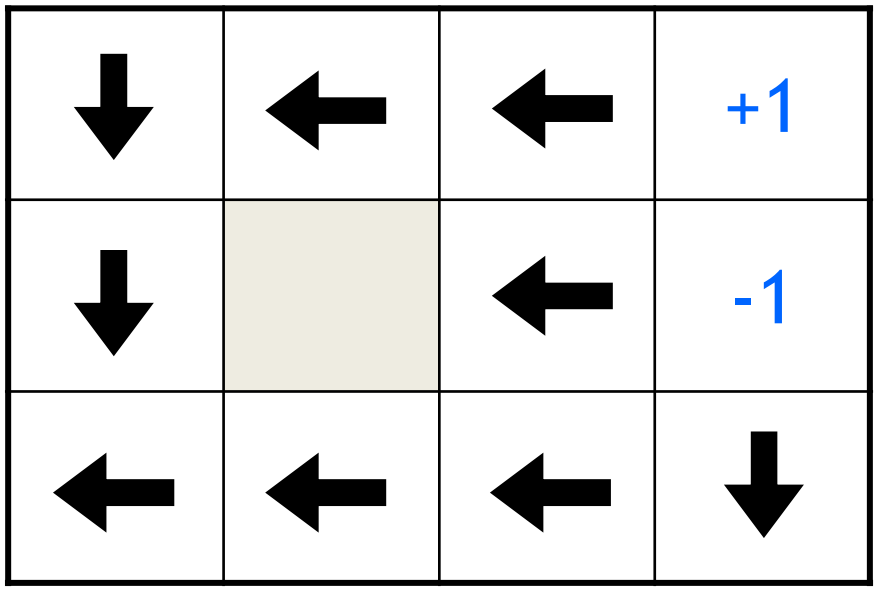


Reward for each step: -0.04



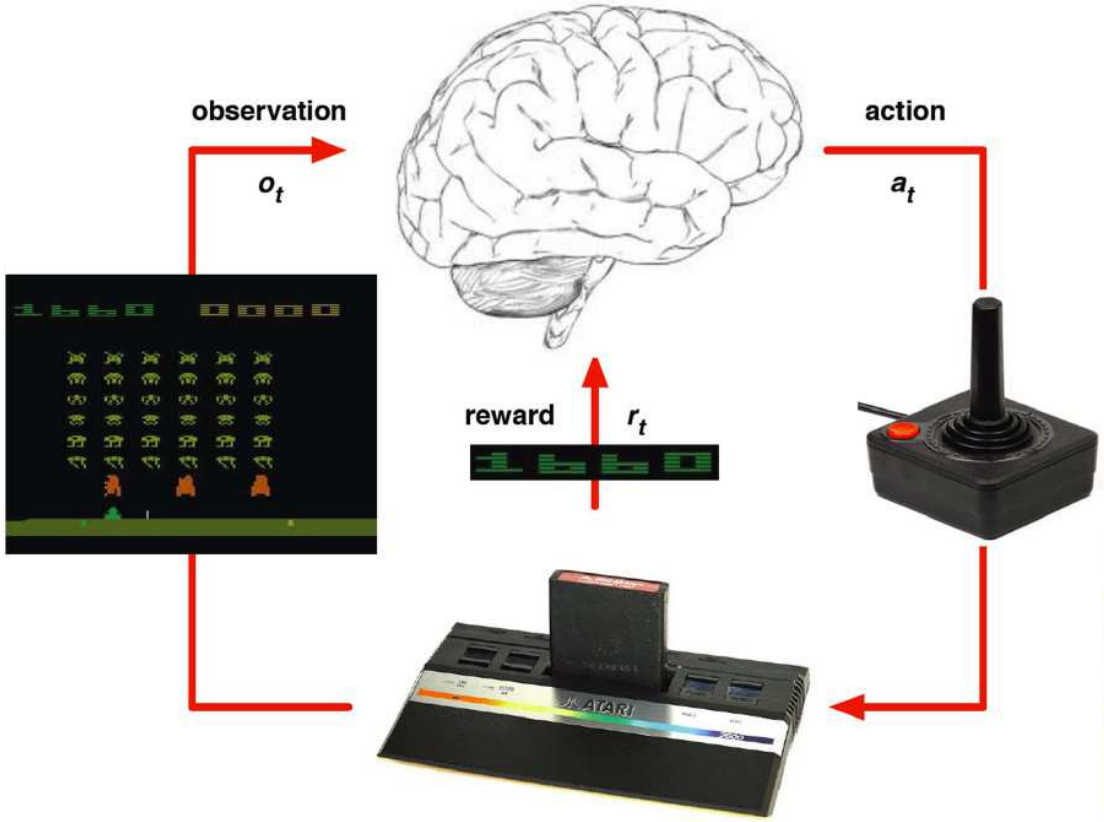
Reward for each step: -0.01





Un agente reactivo se preocupa de decidir qué acción ejecutar para un estado actual y extrae conclusiones de la observación del estado posterior.

Un agente deliberativo establece proyecciones de acciones (planes) para alcanzar un estado “más alejado en el tiempo” . Los planes incorporan trayectorias de estados, los planes pueden fallar, hay que plantearse qué hacer cuando hay un fallo,... ¿cómo afrontar la autonomía total de un agente deliberativo?



David Silver

- Aprendizaje de objetivos en planificación
- Aprendizaje de dominios de planificación

Aprender objetivos de planificación

- Planificación contínua
- Autonomía dirigida por objetivos (Goal Driven Autonomy)
  - Plantearse objetivos ante excepciones.
    - Eventos previstos
    - Eventos imprevistos
- Aprendizaje por refuerzo para aprender a seleccionar objetivos.

Aprender a plantearse objetivos por sí mismo

Ciclo de planificación contínua (recuerdo)

- Adaptive Case Management:
  - Procesos Dinámicos, Flexibles y Adaptativos.
  - Procesos que dependen de una situación
  - que pueden cambiarse mientras se usan
  - como respuesta a cambios de situaciones (producidos por eventos externos)
  - se adaptan a la nueva situación
  - En un ciclo de planificación contínua.
- Ya sabemos que con la planificación automática se pueden cubrir estas necesidades.
- ¿Podemos incrementar la autonomía?

- Autonomía dirigida por objetivos (Goal Driven Autonomy)
  - Un planificador está integrado en un ciclo de planificación contínua en el que cada vez que hay un fallo en la ejecución el sistema se plantea cómo resolver el fallo auto-proponiéndose nuevos objetivos.
  - A continuación, en más detalle.

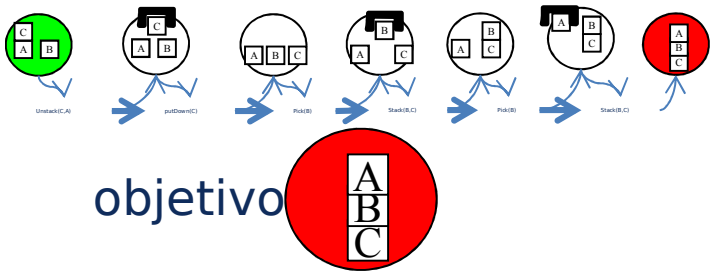
[1]
M. Klenk, M. Molineaux, y D. W. Aha, «Goal-Driven Autonomy For Responding To Unexpected Events In Strategy Simulations», *Computational Intelligence*, vol. 29, n.º 2, pp. 187–206, 2013.

[1]
N. Hawes, «A survey of motivation frameworks for intelligent systems», *Artificial Intelligence*, vol. 175, n.º 5-6, pp. 1020-1036, abr. 2011.

Aprender a plantearse objetivos por sí mismo

Autonomía Dirigida por Objetivos

El usuario establece los objetivos y los envía al planificador que, a partir de la situación actual genera un plan adaptado a la situación.



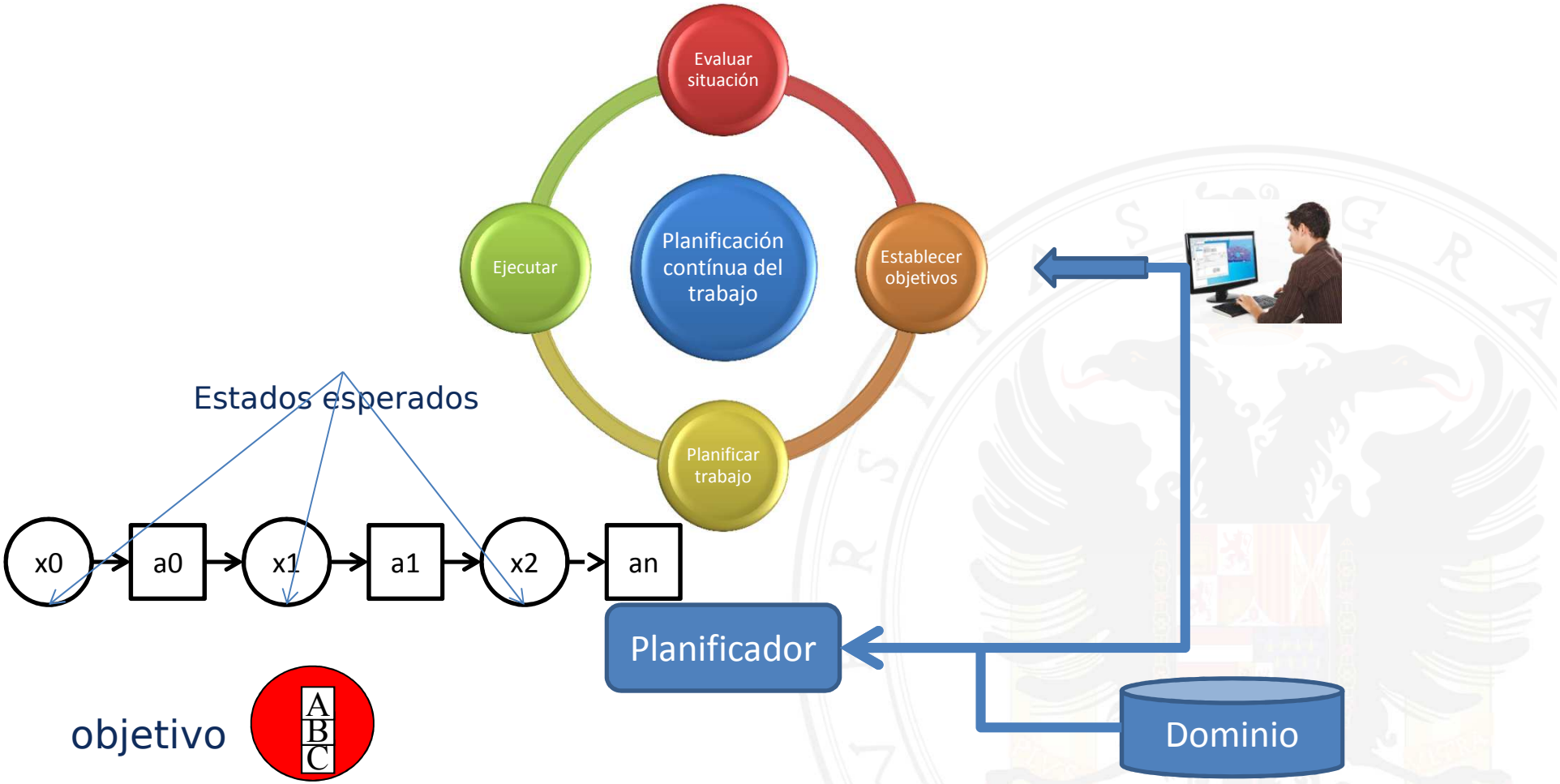
Planificador

Dominio

Aprender a plantearse objetivos por sí mismo

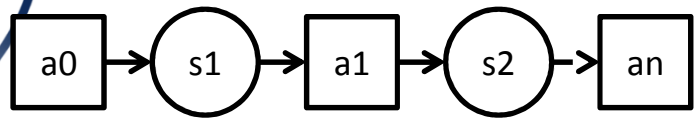
Autonomía Dirigida por Objetivos

El plan es una secuencia de acciones junto a un estado esperado previo y otro posterior para cada acción.



Aprender a plantearse objetivos por sí mismo

Autonomía Dirigida por Objetivos

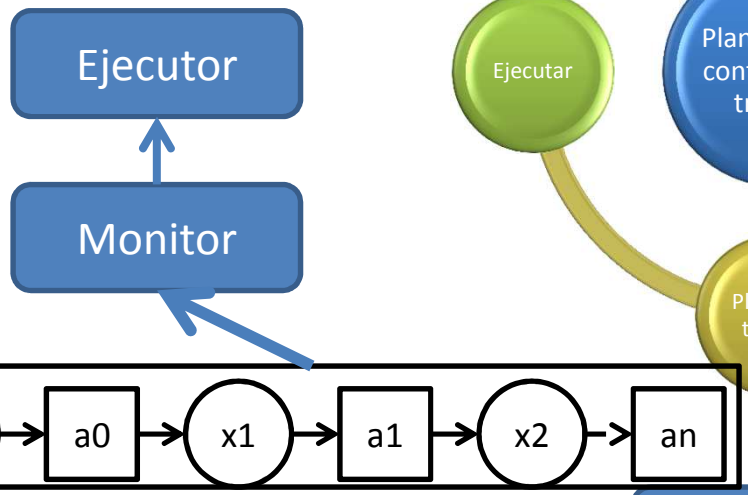


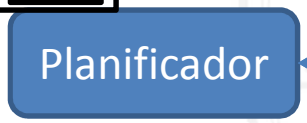
El proceso de monitorización selecciona la siguiente acción a ejecutar del plan.

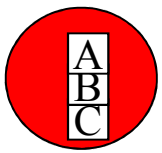
El proceso de ejecución la hace efectiva en el entorno

Se percibe el nuevo estado observado

El plan realmente ejecutado se registra en un log.

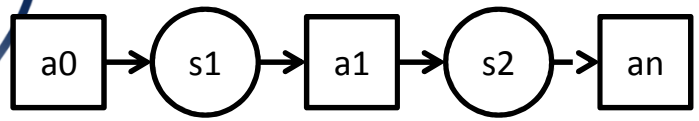


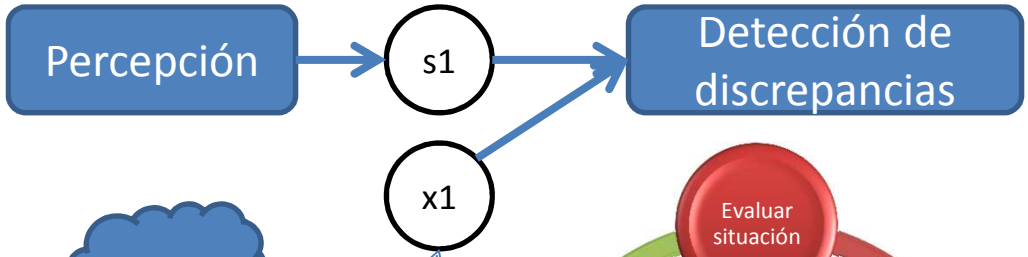


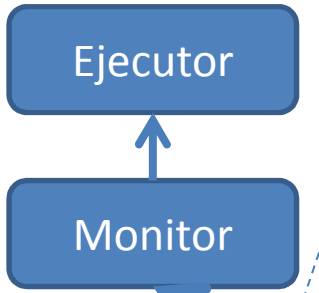


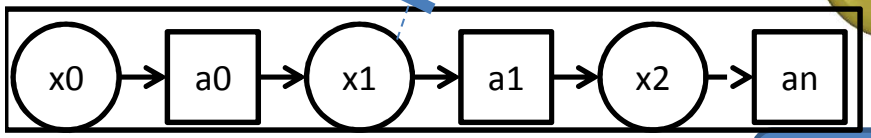
Aprender a plantearse objetivos por sí mismo

Autonomía Dirigida por Objetivos





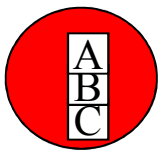




Evaluación de la situación:

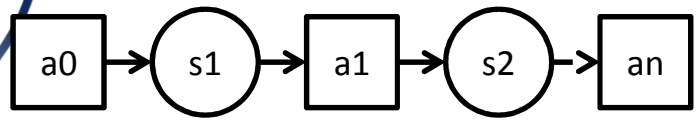
Un proceso de detección de discrepancias determina si el estado actual es discrepante con el estado esperado.

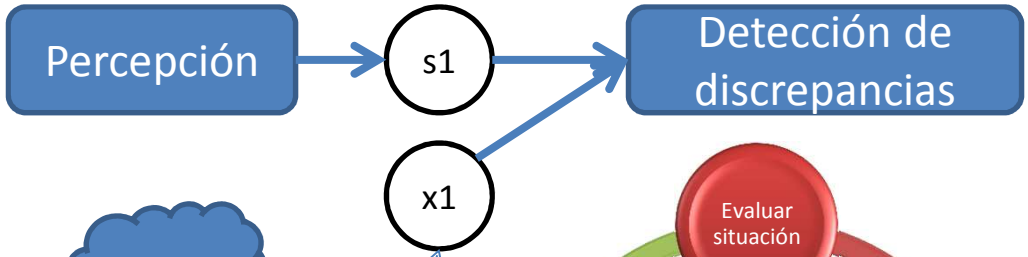
Si no hay discrepancia se pasa el control a monitorización para escoger la siguiente acción a ejecutar

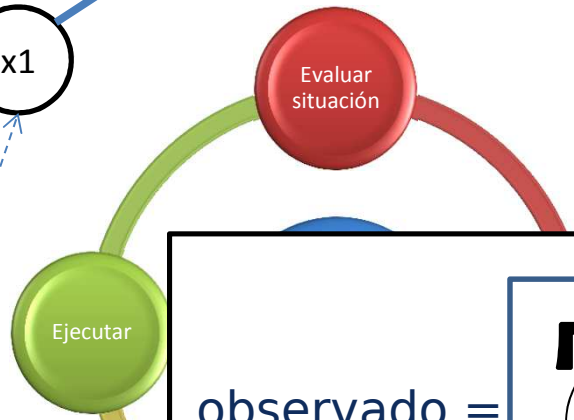


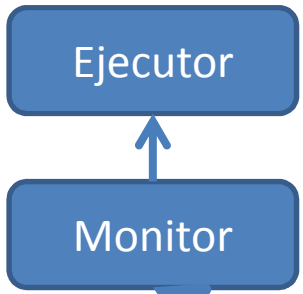
Aprender a plantearse objetivos por sí mismo

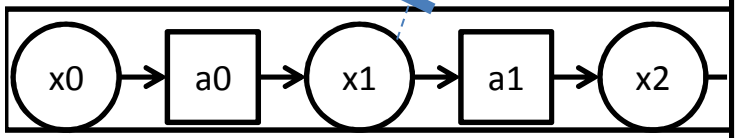
Autonomía Dirigida por Objetivos

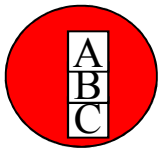








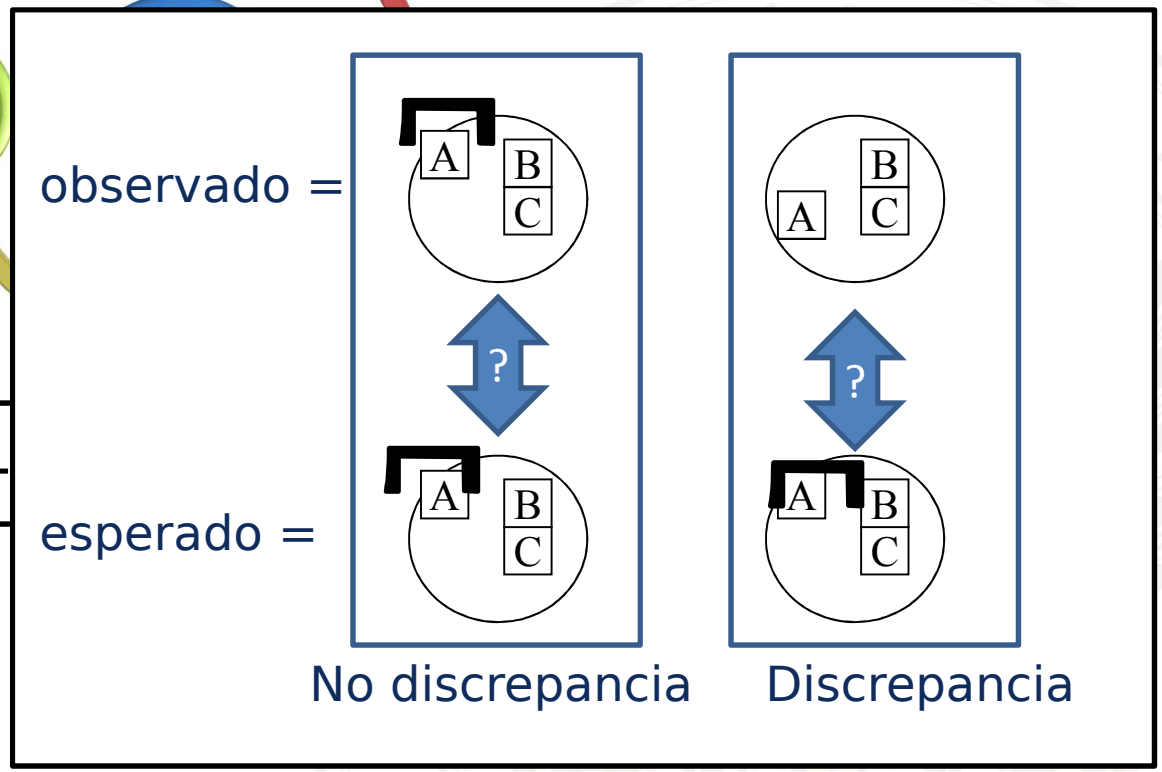




Evaluación de la situación:

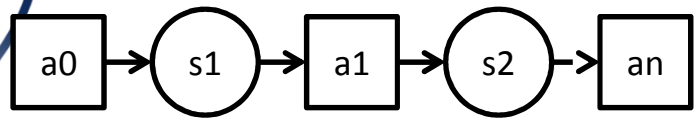
Un proceso de detección de discrepancias determina si el estado actual es discrepante con el estado esperado.

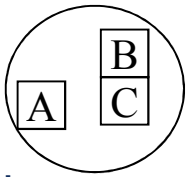
Si no hay discrepancia se pasa el control a monitorización para escoger la siguiente acción a ejecutar

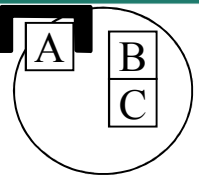


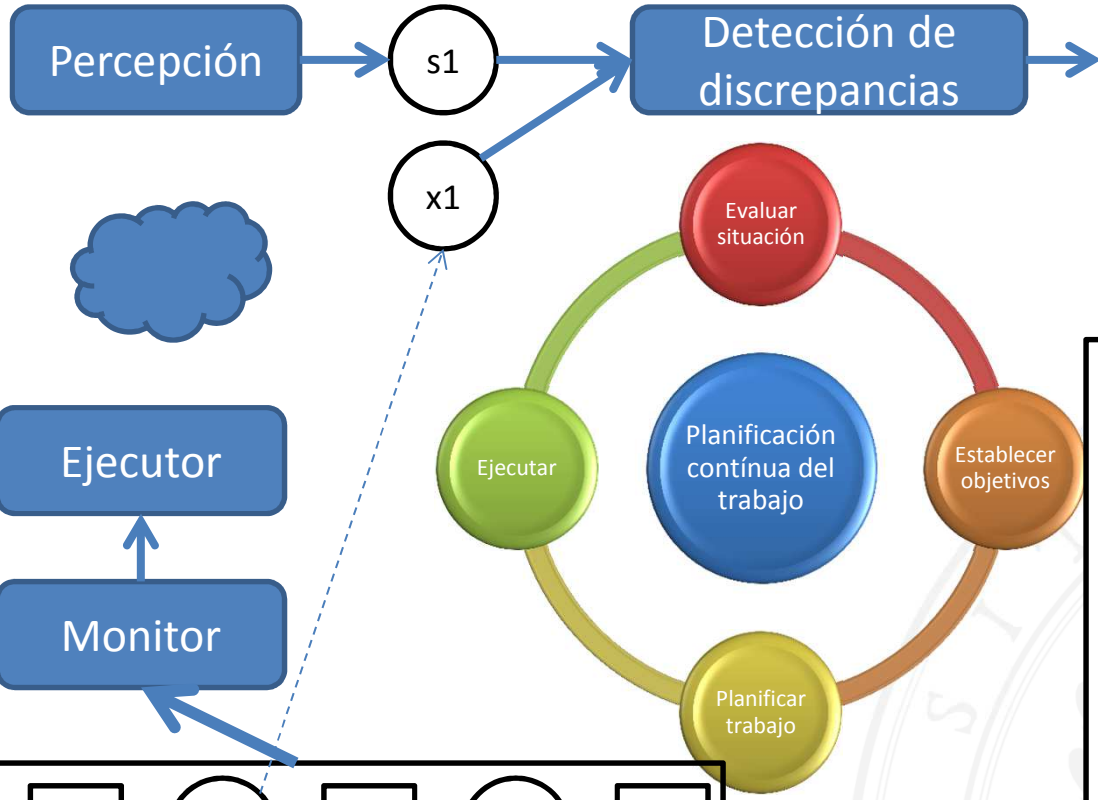
Aprender a plantearse objetivos por sí mismo

Autonomía Dirigida por Objetivos





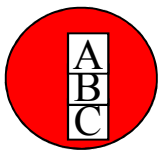




Evaluación de la situación:

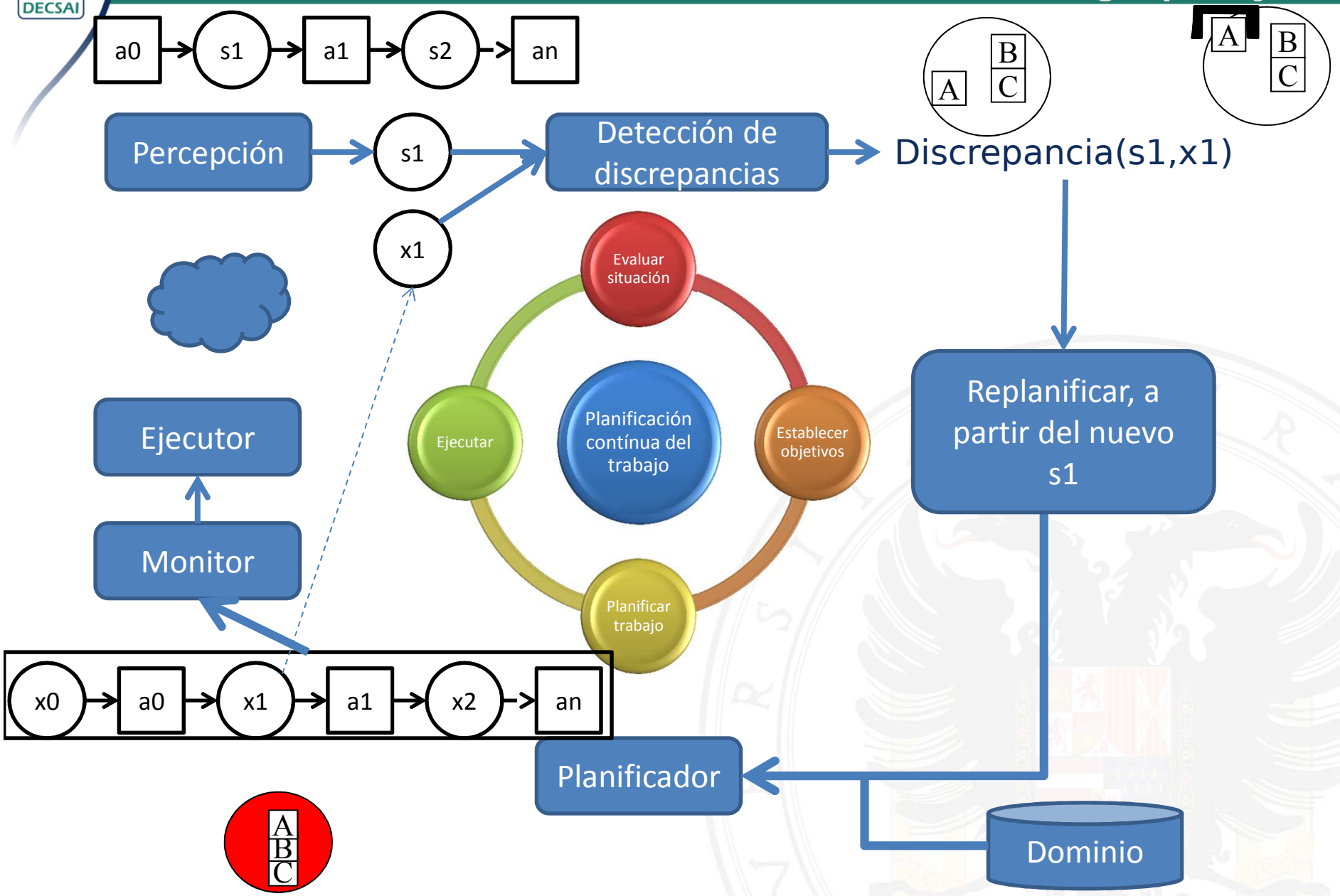
Si hay discrepancia se envía el estado esperado y el observado a un módulo de formulación de objetivos.

Si la discrepancia se debe a una excepción prevista esta etapa se puede obviar, porque hay conocimiento en el dominio para resolverla



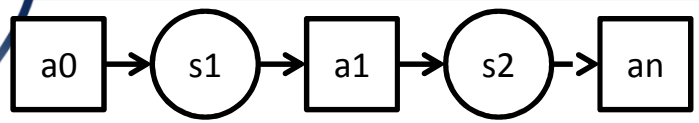
Aprender a plantearse objetivos por sí mismo

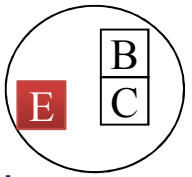
Autonomía Dirigida por Objetivos

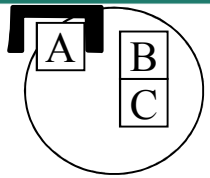


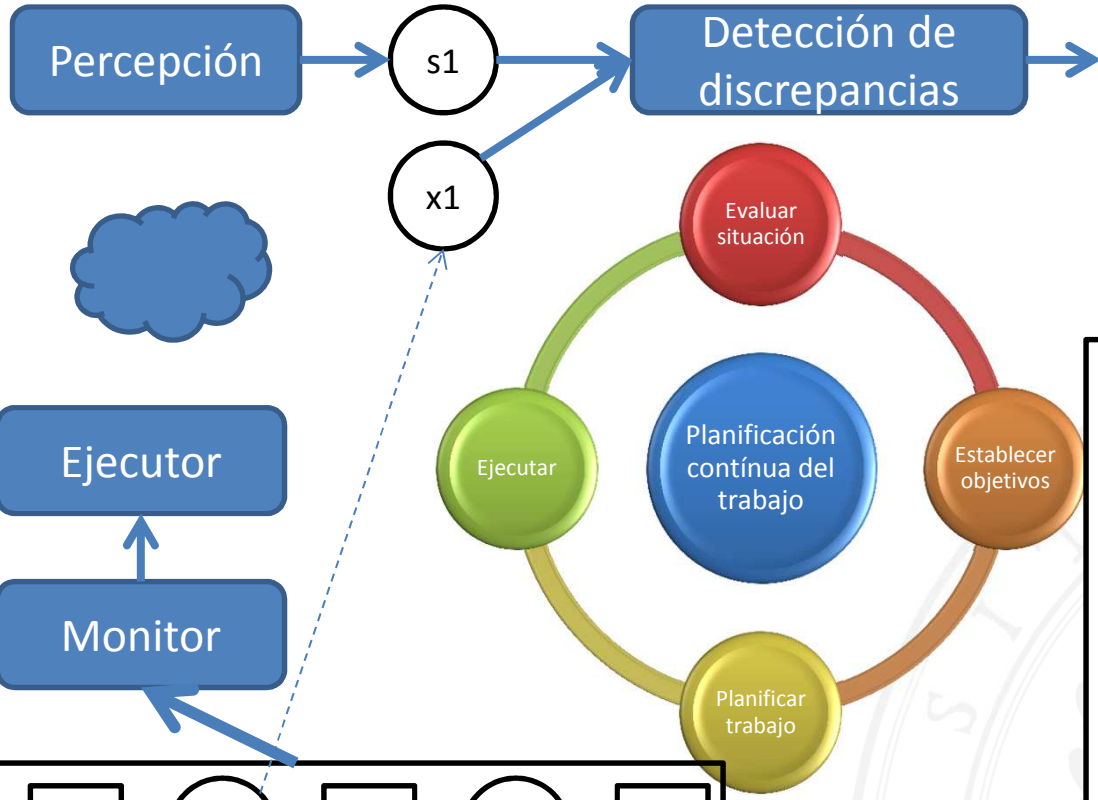
Aprender a plantearse objetivos por sí mismo

Autonomía Dirigida por Objetivos





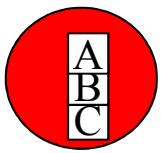




Evaluación de la situación:

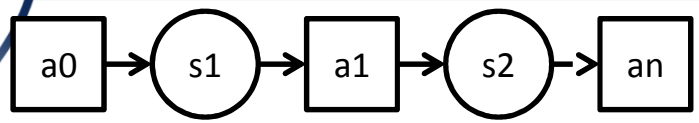
Si hay discrepancia se envía el estado esperado y el observado a un módulo de formulación de objetivos.

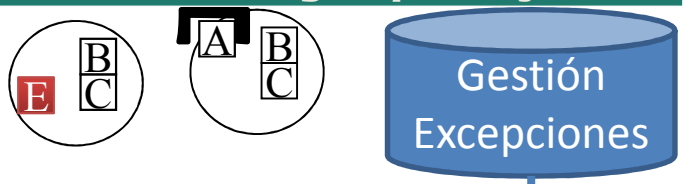
*Si la discrepancia se debe a una **excepción no prevista** es necesario determinar un objetivo nuevo para gestionar la excepción.*

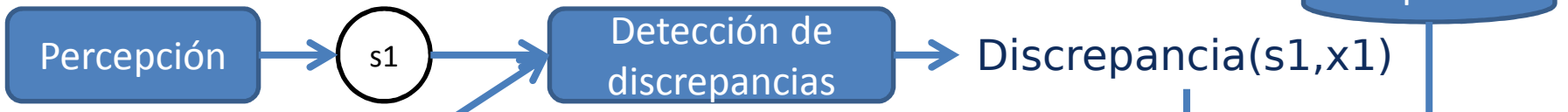


Aprender a plantearse objetivos por sí mismo

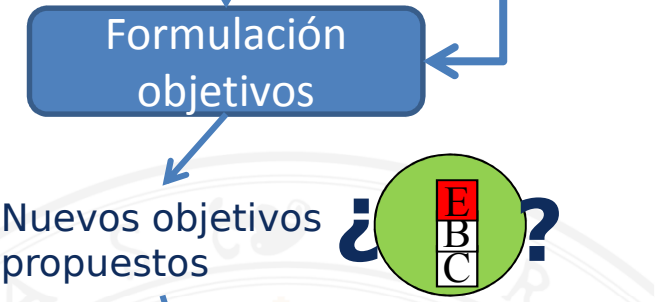
Autonomía Dirigida por Objetivos

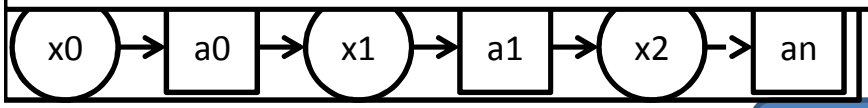


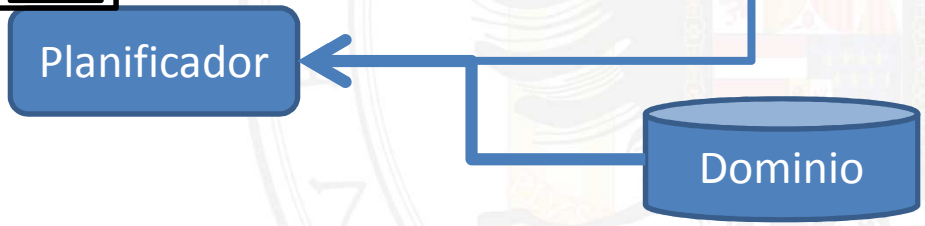




- Formulación de objetivos:
- Se apoya en una base de casos en la que se asocian situaciones con objetivos.
  - S->Objetivo
- La formulación de objetivos envía los objetivos formulados al proceso de gestión de objetivos.
- ¿Cómo se añaden?¿Qué previos se eliminan?¿Se mezclan?
- Interacción con usuario
- O Autonomía total

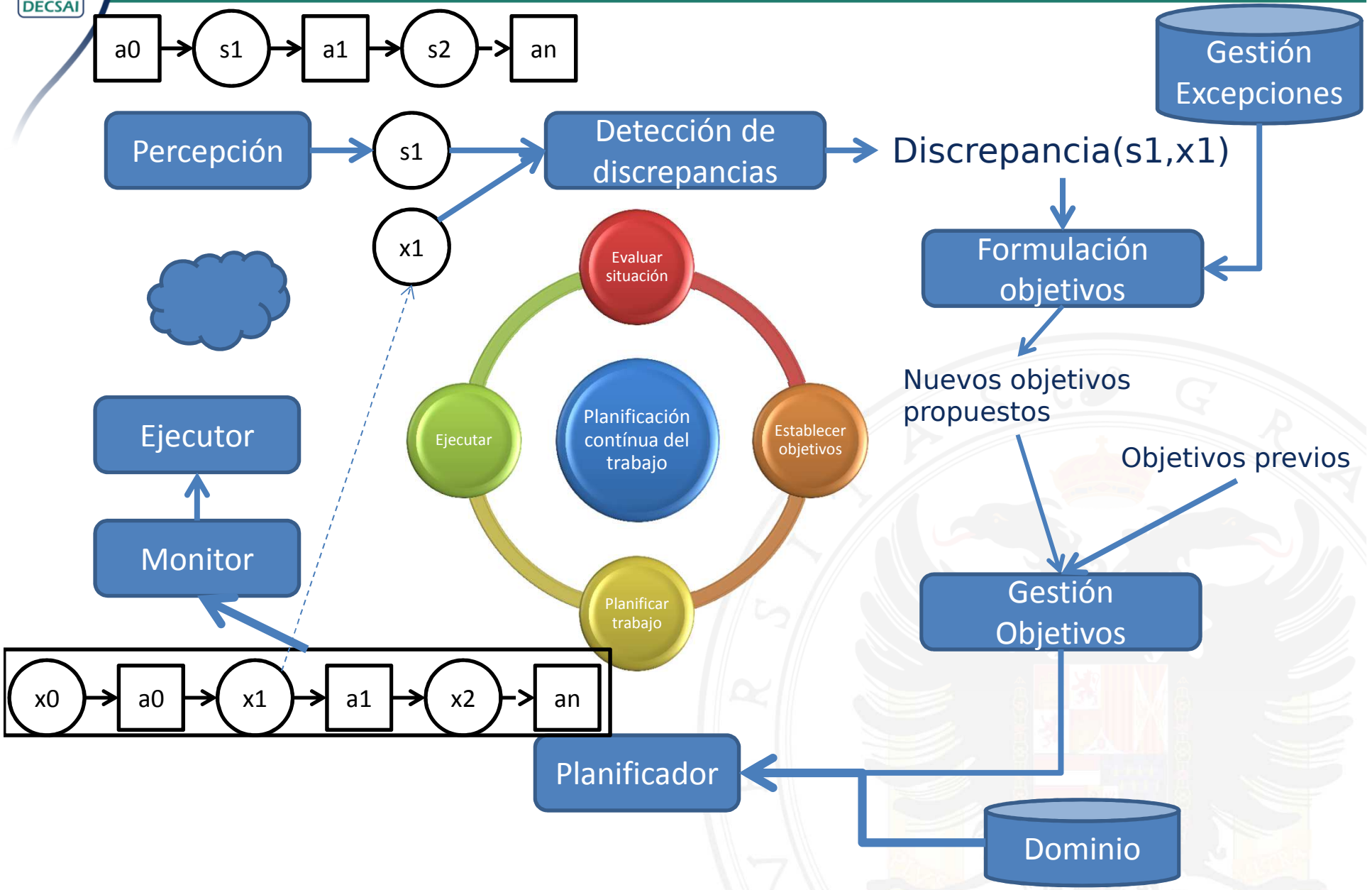






Aprender a plantearse objetivos por sí mismo

Autonomía Dirigida por Objetivos



- ¿Cómo aprender a plantearse nuevos objetivos?
  - Supervisado: aprendizaje activo mediante la interacción con el usuario

[1]
J. Powell, M. Molineaux, y D. W. Aha, «Active and Interactive Discovery of Goal Selection Knowledge.», en *FLAIRS Conference*, 2011.

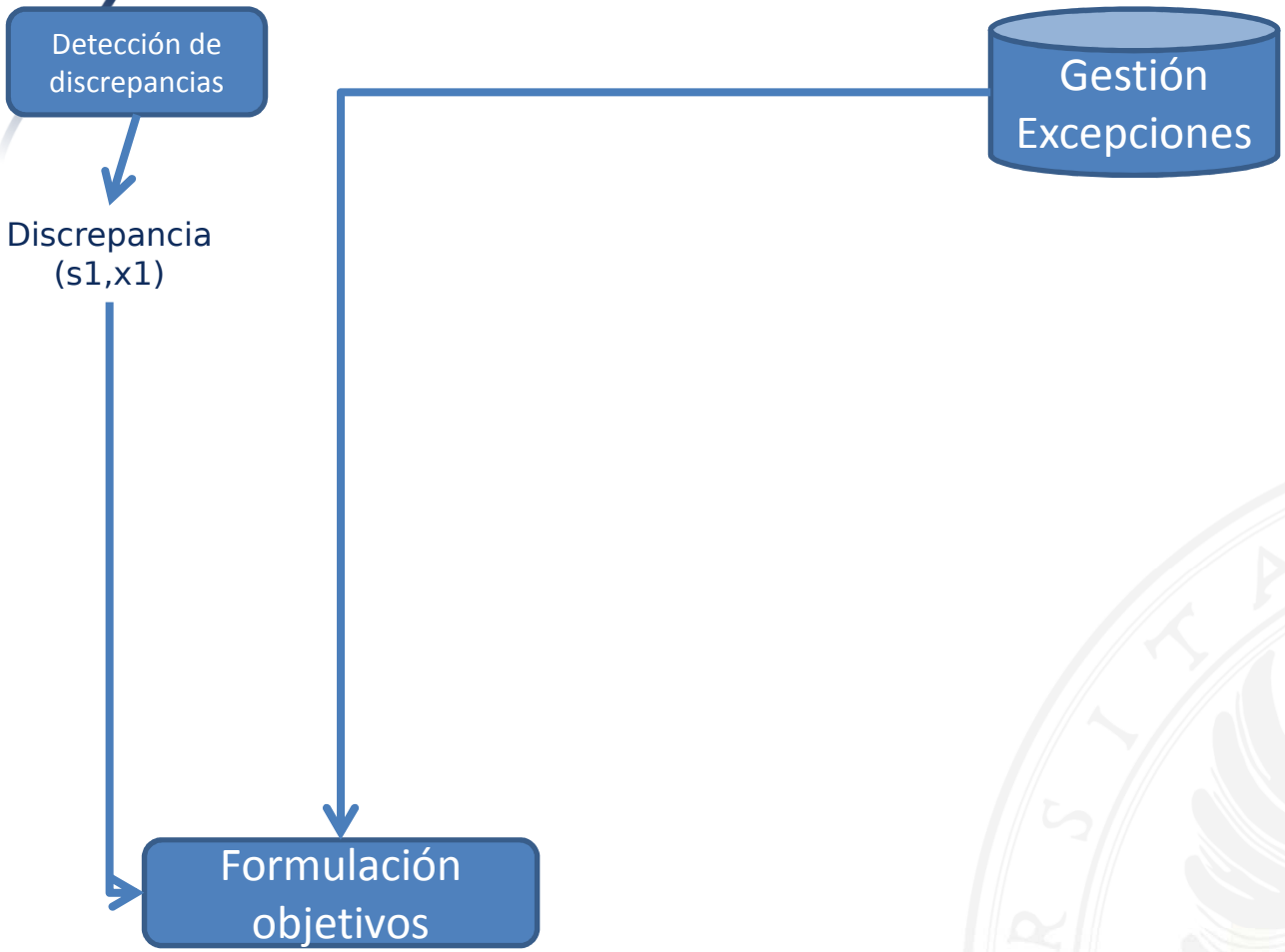
  - No supervisado: aprendizaje activo mediante aprendizaje por refuerzo.

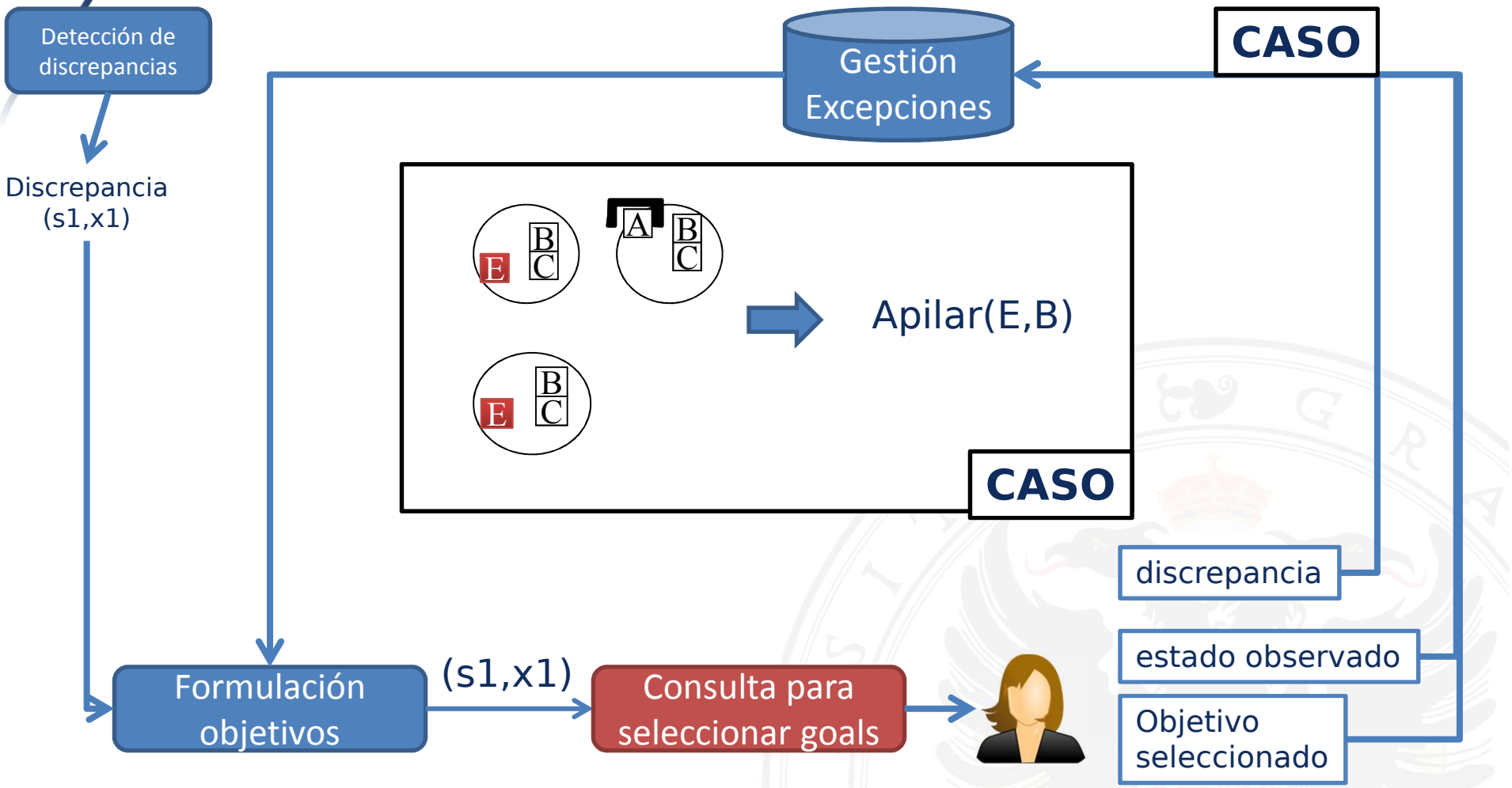
[1]
U. Jaidee, H. Muñoz-Avila, y D. W. Aha, «Integrated learning for goal-driven autonomy», en *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, 2011, pp. 2450–2455.

[1]
U. Jaidee, H. Muñoz-Avila, y D. W. Aha, «Learning and Reusing Goal-Specific Policies for Goal-Driven Autonomy», en *Case-Based Reasoning Research and Development*, B. D. Agudo y I. Watson, Eds. Springer Berlin Heidelberg, 2012, pp. 182-195.

Aprender a plantearse objetivos por sí mismo

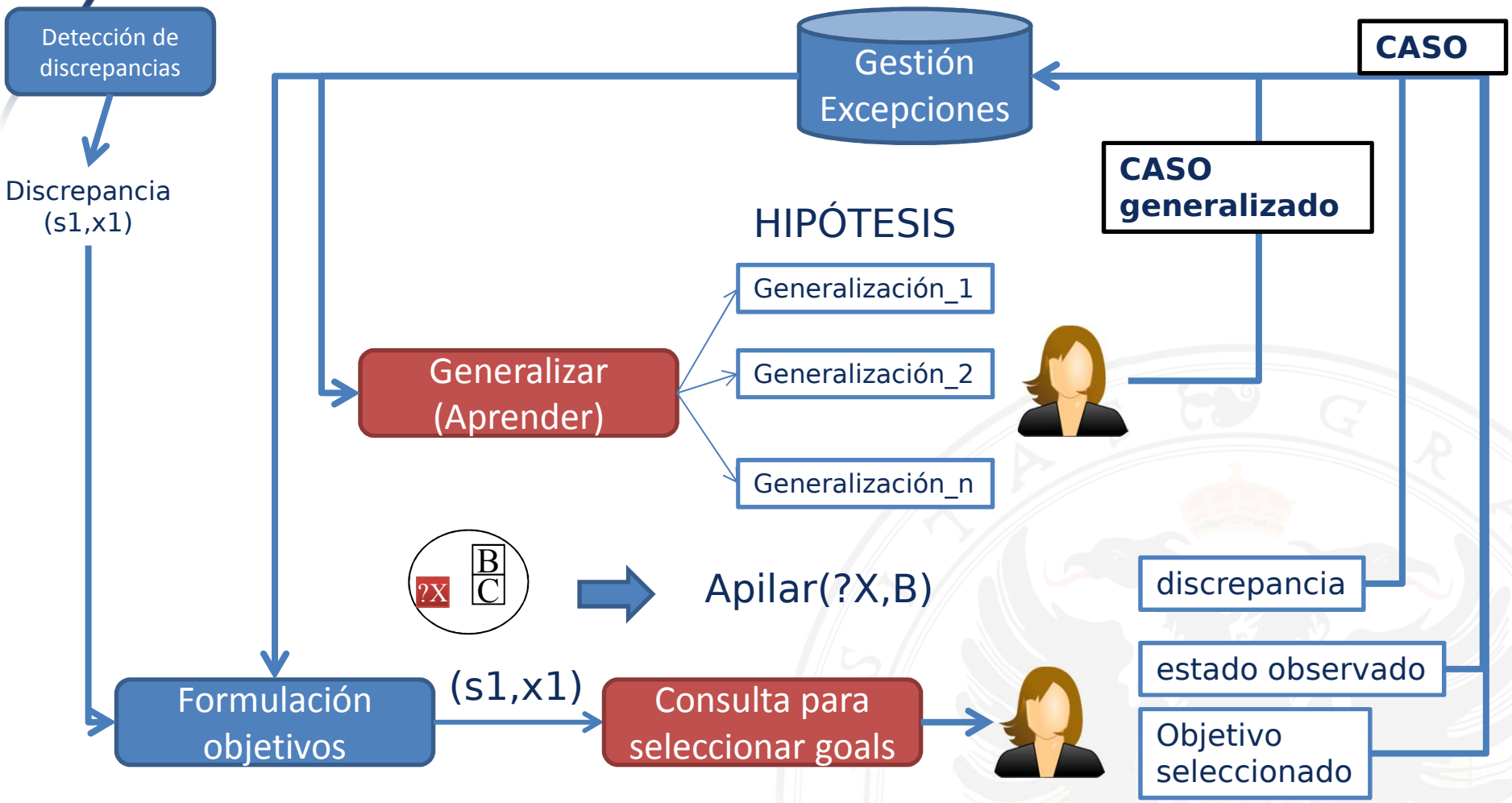
Aprendizaje Supervisado




Detección de discrepancias
Discrepancia (s1,x1)
Gestión Excepciones
CASO
A
B
C
E
Apilar(E,B)
CASO
discrepancia
estado observado
Objetivo seleccionado
Formulación objetivos
(s1,x1)
Consulta para seleccionar goals

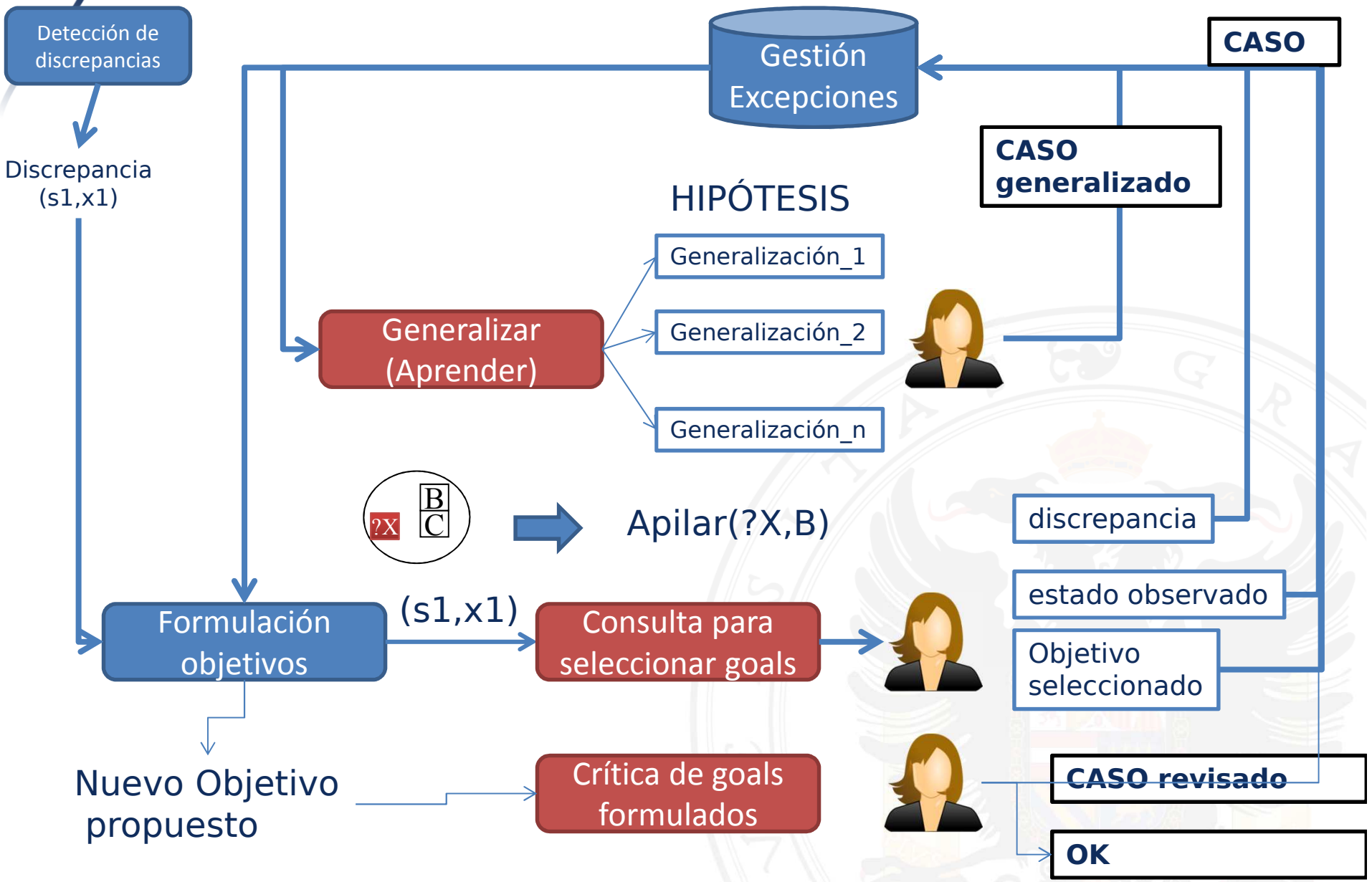
Aprender a plantearse objetivos por sí mismo

Aprendizaje Supervisado



Aprender a plantearse objetivos por sí mismo

Aprendizaje Supervisado



Aprender a plantearse objetivos por sí mismo

No supervisado

- ¿Cómo lo hacemos completamente autónomo, no dependiente del humano?

Aprender a plantearse objetivos por sí mismo

No supervisado

Sustituimos el experto por un proceso de aprendizaje por refuerzo.

La política ahora define la utilidad de **proponer un objetivo** en un estado posible.

Gestión Excepciones = Política Π(g,s)

	S1	…	S	…	S'	…	S_n
g1					Q'1		
…					Q'2		
G			Q				
…					Q'n-1		
g_n					Q'n		

Experto sustitui do por Actualizar política = Aprender

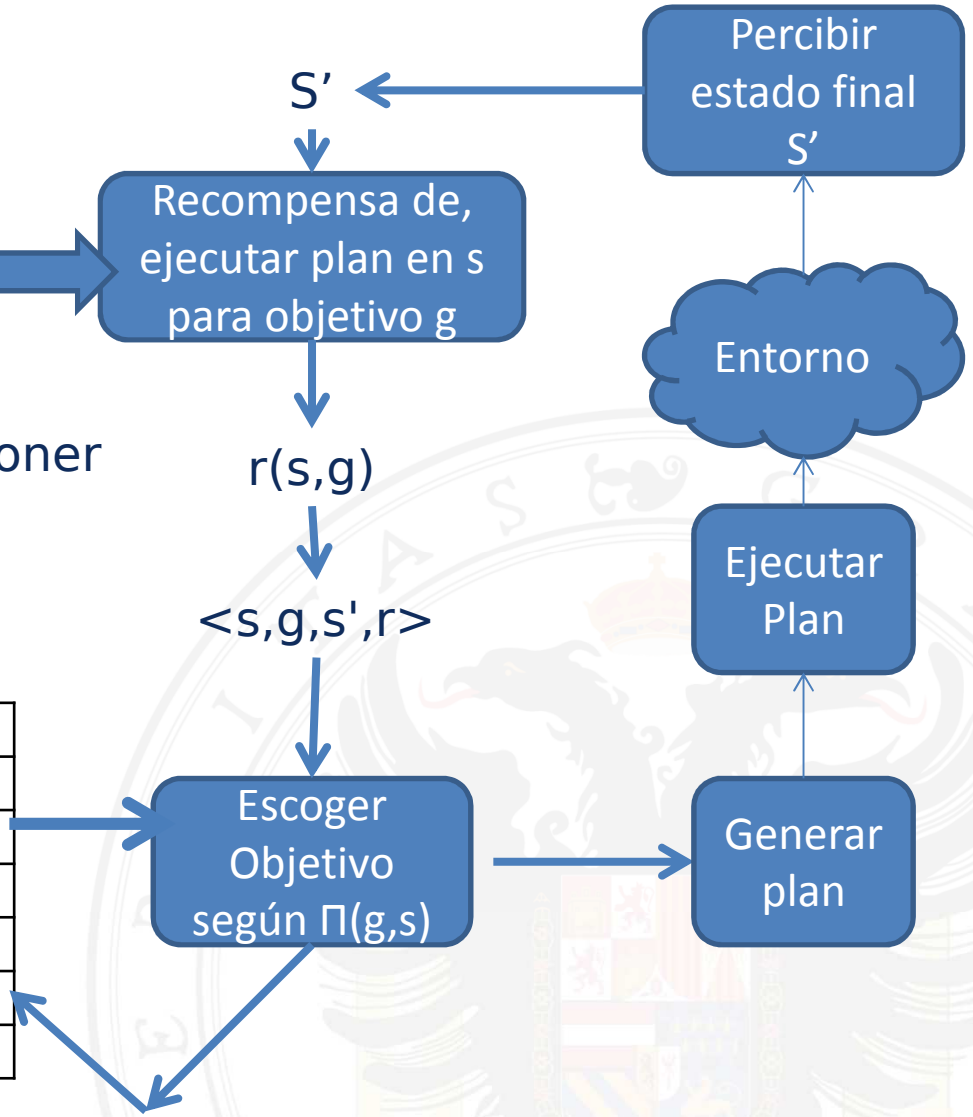
Aprender a plantearse objetivos por sí mismo

Ahora las recompensas son por alcanzar estados a partir de objetivos

La política define la utilidad de proponer un objetivo en un estado posible.

Gestión Excepciones = Política Π(g,s)

	S1	…	S	…	S’	…	S_n
g1					Q’1		
…					Q’2		
G			Q				
…					Q’n-1		
g_n					Q’n		



Experto sustitudo por Actualizar política = Aprender

Aprendiendo a proponerse objetivos, el sistema incrementa su autonomía, pero hasta ahora consideramos un dominio de planificación preexistente. ¿Y si no tenemos dominio de planificación y queremos obtenerlo a partir de trazas de planes, como en Process Mining clásico?

Process Mining vs Learning in Planning

	Process Mining	Aprendizaje planificación
Representación modelo	• Procesos rutinarios, estáticos. • Un modelo de proceso clásico (Red de Petri, Diagrama BPMN,...)	• Procesos dinámicos y orientados a objetivos. • Dominio de planificación: basado en acciones o jerárquico
Ejemplos de entrada	• Logs de eventos (conjuntos de secuencias de actividades	• Log de planes • Por cada traza: estado inicial, estado final • Jerárquicos: tareas indicando a qué objetivo están destinadas.
Algoritmo de aprendizaje	• Salida: un modelo de proceso con relaciones de orden/causales entre actividades y contemplando patrones de proceso comunes	• Basado en acciones: precondiciones y efectos de acciones • Jerárquico: jerarquía de tareas y métodos, y precondiciones de los métodos
Explotación del conocimiento aprendido	•Despliegue de procesos rutinarios que se ajusten mejor a los procedimientos de una organización (normativa, estándars, ….)	•Agentes deliberativos autónomos (videojuegos, robots) •Soporte a decisiones y trabajo del conocimiento.

- Dos ejemplos de sistemas de aprendizaje en planificación:
  - ARMS: aprendizaje de dominios basados en acciones

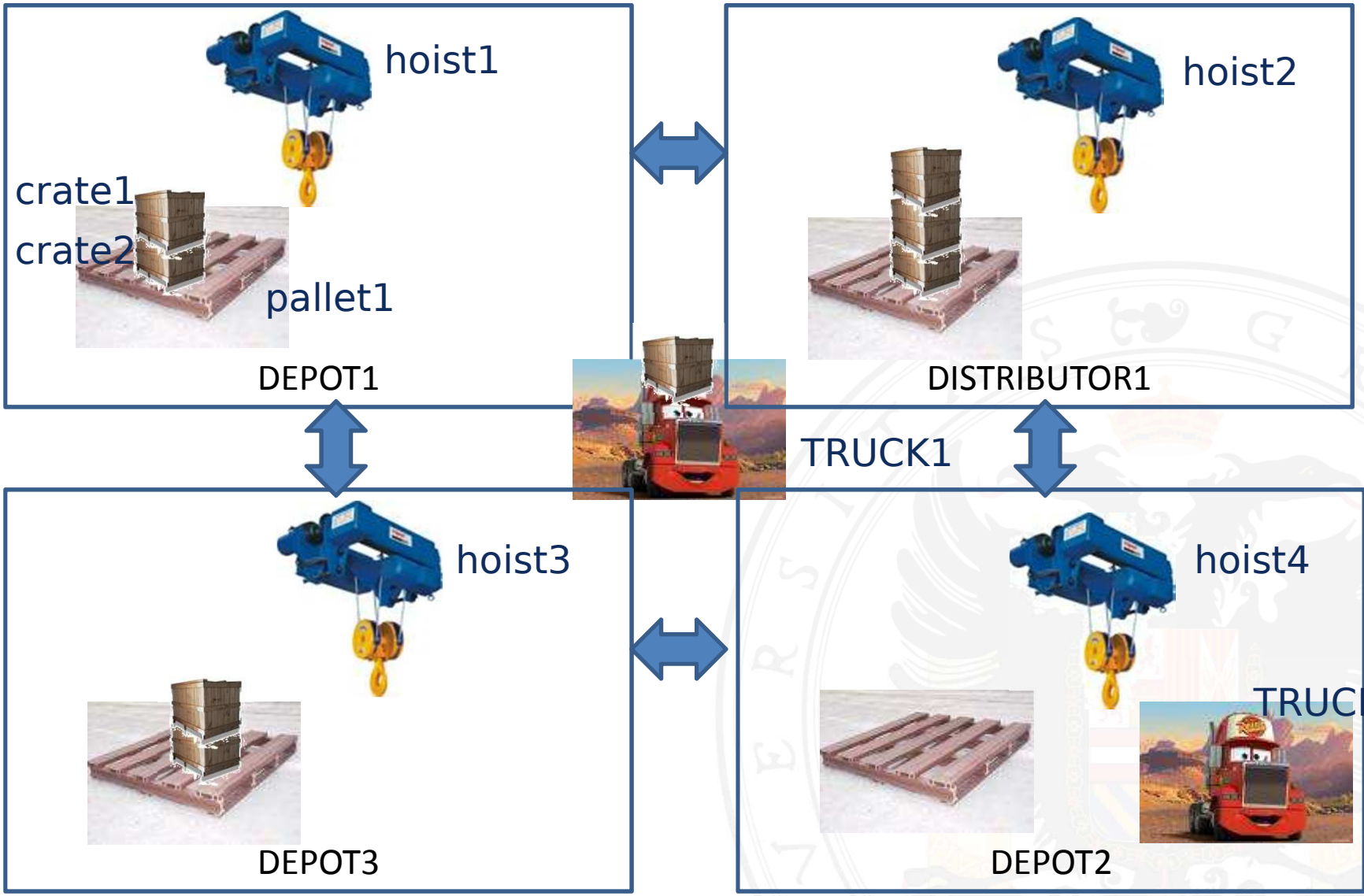
[1]
Q. Yang, K. Wu, y Y. Jiang, «Learning action models from plan examples using weighted MAX-SAT», *Artificial Intelligence*, vol. 171, n.º 2, pp. 107–143, 2007.

[1]
K. Wu, Q. Yang, y Y. Jiang, «Arms: an automatic knowledge engineering tool for learning action models for ai planning», *The Knowledge Engineering Review*, vol. 22, n.º 02, pp. 135–152, 2007.

  - HTN-maker: aprendizaje de dominios jerárquicos.

[1]
C. Hogg, H. Muñoz-Avila, y U. Kuter, «Learning Hierarchical Task Models from Input Traces», *Computational Intelligence*, 2014.

- Depots domain



Aprender modelos de acciones clásicos

Planteamiento

- Objetos
  - Cajas y pallets, Almacenes y Distribuidores, Grúas y camiones
- Aspectos relevantes
  - Pilas de cajas en pallets, distribuidas por Almacenes y Distribuidores
  - Todos las ubicaciones están conectadas
    - Camiones ir directamente a cualquier sitio
    - Camiones pueden cargarse ilimitadamente de cajas
  - Grúas
    - Cargan y Descargan cajas de camiones
    - Cogen y Dejan (Apilan) cajas (sobre cajas o sobre pallets)
  - Pallets son como “mesas” del mundo de bloques
  - Camiones también como “mesas” que cambian de sitio

- **Conducir (Drive)** un camión de una ubicación origen a otra destino

```
(:action Drive
:parameters (?x - truck ?y - place ?z - place)
:precondition (and (at ?x ?y))
:effect (and (not (at ?x ?y)) (at ?x ?z)))
```

- **Coger (Lift)** una *caja que puede estar en un pallet o apilada, con una grua, estando todos en la misma ubicación*

```
(:action Lift
:parameters (?x - hoist ?y - crate ?z - surface ?p -
    place)
:precondition (and (at ?x ?p) (available ?x) (at ?y
    ?p)
                   (on ?y ?z) (clear ?y))
:effect (and (not (at ?y ?p)) (lifting ?x ?y) (not
    (clear ?y))
          (not (available ?x))(clear ?z) (not (on ?y
    ?z))))
```

- .....

A partir de trazas de ejemplos de resolución de problemas en este dominio, ¿cómo aprender las precondiciones y efectos de las acciones?.

- **Inputs: tipos de objetos del dominio, relaciones y cabeceras de acciones**

Table 1
Input domain description for Depot planning domain

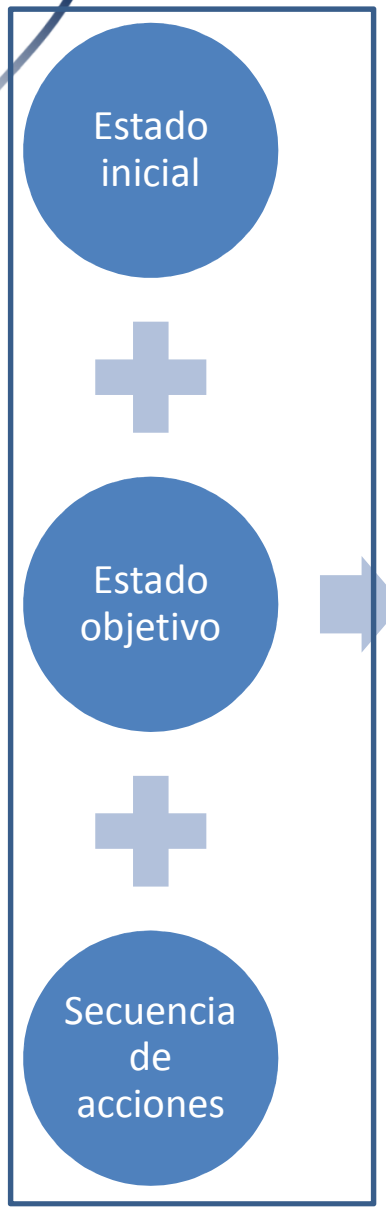
Domain	Depot
types	place locatable - object depot distributor - place truck hoist surface - locatable pallet crate - surface
relations	(at ?x:locatable ?y:place) (on ?x:crate ?y:surface) (in ?x:crate ?y:truck) (lifting ?x:hoist ?y:crate) (available ?x:hoist) (clear ?x:surface)
actions	drive(?x:truck ?y:place ?z:place) lift(?x:hoist ?y:crate ?z:surface ?p:place) drop(?x:hoist ?y:crate ?z:surface ?p:place) load(?x:hoist ?y:crate ?z:truck ?p:place) unload(?x:hoist ?y:crate ?z:truck ?p:place)

- **Inputs: trazas de planes, con estado inicial y objetivo, algún estado intermedio. Cada acción del plan tiene parámetros instanciados**

Table 2
Three plan traces as part of the training examples

	Plan1	Plan2	Plan3
Initial	I_1	I_2	I_3
Step1	lift(h1 c0 p1 ds0), drive(t0 dp0 ds0)	lift(h1 c1 c0 ds0)	lift(h2 c1 c0 ds0)
State		(lifting h1 c1)	
Step2	load(h1 c0 t0 ds0)	load(h1 c1 t0 ds0)	load(h2 c1 t1 ds0)
Step3	drive(t0 ds0 dp0)	lift(h1 c0 p1 ds0)	lift(h2 c0 p2 ds0), drive(t1 ds0 dp1)
State	(available h1)		
Step4	unload(h0 c0 t0 dp0)	load(h1 c0 t0 ds0)	unload(h1 c1 t1 dp1), load(h2 c0 t0 ds0)
State	(lifting h0 c0)		
Step5	drop (h0 c0 p0 dp0)	drive(t0 ds0 dp0)	drop(h1 c1 p1 dp1), drive(t0 ds0 dp0)
Step6		unload(h0 c1 t0 dp0)	unload(h0 c0 t0 dp0)
Step7		drop(h0 c1 p0 dp0)	drop(h0 c0 p0 dp0)
Step8		unload(h0 c0 t0 dp0)	
Step9		drop(h0 c0 c1 dp0)	
Goal	(on c0 p0)	(on c1 p0) (on c0 c1)	(on c0 p0) (on c1 p1)

I_1: (at p0 dp0), (clear p0), (available h0), (at h0 dp0), (at t0 dp0), (at p1 ds0), (clear c0), (on c0 p1), (available h1), (at h1 ds0).
I_2: (at p0 dp0), (clear p0), (available h0), (at h0 dp0), (at t0 ds0), (at p1 ds0), (clear c1), (on c1 c0), (on c0 p1), (available h1), (at h1 ds0).
I_3: (at p0 dp0), (clear p0), (available h0), (at h0 dp0), (at p1 dp1), (clear p1), (available h1), (at h1 dp1), (at p2 ds0), (clear c1), (on c1 c0), (on c0 p2), (available h2), (at h2 ds0), (at t0 ds0), (at t1 ds0).



Log de planes

ARMS: a partir de logs de planes aprende dominios de planificación basados en acciones.

Un agente deliberativo autónomo puede aprender a desarrollar su comportamiento mediante ejemplos de planes de actuación.

… +
Relaciones + Tipos de objetos + Cabeceras de acciones

1. Obtener el grafo causal
    1. Parecido al visto en el Tema 1, pero con distintas relaciones y medidas de causalidad.
    2. **Un par de acciones está conectado causalmente si algunos de sus tipos de parámetros coinciden.**
        1. Observar que dos acciones pueden aparecer ordenadas en el plan, pero no tienen por qué estar conectadas.
    3. Cada par conectado (a_i, a_j) del grafo causal se etiqueta con
        1. Soporte(a_i, a_j)=Número de veces que aparece el par
        2. Tasa Soporte(a_i,a_j)= Soporte(a_i,a_j)/Número total de pares de acciones
    4. Solo considerar las relaciones causales que cumplan
        1. Tasa_Soporte > Umbral

2. Idea principal
    1. Si un par (a1,a2) aparece en el grafo causal es porque tiene que existir un predicado “conector” para a1 -> **p** -> a2
    2. Además, **p** tiene que estar en la lista de adición de a1 y en las precondiciones de a2.
    3. Los predicados así obtenidos y asignados tienen que respetar las reglas de escritura de dominios basados en operadores y tienen que garantizar que los planes son correctos.

1. ¿Cómo determinar ese predicado conector?
    1. Mediante búsqueda (prueba y error) entre los predicados dados como entrada.
        1. Hay maneras de limitar el número de candidatos, dependiendo de las acciones.
    2. Problema de optimización de restricciones booleanas. MAX-SAT
2. Restricciones
    1. Siguiente transparencia (solo para hacerse una idea)
3. La solución al problema de satisfacción de restricciones es una asignación de predicados (relaciones) a listas de precondiciones y efectos de las acciones.

1.

(1) (Constraint A.1) The intersection of the precondition and add lists of all actions must be empty.

$$pre_i \cap add_i = \phi.$$

(2) (Constraint A.2) In addition, if an action's delete list includes a relation, this relation is in the action's precondition list. Thus, for every action, we require that the delete list is a subset of the precondition list.

$$del_i \subseteq pre_i.$$

- (Constraint I.1) The relation p must be generated by an action $a_{i_k} (0 \leqslant i_k \leqslant n)$, that is, p is selected to be in the add-list of a_{i_k}. $p \in (add_{i_1} \cup add_{i_2} \cup \cdots \cup add_{i_k})$, where $\cup$ means logical "or".
- (Constraint I.2) The last action a_{ik} must not delete the relation p; that is, p must not be selected to be in the delete list of a_{ik}: $p \notin del_{ik}$.

- (Constraint I.3) We define the weight value of a relation-action pair (p, a) as the occurrence probability of this pair in all plan examples. If the probability of a relation-action pair is higher than the probability threshold θ, then we set a corresponding relation constraint $p \in \mathrm{PRECOND}_a$, which receives a weight value equal to its prior probability.

- (Constraint P.1) Every precondition p of every action b must be in the add list of a preceding action a and is not deleted by any actions between a and b.
- (Constraint P.2) In addition, at least one relation r in the add list of an action must be useful in achieving a precondition of a later action. That is, for every action a, an add list relation r must be in the precondition of a later action b, and there is no other action between a and b that either adds or deletes r.

- (Plan constraint P.3) One of the relevant relations p must be chosen to be in the preconditions of both a_i and a_j, but not in the delete list of a_i,

$$\exists p \ (p \in (pre_i \cap pre_j) \wedge p \notin (del_i))$$

- (Constraint P.4) The first action a_i adds a relevant relation that is in the precondition list of the second action a_j in the pair,

$$\exists p \ (p \in (add_i \cap pre_j))$$

- (Constraint P.5) A relevant relation p that is deleted by the first action a_i is added by a_j. The second clause is designed for the event when an action re-establishes a fact that is deleted by a previous action.

$$\exists p \ (p \in (del_i \cap add_j))$$

- Aprender dominios jerárquicos

```
(:task Tstack
 :parameters (?x ?y - block)
 (:method clear
  :precondition (clear ?y)
  :tasks ((Tpickup ?x)(stack ?x ?y)))
 (:method not_clear
  :precondition(on ?z ?y)
  :tasks ((Tu
           (T

(:task Tunsta
 :parameters (?x ?y - block)
 (:method clear
  :precondition (clear ?x)
  :tasks (unstack ?x ?y))
 (:method not_clear
  :precondition(on ?z ?x)
  :tasks ((Tunstack ?z ?x)(Tputdown ?z)(unstack ?x
?y))))
```

```
(:task Tpickup
 :parameters (?x - block)
 (:method clear
  :precondition (and (clear ?x) (ontable ?x))
  :tasks (pickup ?x))
 (:method clear2
  :precondition (and (clear ?x)(on ?x ?z))
                                          pickup ?x))))

(:task Tputdown
 :parameters (?x - block)
 (:method clear
  :precondition (holding ?x)
  :tasks (putdown ?x))
 (:method ocupado
  :precondition ()
  :tasks ((Tpickup ?x)(putdown ?x))))
```

¿Cómo aprender esto?

(:**action** pickup
:**parameters** (?x - block)
:**precondition** (and (ontable ?x)(clear ?x)(handempty))
:**effect** (and (not (ontable ?x)) (not (clear ?x))(not (handempty)) (holding ?x)))

(:**action** putdown
:**parameters** (?x - block)
:**precondition** (holding ?x)
:**effect** (and (ontable ?x) (clear ?x) (handempty) (not (holding ?x))))

(:**action** stack
:**parameters** (?x ?y - block)
:**precondition** (and (holding ?x)(clear ?y))
:**effect** (and (not (holding ?x)) (not (clear ?y)) (clear ?x)(on ?x ?y) (handempty)))

(:**action** unstack
:**parameters** (?x ?y - block)
:**precondition** (and (handempty) (clear ?x)(on ?x ?y))
:**effect** (and (holding ?x)(clear ?y)(not (clear ?x)) (not(on ?x ?y)) (not (handempty)

¿Conocido esto?

Aprender modelos jerárquicos

Entradas y salida

Dominio de acciones

Conjunto de trazas de planes con todos los estados intermedios.

Tareas anotadas con pre y post-condiciones

Conjunto de métodos preexistente (para que sea incremental)

Dominio de acciones

Conjunto de trazas de planes con todos los estados intermedios.

Tareas anotadas con pre y post-condiciones

Conjunto de métodos preexistente (para que sea incremental)

```
(:action pickup
 :parameters (?x - block)
 :precondition (and (ontable ?x)(clear
?x)(handempty))
 :effect (and (not (ontable ?x)) (not (clear
?x))(not (handempty)) (holding ?x)))

(:action putdown
 :parameters (?x - block)
 :precondition (holding ?x)
 :effect (and (ontable ?x) (clear ?x)
(handempty) (not (holding ?x))))
```

```
(:action stack
 :parameters (?x ?y - block)
 :precondition (and (holding ?x)(clear ?y))
 :effect (and (not (holding ?x)) (not (clear
?y)) (clear ?x)(on ?x ?y) (handempty)))

(:action unstack
 :parameters (?x ?y - block)
 :precondition (and (handempty) (clear ?x)(on
?x ?y))
 :effect  (and (holding ?x)(clear ?y)(not
(clear ?x)) (not(on ?x ?y)) (not (handempty)
```

Dominio de acciones

Conjunto de trazas de planes con todos los estados intermedios.

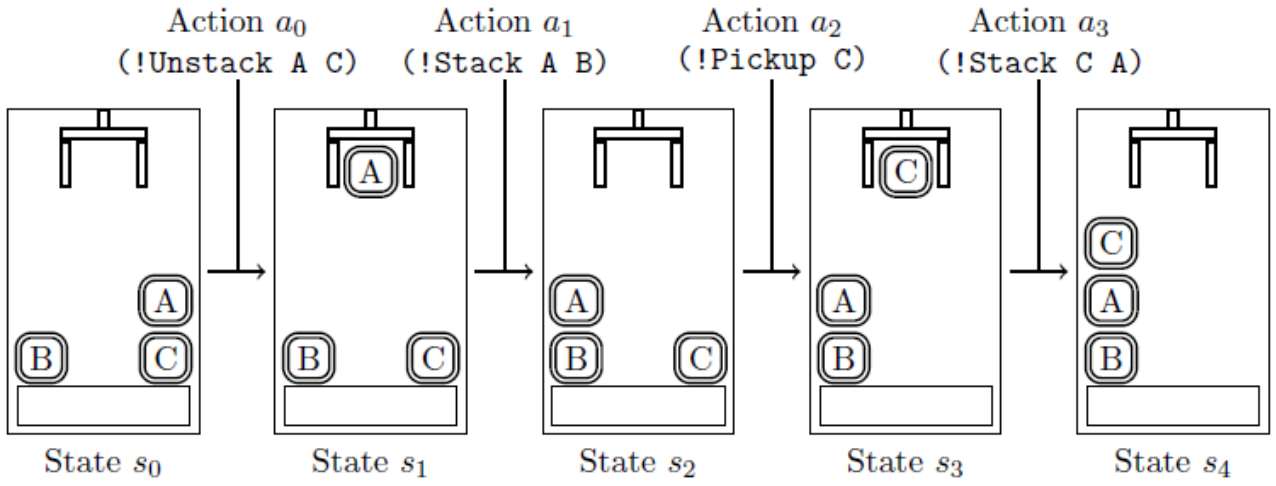


Figure 6: An example plan in the BLOCKS-WORLD domain.

```
( :task Make-2Pile
  :parameters ( ?a ?b )
  :precondition
  ( and )
  :postcondition
  ( and (on-table ?b) (on ?a ?b) (clear ?a) ) )
```

Figure 5: An example annotated task in the BLOCKS-WORLD domain.

Tareas anotadas con pre y post-condiciones

Conjunto de métodos preexistente (para que sea incremental)

Ojo que no son acciones, son tareas de alto nivel para las que se especifican condiciones que deben cumplirse antes de ejecutar y una vez ejecutada la tarea.

En el ejemplo vamos a asumir que como entrada tiene tareas para conseguir pilas de 1, 2 o 3 bloques.

Son necesarias para guiar el proceso de aprendizaje.

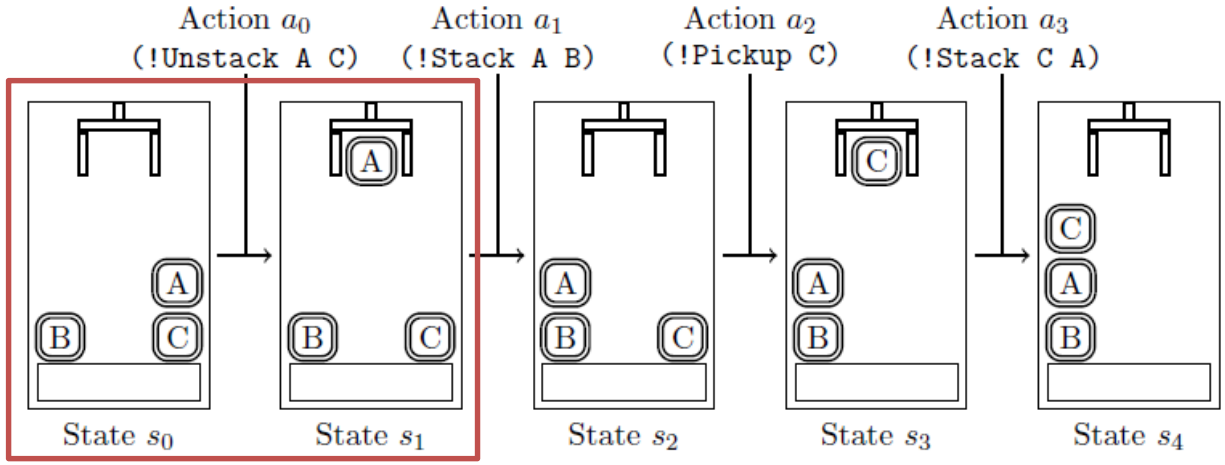
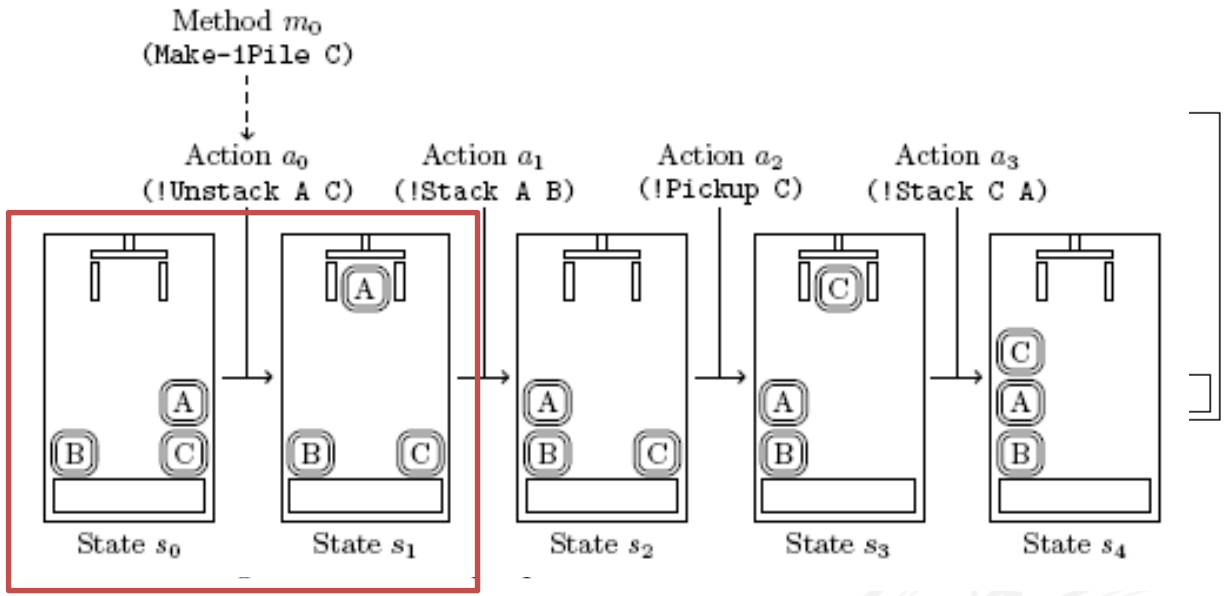
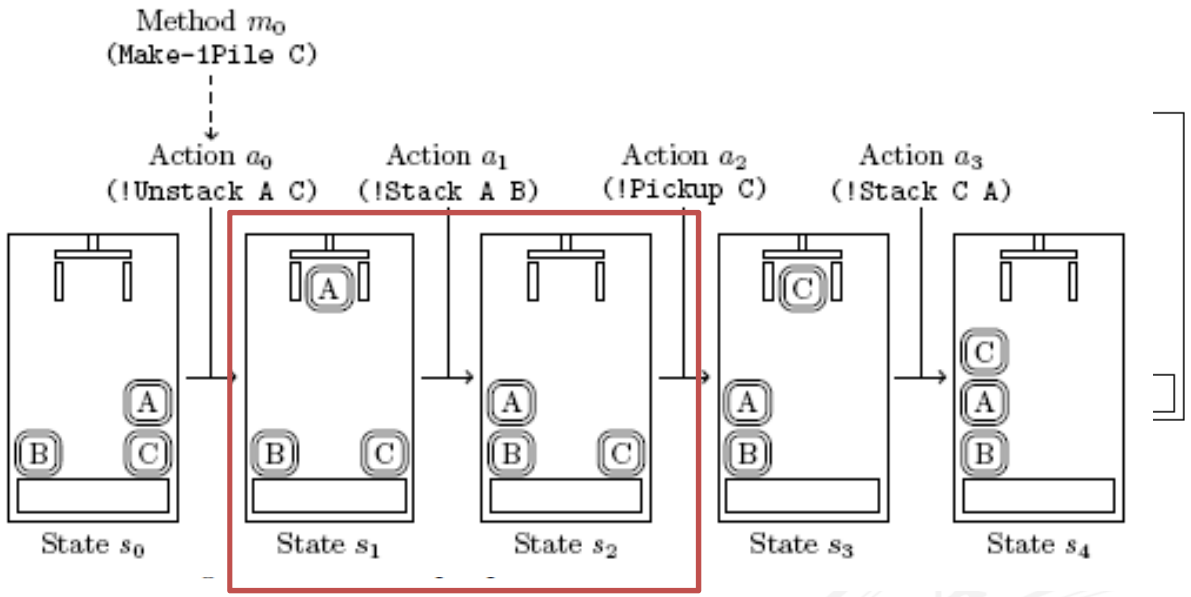


Figure 6: An example plan in the BLOCKS-WORLD domain.

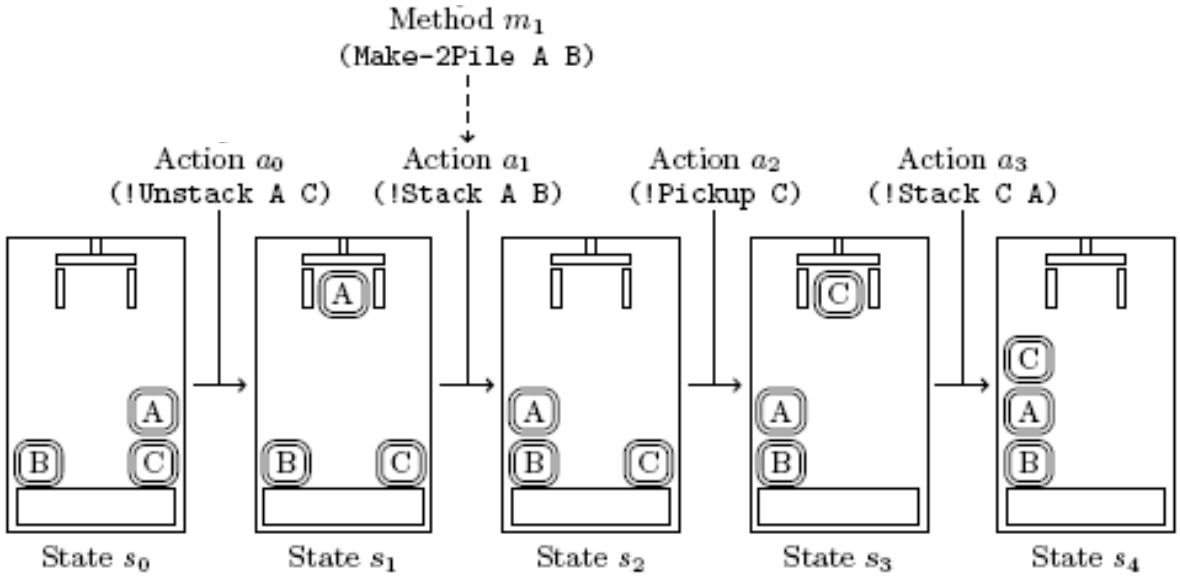
- Considera el subplan conteniendo sólo <a_0>.
- ¿Hay alguna tarea anotada cuyas precondiciones son satisfechas en s_0 y cuyas postcondiciones son satisfechas en s_1 pero no en s_0?
- Sí la hay.



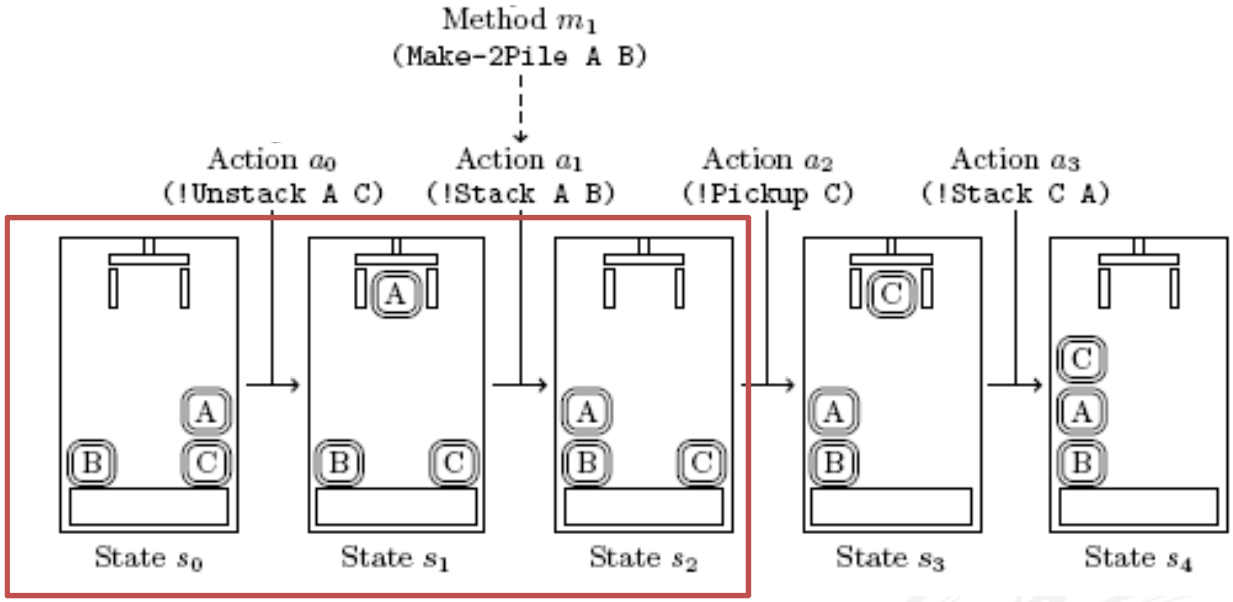
- Considera el subplan conteniendo sólo <a_0>.
- ¿Hay alguna tarea anotada cuyas precondiciones son satisfechas en s_0 y cuyas postcondiciones son satisfechas en s_1 pero no en s_0?
- Sí la hay (asumir que se ha dado como entrada).
- HTN-maker aprende el método m_0 para explicar cómo se puede hacer una pila de tamaño 1.
  - m_0 tiene una sola tarea (a_0) y como precondiciones las propias de la acción a_0.



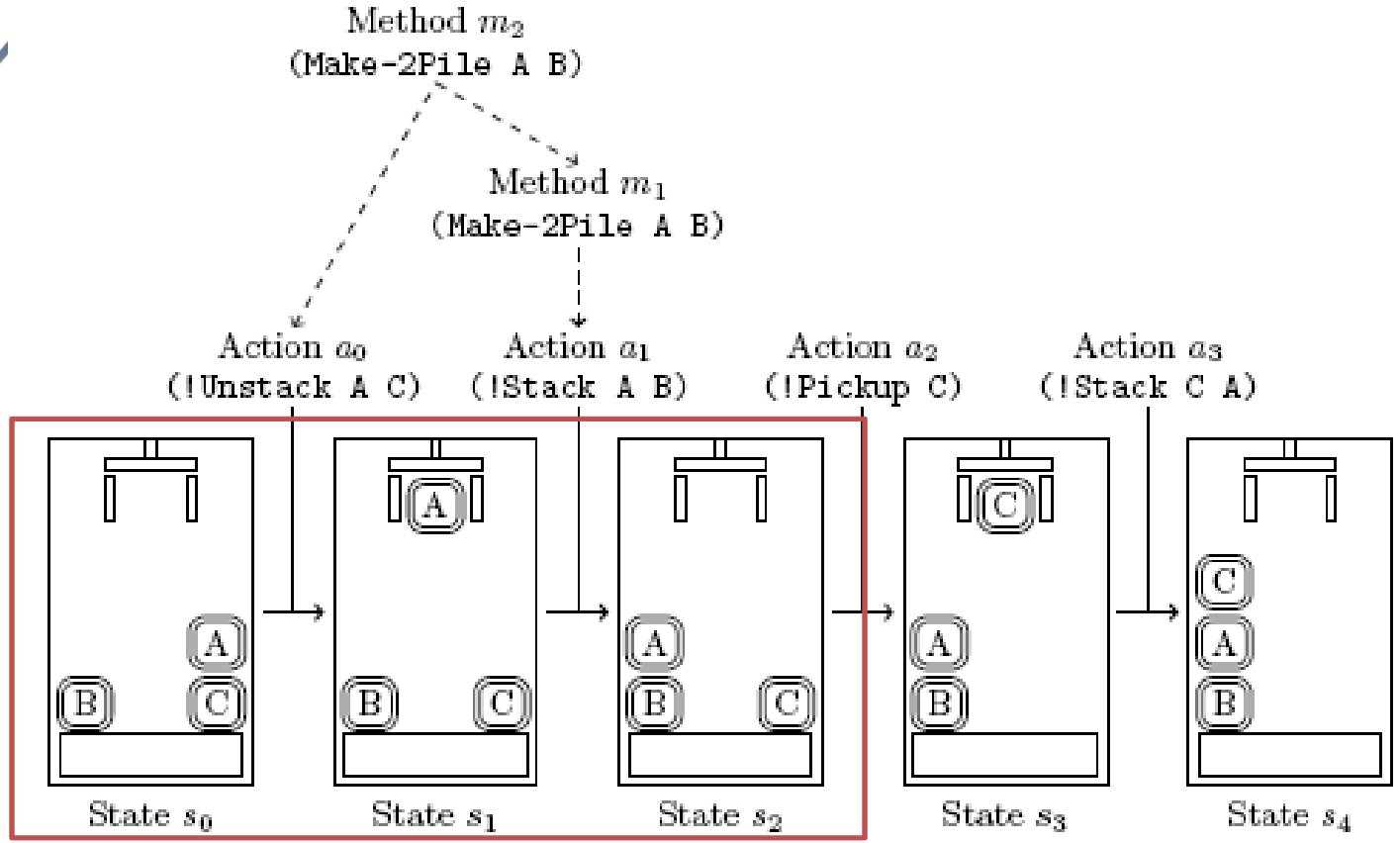
- Considera el subplan conteniendo sólo <a_1>.
- Detecta que s_2 empareja con las postcondiciones de `(make_2Pile A B)`



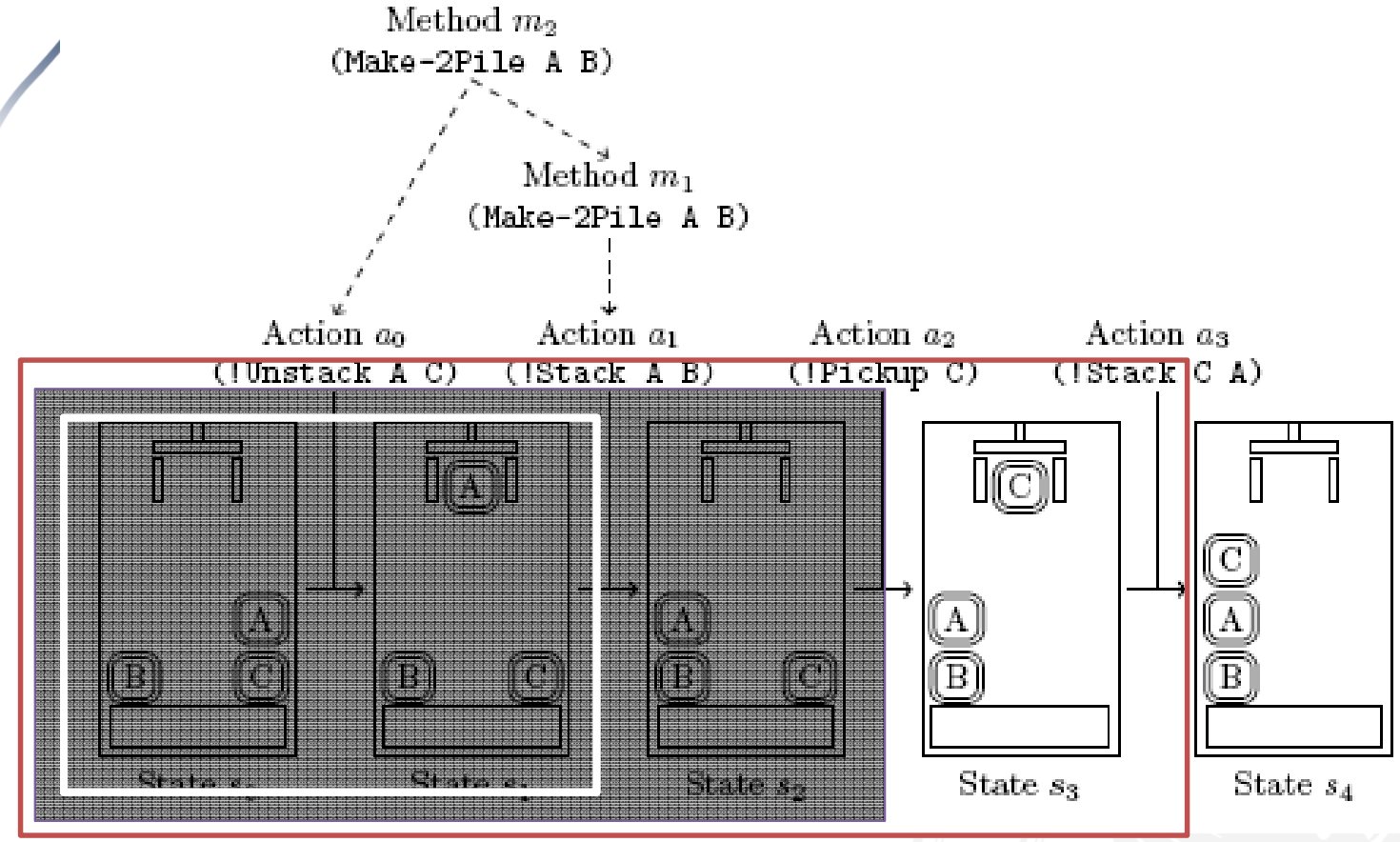
- Considera el subplan conteniendo sólo <a_1>.
- Detecta que s_2 empareja con las postcondiciones de `(make_2Pile A B)`
- Aprende otro nuevo método m_1



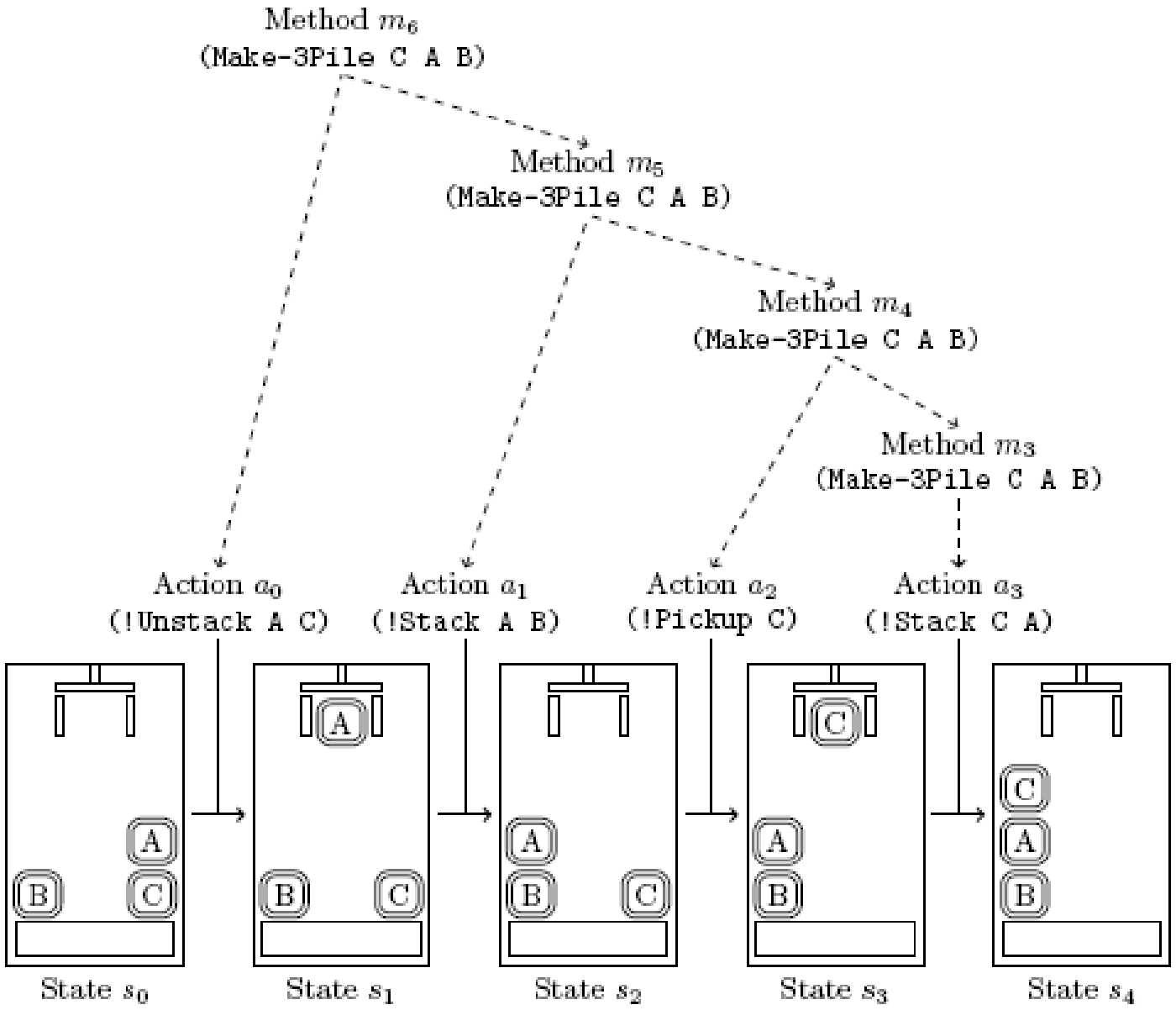
- Considera el subplan conteniendo sólo <a_0, a_1>.
- Detecta que s_2 empareja con las postcondiciones de `(make_2Pile A B)`



- Considera el subplan conteniendo sólo <a_0, a_1>.
- Detecta que s_2 empareja con las postcondiciones de `(make_2Pile A B)`
- Aprende otro nuevo método m_2 y observar que es recursivo.



- Luego considera <a2>, <a1,a2>, <a0,a1,a2>, pero no encuentra ninguna tarea de las anotadas que pueda encajar con esos subplanes.



A3

A2 a3

A1 a2 a3

A0 a1 a2 a3

Finalmente para estos otros subplanes encuentra otro conjunto de métodos en un proceso análogo al anterior.

- HTN maker repite este proceso para cada traza.
- Aplica posteriormente un proceso de generalización para convertir objetos instanciados en variables y reducir repeticiones.

- Obtiene como resultado un dominio de planificación jerárquico.

- Los dominios jerárquicos obtenidos no se parecen a los que crearía un humano, pero permiten establecer estrategias jerárquicas automáticamente para resolver problemas.

Recapitulación

- Tema1: Process Mining
  - Aprender modelos de proceso para trabajo rutinario (estáticos) a partir de logs de eventos
  - Algoritmo-alpha: patrones de proceso básicos
  - Process Mining Heurístico: patrones de proceso más comunes->modelos de proceso más expresivos.
  - Para reducir el esfuerzo en modelado o para mejorar procesos ya existentes,
- Tema2: ACM y Planificación automática
  - ACM requiere modelos de procesos para "knowledge work" (orientados a objetivos) que permitan generar procesos dinámicos, que se adaptan a la situación y que pueden modificarse on-line para adaptarse a nuevas situaciones, en un ciclo cerrado de planificación continua.
  - La planificación automática es una tecnología que cubre estas necesidades, porque permite generar planes para alcanzar un objetivo a partir de una situación dada.
    - Dominio de planificación: modelo de proceso para procesos orientados a objetivos (trabajo del conocimiento)
    - Objetivos: establecen qué tiene que alcanzarse.
    - Estado inicial: situación dada a partir de la que se generan planes (procesos dinámicos) para alcanzar objetivos.
  - Modelos basados en acciones y modelos jerárquicos (HTN)
  - Planificación continua.
- Tema3: Aprendizaje en planificación
  - Igual que en Process Mining clásico, en ACM podemos aprender modelos de procesos orientados a objetivos aplicando técnicas de aprendizaje en planificación
  - Doble vertiente aprendizaje en planificación:
    - Mejorar el comportamiento de agentes autónomos.
    - Mejorar las decisiones de expertos soportadas por herramientas inteligentes.
  - Aprendizaje de:
    - Objetivos: cómo un agente deliberativo puede auto-plantearse objetivos para continuar con su trabajo
    - Dominios de planificación: un agente deliberativo puede aprender a comportarse/mejorar su comportamiento orientado a objetivos a partir de ejemplos de actuación.

- Lecturas recomendadas:
  - ACM:
    - SlideShare:http://social-biz.org/2010/04/20/nature-of-knowledge-work/
    - Blog de Keith Swenson https://social-biz.org/
  - Process Mining: *van der Aalst, Wil M. P. Process Mining. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. http://link.springer.com/10.1007/978-3-642-19345-3.*
    - Introducción y Petri Nets: Cap1, Cap 2.2.1 a 2.2.3.
    - Alpha algorithm y Heuristic Mining (Cap5.1, 5.2 y Cap 6.2)
  - Survey on HTN: *Georgievski, Ilche, y Marco Aiello. «HTN planning: Overview, comparison, and beyond». Artificial Intelligence 222 (mayo de 2015): 124-56. doi:10.1016/j.artint.2015.02.002.*
  - HTN aplicado en toma de decisiones médicas: *Sánchez-Garzón, Inmaculada, Arturo González-Ferrer, y Juan Fernández-Olivares. «A knowledge-based architecture for the management of patient-focused care pathways». Applied intelligence 40, n.º 3 (2014): 497–524.*
  - Learning planning domains
    - Learning action models from plan traces: *Q. Yang, K. Wu, y Y. Jiang, «Learning action models from plan examples using weighted MAX-SAT», Artificial Intelligence, vol. 171, n.º 2, pp. 107–143, 2007.*
    - Learning HTN domains from plan traces *C. Hogg, H. Muñoz-Avila, y U. Kuter, «Learning Hierarchical Task Models from Input Traces», Computational Intelligence, 2014.*