

Fundamentos de la robótica autónoma

Paradigmas en Robótica

**Instituto de Computación
Facultad de Ingeniería
Universidad de la República**

Organización

Paradigmas

Jerarquico

Reactivo

Hibrido

Arquitecturas representativas

Paradigmas

Existen tres paradigmas para organizar la inteligencia en un robot:

- Jerárquico

- Reactivo

- Híbrido Deliberativo/Reactivo

Existen tres funciones primitivas en robótica:

- Sensar (SENSE)

- Planificar (PLAN)

- Actuar (ACT)

Paradigmas

Primitivas robóticas

Primitiva robótica	Entrada	Salida
Sensar (SENSE)	Datos de los sensores	Información sensada
Planificar (PLAN)	Información (sensorial o cognitiva)	Directivas
Actuar (ACT)	Información sensada o directivas	Comandos a los actuadores

Paradigmas

Los paradigmas se describen de dos maneras:

- Por la relación entre las primitivas SENSAR, PLANIFICAR y ACTUAR.
- Por la manera en que los datos sensoriales son procesados y distribuidos en el sistema.

Arquitecturas

Las arquitecturas proporcionan la manera general de organizar un sistema de control.

Describe un conjunto de componentes y la forma en que interactúan.

Las características más importantes a tener en cuenta al momento de evaluar una arquitectura son:

- Modularidad

- Lugar de aplicación

- Portabilidad

- Robustez

Paradigma Jerárquico

Organización

Introducción al paradigma.

Strip.

Arquitecturas representativas:

NHC

RCS

Consideraciones de programación.

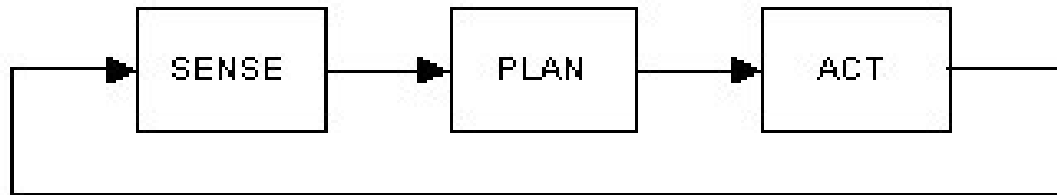
Introducción

Jerárquico (1967-1990)

Esta basado en una visión introspectiva de cómo las personas piensan.

Secuencial y ordenado.

En cada paso se planifica que hacer.



Se arma un modelo global del mundo el cual es utilizado para planificar las acciones.

Modelo del mundo

Todas las observaciones se juntan para formar una estructura de datos global accedida por el planificador, esta estructura es denominada modelo del mundo.

El modelo del mundo típicamente contiene:

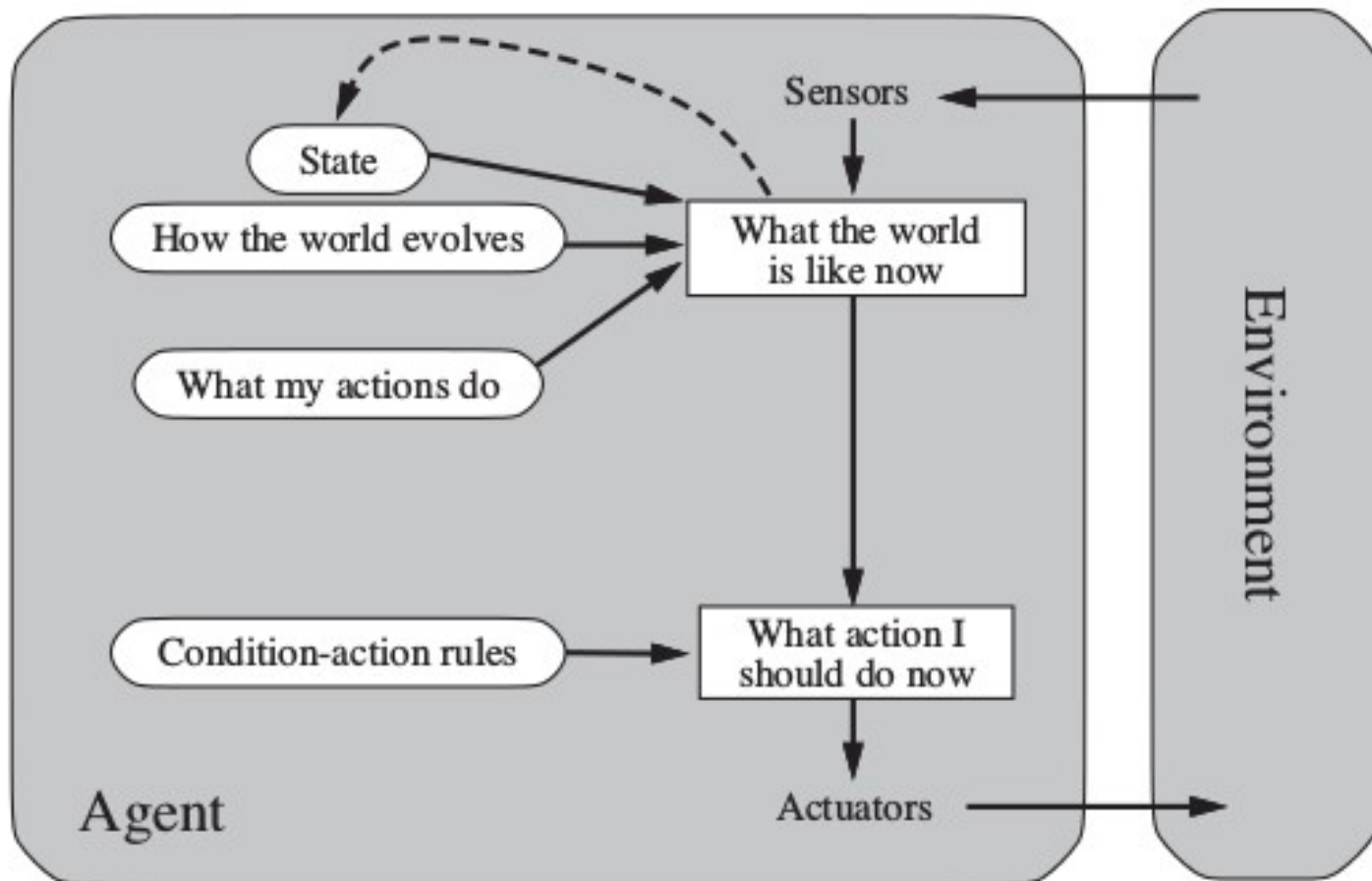
- Una representación a priori del entorno en el cual opera el robot.

- Información sensorial.

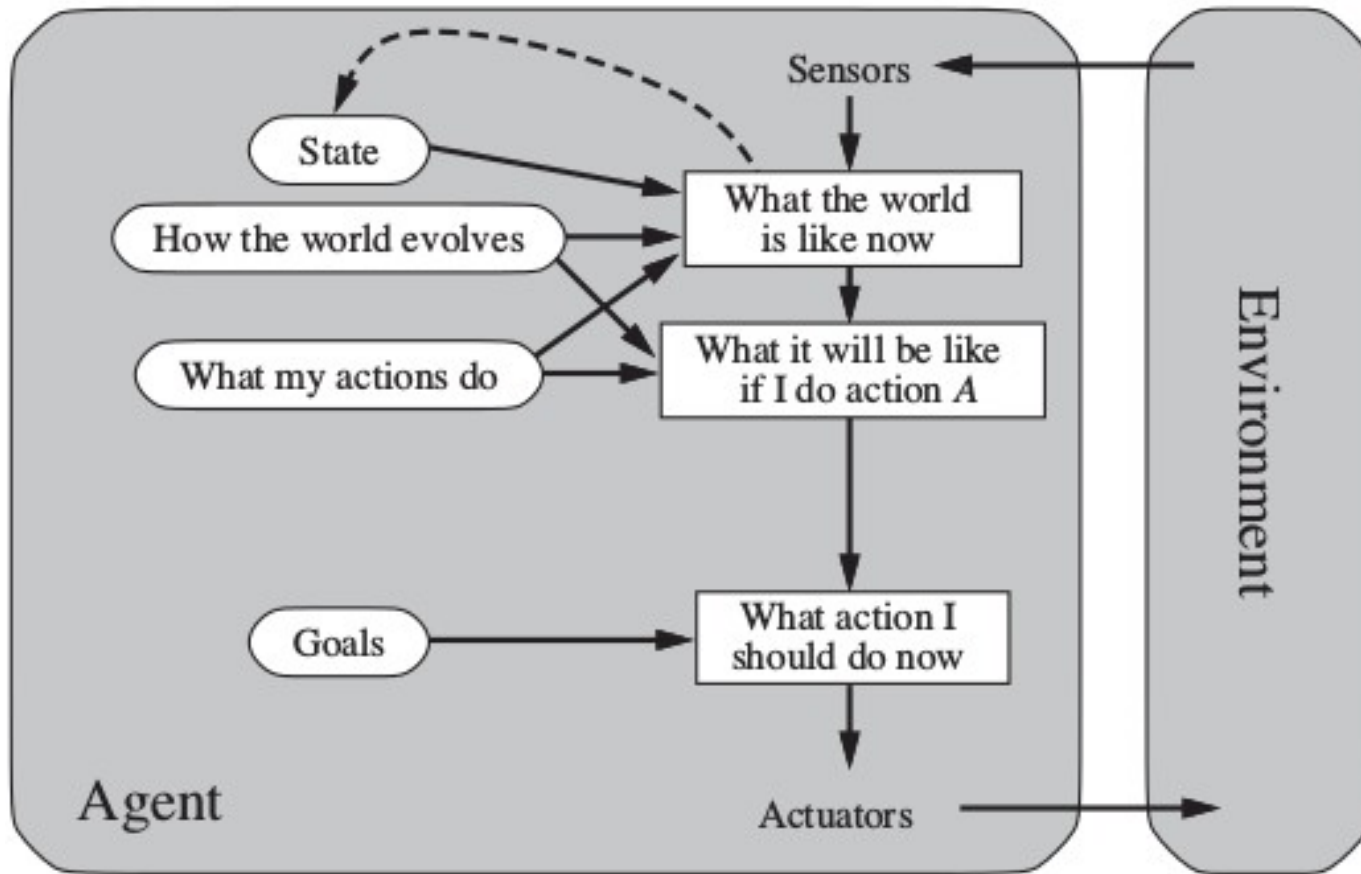
- Conocimiento necesario para realizar la tarea.

Todo lo relevante para el problema es representado por axiomas.


Agente reflejo basado en modelo

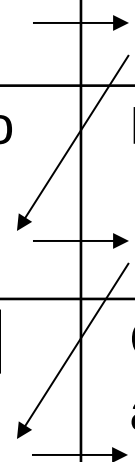


Agente basado en objetivos y modelo



Introducción

Primitiva robótica	Entrada	Salida
Sensar (SENSE)	Datos de los sensores	Información sensada
Planificar (PLAN)	Información (sensorial o cognitiva)	Directivas
Actuar (ACT)	 directivas	Comandos a los actuadores



Strip

Ejemplo
Algoritmo
Definiciones

Shakey

Primer robot con IA.

Desarrollado por el Stanford Research Institute (SRI) para DARPA 1967-9.

Usa STRIP para determinar que acción tomar.

Costo u\$s 100.000.



Strip

El robot Shakey necesitaba un algoritmo de propósito general para planificar de forma de alcanzar su objetivo.

El método utilizado es una versión simplificada del General Problem Solver (GPS), denominado Strip.

Strip utiliza una técnica denominada means-ends analysis.

Strip

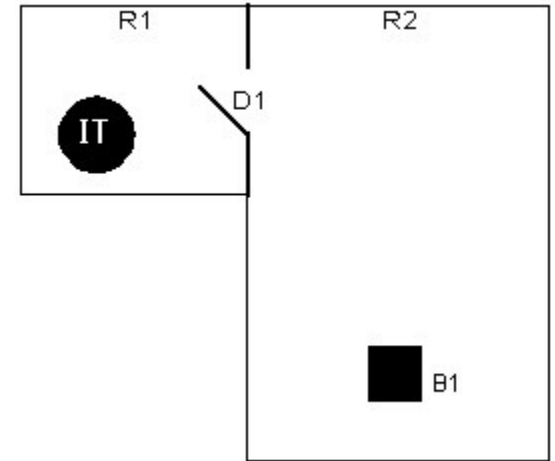
El conocimiento del robot puede estar representado por los siguientes predicados:

$INROOM(x,r)$, donde x es un objeto de tipo `movable_object` y r es de tipo `room`.

$NEXTTO(x,t)$, donde x es un objeto de tipo `movable_object` y t es un objeto de tipo `movable_object` o `door`.

$STATUS(d,s)$, donde d es de tipo `door` y s es un enumerado de tipo: `OPEN` o `CLOSED`.

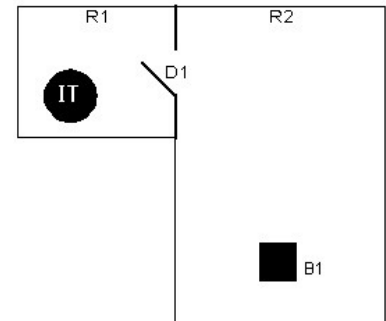
$CONNECTS(d,r_x,r_y)$, donde d es de tipo `door` y r_x,r_y son de tipo `room`.



Strip

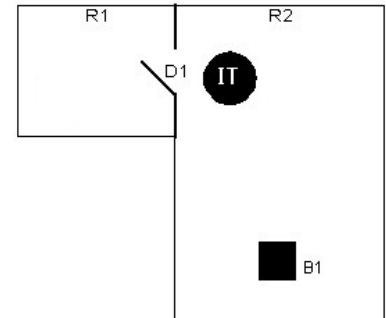
El modelo del mundo para el estado inicial es:

INROOM(IT,R1)
INROOM(B1,R2)
CONNECTS(D1,R1,R2)
CONNECTS(D1,R2,R1)
STATUS(D1,OPEN)



El modelo del mundo para el estado final es:

INROOM(IT,R2)
INROOM(B1,R2)
CONNECTS(D1,R1,R2)
CONNECTS(D1,R2,R1)
STATUS(D1,OPEN)



Strip

Tabla parcial de diferencias

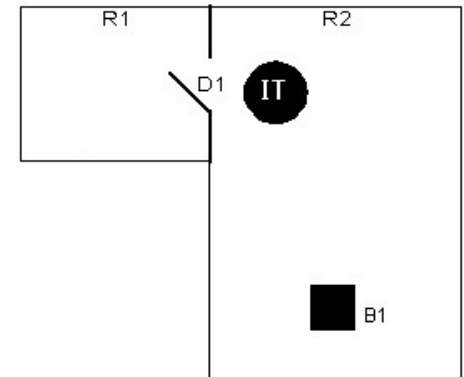
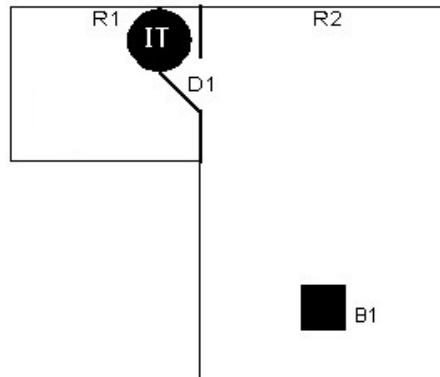
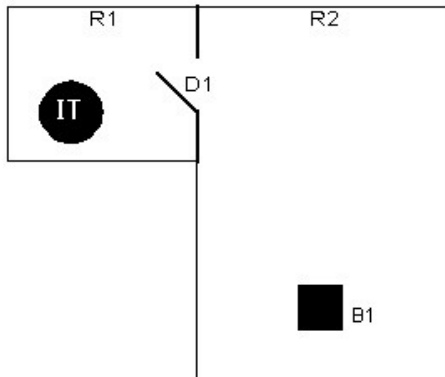
Operador	Precondición	add-list	delete-list
OP1:GOTODOOR(IT,dx)	INROOM(IT,rk) CONNECTS(dx,rk,rm)	NEXTTO(IT,dx)	
OP2:GOTHRUDOOR(IT,dx)	CONNECTS(dx,rk,rm) NEXTTO(IT,dx) STATUS(dx,OPEN) INROOM(IT,rk)	INROOM(IT,rm)	INROOM(IT,rk)

Si $rm=R2$ el resultado de aplicar OP2 será
INROOM(IT,R2).

Antes de aplicar OP2 Strip debe verificar las
precondiciones.

Strip

Estado inicial	Estado luego de aplicar el OP1	Estado luego de aplicar el OP2
INROOM(IT,R1) INROOM(B1,R2) CONNECTS(D1,R1,R2) CONNECTS(D1,R2,R1) STATUS(D1,OPEN)	INROOM(IT,R1) INROOM(B1,R2) CONNECTS(D1,R1,R2) CONNECTS(D1,R2,R1) STATUS(D1,OPEN) NEXTTO(IT,D1)	INROOM(IT,R2) INROOM(B1,R2) CONNECTS(D1,R1,R2) CONNECTS(D1,R2,R1) STATUS(D1,OPEN) NEXTTO(IT,D1)



Algoritmo Strip

Computar la diferencia entre el estado objetivo y el estado inicial, utilizando la función de evaluación de diferencia. Si no hay diferencias termina.

Si hay diferencia, reduce la diferencia seleccionando el primer operador de la lista cuya add-list incluya un predicado que niegue la diferencia.

Examina las precondiciones para determinar si existe un conjunto de variables que las evalúe verdaderas. Si no, toma la primera precondición falsa, la define como nuevo objetivo y almacena el objetivo original en un stack. Recursivamente reduce esa diferencia aplicando los pasos 2 y 3.

Cuando todas las precondiciones para un operador son verdaderas, coloca el operador en el stack de planificación y actualiza una copia del modelo del mundo. Luego selecciona el operador anterior pudiendo aplicarlo o volver a 3.

Closed World Assumption

El modelo del mundo contiene todo lo que el robot necesita saber.

Si esta condición no se cumple el robot no podrá realizar la tarea asignada.

Lo opuesto a Closed World Assumption es conocido como Open World Assumption.

Strips asume closed world.

Frame Problem

El problema de representar una situación real del mundo de forma que sea computacionalmente tratable.

Como determinar eficientemente que las cosas siguen siendo iguales en un mundo que cambia.

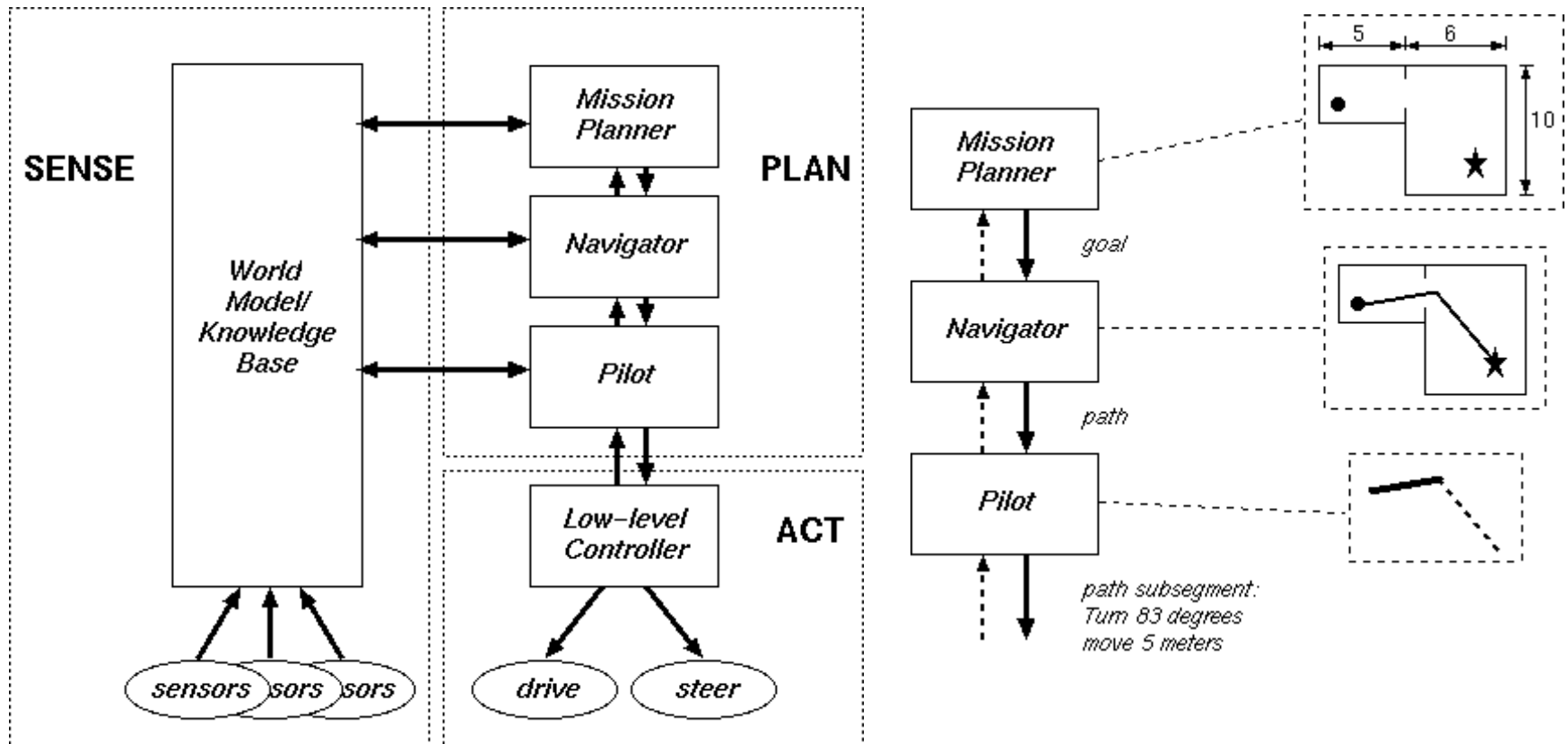
Strip sufre el problema del marco.

Arquitecturas Representativas

Nested Hierarchical Controller
Realtime Control System

Nested Hierarchical Controller (1/2)

La arquitectura NHC fue desarrollada por A. Meystel en 1993.



Nested Hierarchical Controller (2/2)

Ventajas

Intercala Planificación con la ejecución de comandos. El robot obtiene un plan, comienza a ejecutarlo, luego lo cambia si el mundo es diferente del que esperaba.

Desventajas

La descomposición de la etapa de planificación es específica para navegación, difícilmente generalizable a otras tareas.

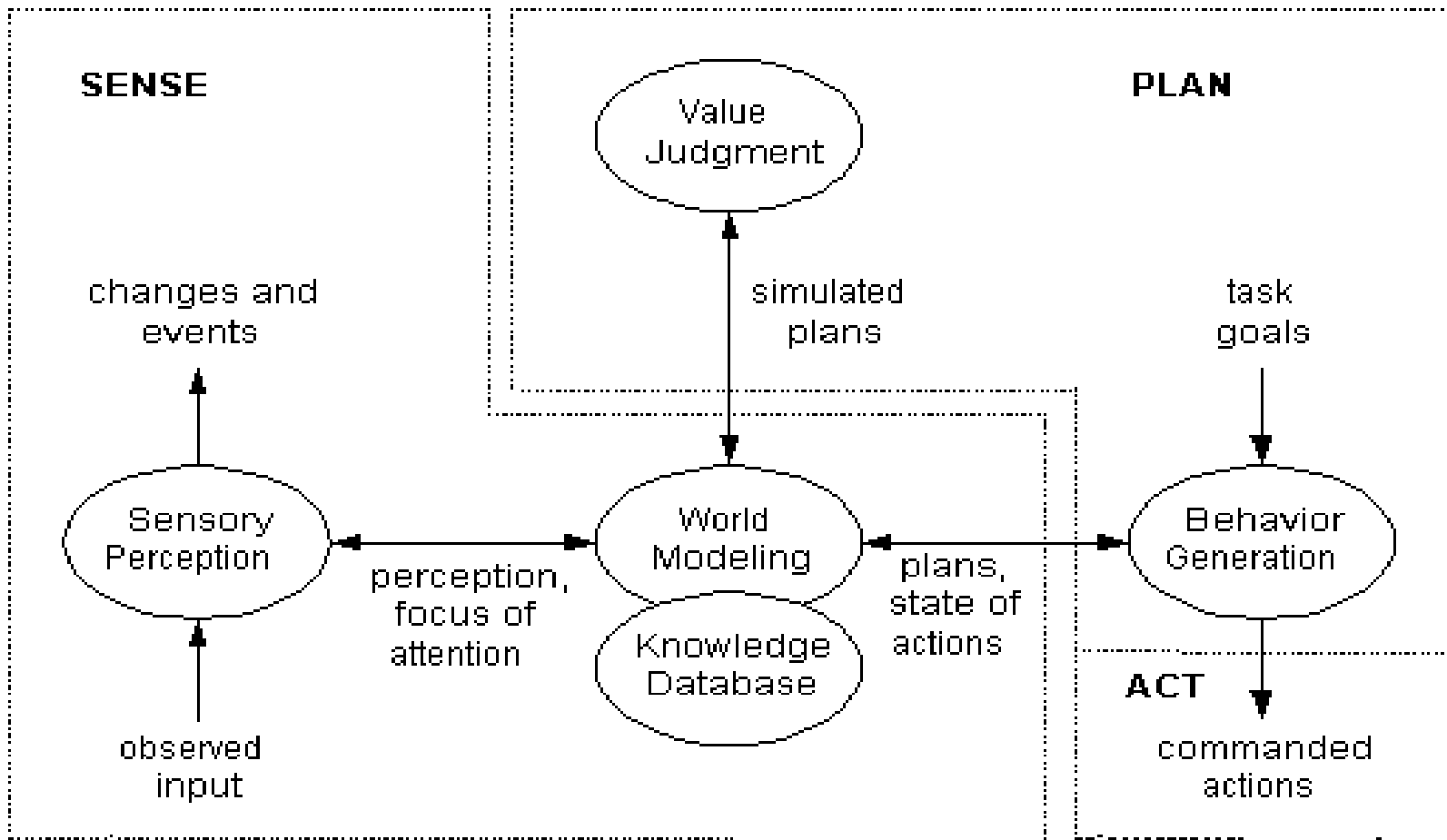
No hay muchas implementaciones de esta arquitectura sobre robots reales.

Real-Time Control System

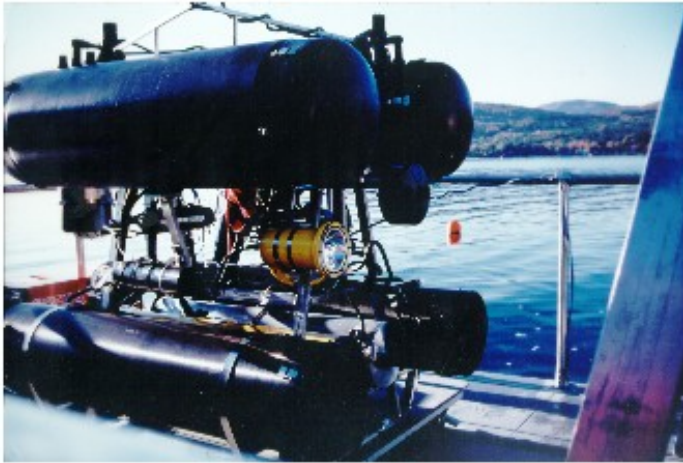
La arquitectura Real-Time Control System, fue desarrollada como guía para controlar robots industriales agregándole mayor inteligencia.

Desarrollada por J. Albus en el National Institute of Standards and Technology (NIST).

Real-Time Control System



Ejemplos de RCS



Evaluación de las arquitecturas

La arquitectura NHC se compone del Mission Planner, Navigator y Pilot, la misma se centra principalmente en navegación mientras que RCS parece ser de más amplia aplicación.

Ambas han sido utilizadas para guiar vehículos, por lo que ambas tienen un lugar de aplicación.

La portabilidad a otros dominios no es clara.

En términos de robustez, la arquitectura RCS no pretende atacar el tema. Asume que el módulo Value Judgement simula el plan de forma de confirmar el éxito de ejecutarlo, esto es un mecanismo muy limitado de robustez.

Ventajas y Desventajas_(1/2)

Los robots construidos antes de los '90 presentaban una organización del software jerárquica.

Se desarrollaban para aplicaciones específicas, en lugar de pensar en arquitecturas genéricas que puedan ser utilizadas en otras aplicaciones.

La principal ventaja del paradigma jerárquico es que proporciona un orden entre las actividades de sensor, planificar y actuar.

Ventajas y Desventajas_(2/2)

La presencia de un modelo global del mundo está relacionado al problema del marco. En STRIP, una tarea tan simple como abrir una puerta obliga a razonar sobre todo el modelo (incluyendo hechos irrelevantes).

Incertidumbre:

- Semántica.

- Ruido en los sensores.

- Error en los actuadores.

- Al ejecutar una acción.

Consideraciones de Programación

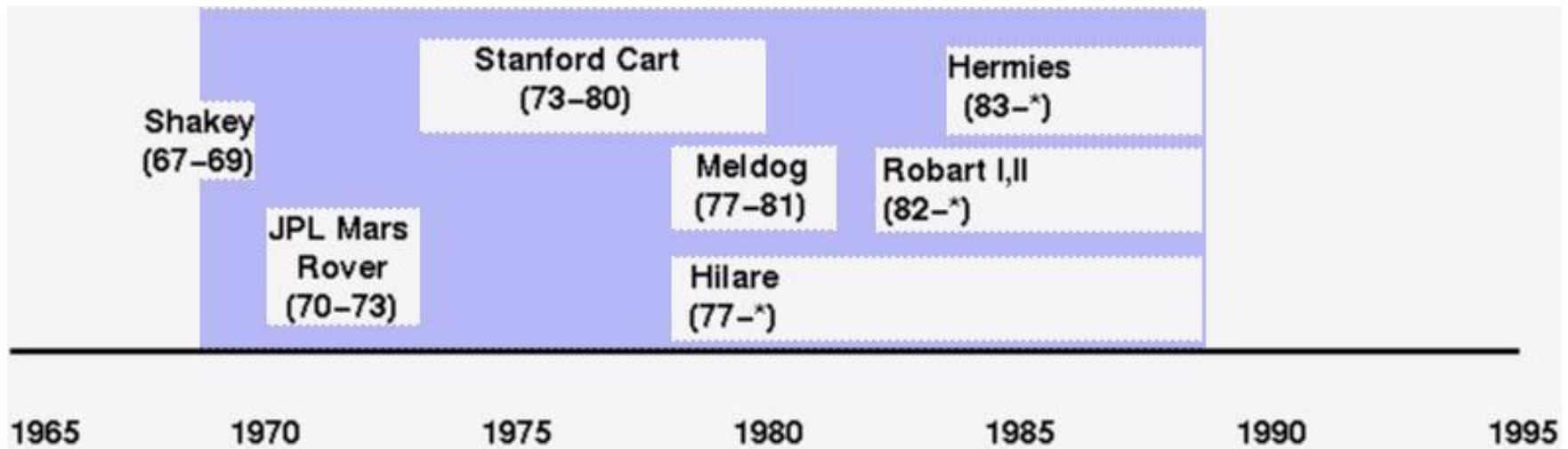
El uso de predicados y recursión favorece el uso de los lenguaje Lisp y Prolog.

Los lenguajes de este tipo no son recomendados para aplicaciones de tiempo real.

Las arquitecturas jerárquicas tienden a la programación monolítica en lugar de OOP.

En particular las arquitecturas NHC y RCS no proporcionan guías para el desarrollo de módulos reutilizables.

Robots dentro del paradigma jerárquico



Resumen

La principal desventaja es la planificación. En cada ciclo de actualización el robot debe actualizar el modelo del mundo y planificar.

Los algoritmos de planificación y sensado son muy lentos (cuello de botella).

Además, el sensar y actuar están desconectados, lo cual elimina cualquier tipo de acción estímulo-respuesta presentes en los seres vivos.

Paradigma Reactivo

Organización

Introducción

Descomposición Horizontal vs Vertical

Comportamientos

Arquitecturas

Subsuption

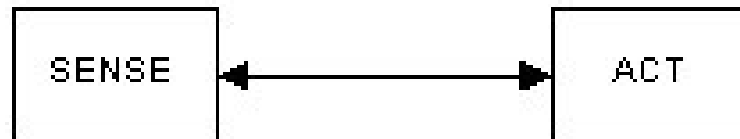
Campos de potencial

Introducción

Reactivo (1988-1992)

Nace a partir de las desventajas de los sistemas jerárquicos y bajo la influencia de la Etología y Psicología Cognitiva.

Elimina totalmente la planificación.



Forma la base para los sistemas híbridos.

Introducción

Primitiva robótica	Entrada	Salida
Sensar (SENSE)	Datos de los sensores	Información sensada
Actuar (ACT)	Información sensada [Redacted]	Comandos a los actuadores

Introducción

Arkin, Brooks y Payton trabajan en la definición de comportamientos y mecanismos de selección cuando se activan múltiples comportamientos en simultáneo.

El paradigma reactivo fue rechazado inicialmente por los organismos militares:

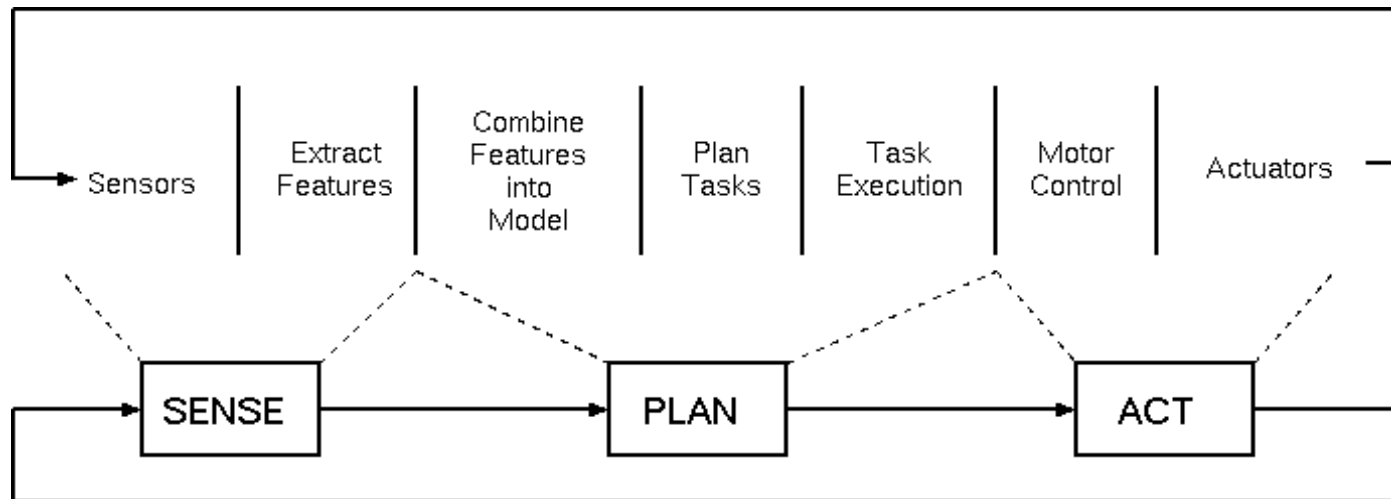
- No es claro el mecanismo por el cual emerge el comportamiento.

- Pruebas matemáticas de correctitud.

Los buenos tiempos de ejecución fueron clave para su aceptación.

Descomposición horizontal

Propuesta en los sistemas jerárquicos.



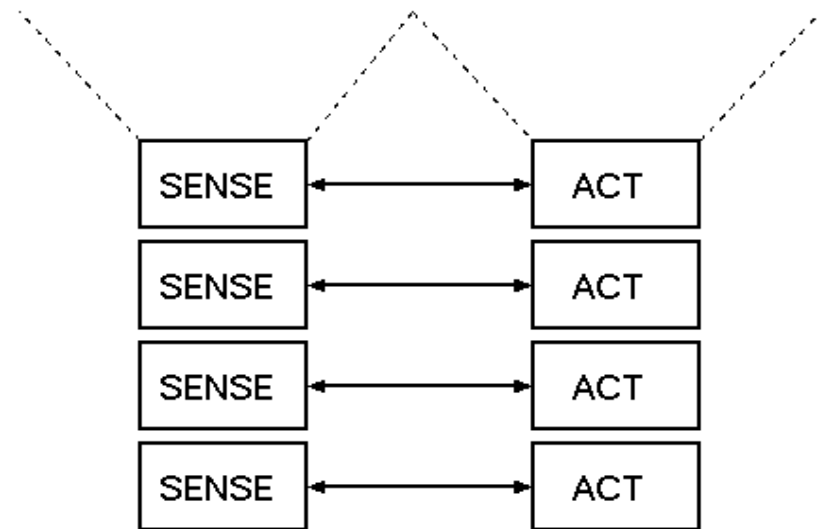
Descomposición vertical

Esta descomposición se asocia a los sistemas Reactivos.

Comportamientos de bajo nivel asociados a instinto de supervivencia.

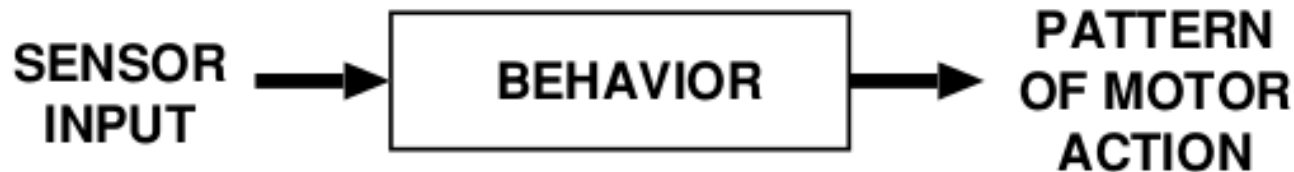
Comportamientos de alto nivel para resolver tareas complejas.

Es necesario definir un mecanismo para determinar la acción a tomar.



Definición de comportamiento

- Un comportamiento es una asignación de estímulos sensoriales a un patrón de acción motora que se utiliza para realizar una tarea.



Tipos de comportamientos

- **Reflexivo**
 - Estimulo-respuesta (S-R)
- **Reactivo**
 - Aprendido o memoria muscular
- **Consciente**
 - Deliberativo

Advertencia acerca de sobrecarga de términos:
Roboticistas a menudo usan el término “comportamiento reactivo” para referirse a comportamientos puramente reflexivos, y hacen referencia a comportamientos reactivos utilizando el término skill

Comportamiento reflexivo

- No implica ningún tipo de proceso cognitivo.
- Elimina cómputo y garantiza buenos tiempos de ejecución.
- Categorías:
 - Reflejos, la respuesta sólo dura mientras el estímulo, y la respuesta es proporcional a la intensidad del estímulo.
 - Taxes, en donde la respuesta es moverse en una orientación particular.
 - Patrones acción fija, donde la respuesta continúa por más tiempo que el estímulo.
- Las categorías anteriores no son mutuamente excluyentes.

Cuatro formas de adquirir un comportamiento

- Innato, nacer con un comportamiento.
- Secuencia de comportamientos innatos.
- Innatos con memoria
- Aprendidos

Atributos

Todas las acciones son tomadas a través de comportamientos.

Comportamiento: es un mapeo directo entre entradas de sensores a patrones de acciones que son usadas para realizar una tarea.

Organización SENSE-ACT.

Características

Ejecutan muy rápido.

No tienen memoria.

Robots como agentes situados en un medio ecológico particular.

No existen “controladores”. El comportamiento global *emerge* a partir de comportamientos básicos.

El sensado es local a cada comportamiento. No utilizan un modelo del mundo.

Siguen buenos principios de desarrollo de software.

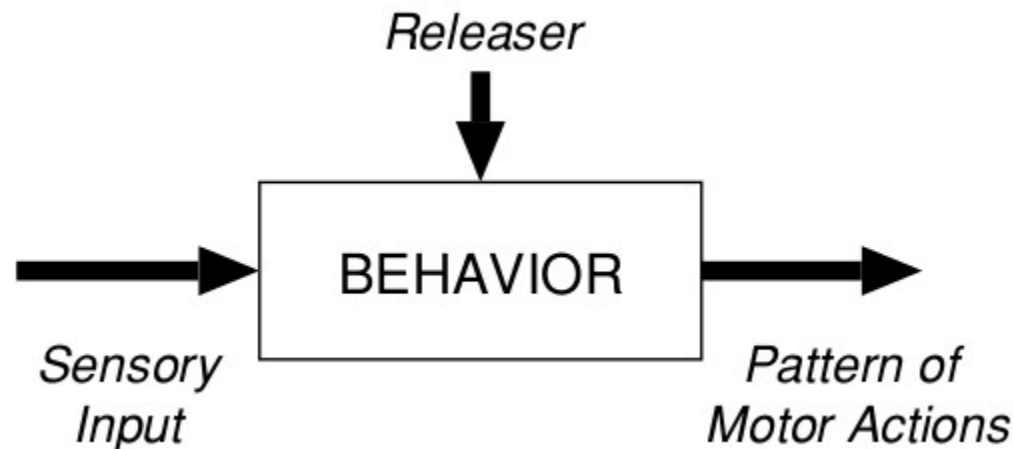
Modelos animales son citados como base para el sistema o un comportamiento en particular.

Innate Releasing Mechanisms

- Lorenz y Tinbergen presentan su trabajo IRM que trata de aclarar cómo los comportamientos se coordinan y controlan.
- IRM presupone que hay un estímulo específico (interno o externo) que libera o desencadena, el modelo estereotipado de la acción.
- El IRM activa el comportamiento.

Liberador (releaser)

- El liberador actúa como una señal de control para activar un comportamiento.
- Si un comportamiento no se libera, no responde a los estímulos sensoriales y no produce salida motora.



Esquema

El término esquema es usado en ciencias cognitivas o en etología para referirse a una manera particular de organización para percibir y actuar en una situación o ante un conjunto de estímulos.

Es genérico equivalente a una clase en OOP:

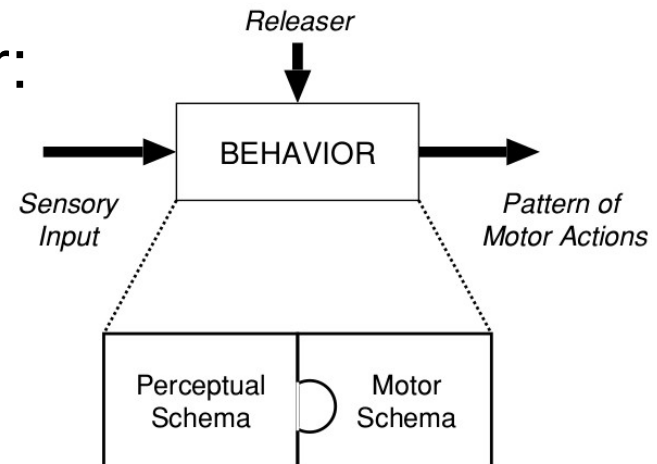
Conocimiento específico (atributos).

Conocimiento procedural (métodos).

Un comportamiento esta formado por:

Esquema sensorial.

Esquema motor.

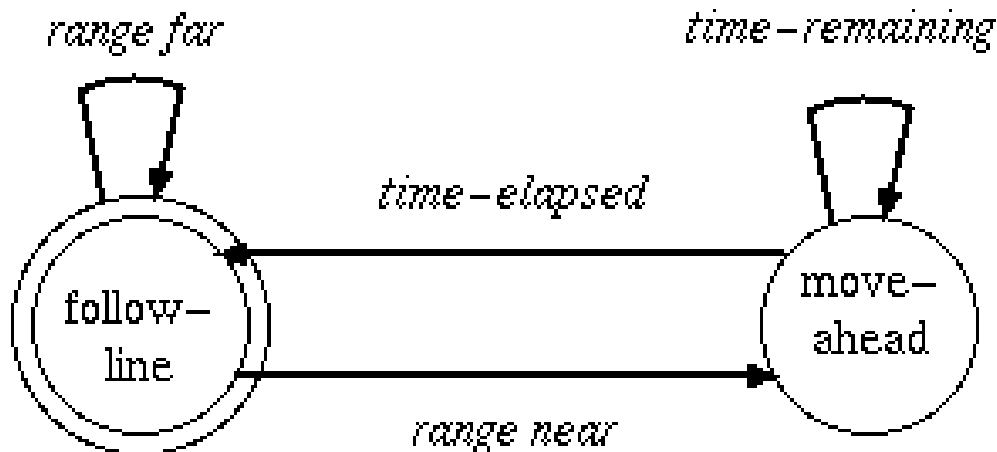


Ensamblando comportamientos

- Cómo tener en cuenta secuencia de comportamientos o comportamientos concurrentes.
- Máquinas de estado finito.

FSA: $M = \{K, \Sigma, \delta, q_0, F\}$

- K : conjunto de estados, cada estado hace referencia a un comportamiento.
- δ : función de transición (siguiente comportamiento).
- Σ : conjunto de entradas, estímulos/affordances.
- q_0 : estado inicial.
- F : conjunto de estados finales.



Comportamientos concurrentes

- Los comportamientos pueden ejecutar concurrentemente.
- Algunos comportamientos pueden ignorar el secuenciamiento cuando aparecen estímulos conflictivos en el entorno.
- Categorías de interacción:
 - Equilibrio
 - Dominancia
 - Cancelación

Dos funciones de la percepción

- Liberar un comportamiento.
- Percibir la información necesaria para llevar adelante el comportamiento.
- En ambos casos, la percepción filtra la información de acuerdo a la tarea que se está realizando (percepción orientada por la acción).
- Varios animales han evolucionado detectores que simplifican la percepción.

Arquitecturas

Una arquitectura reactiva debe proveer mecanismos para resolver:

- Cómo se activan los comportamientos.

- Determinar cómo operar cuando se activan múltiples comportamientos.

Las más conocidas y formalizadas son:

- Subsumption.

- Potential field.

- Rule encoding (Europa y Japón).

Subsumption (R. Brooks)

Un comportamiento es una red de módulos de sensado y actuación que realizan una tarea.

Los módulos son implementados como máquinas de estado finito.

Los módulos se agrupan en capas de competencia.

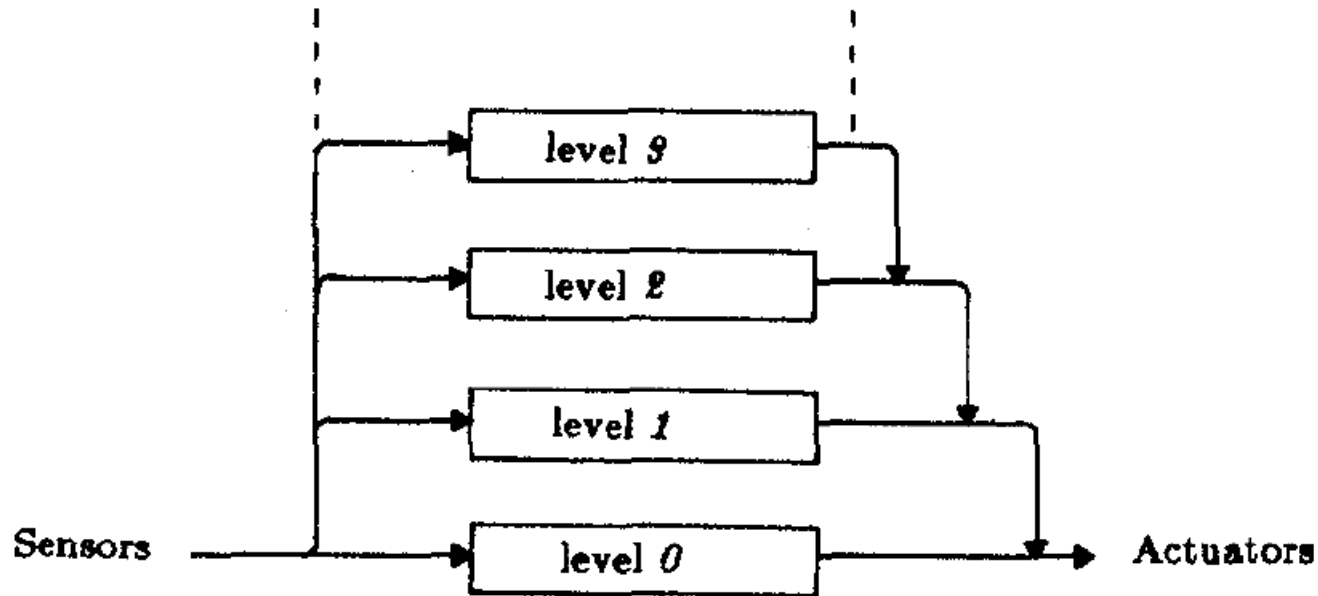
Los módulos de una capa pueden sobrescribir o subsumir la salida de un comportamiento de una capa inferior.

No utiliza mantiene ningún modelo del mundo.

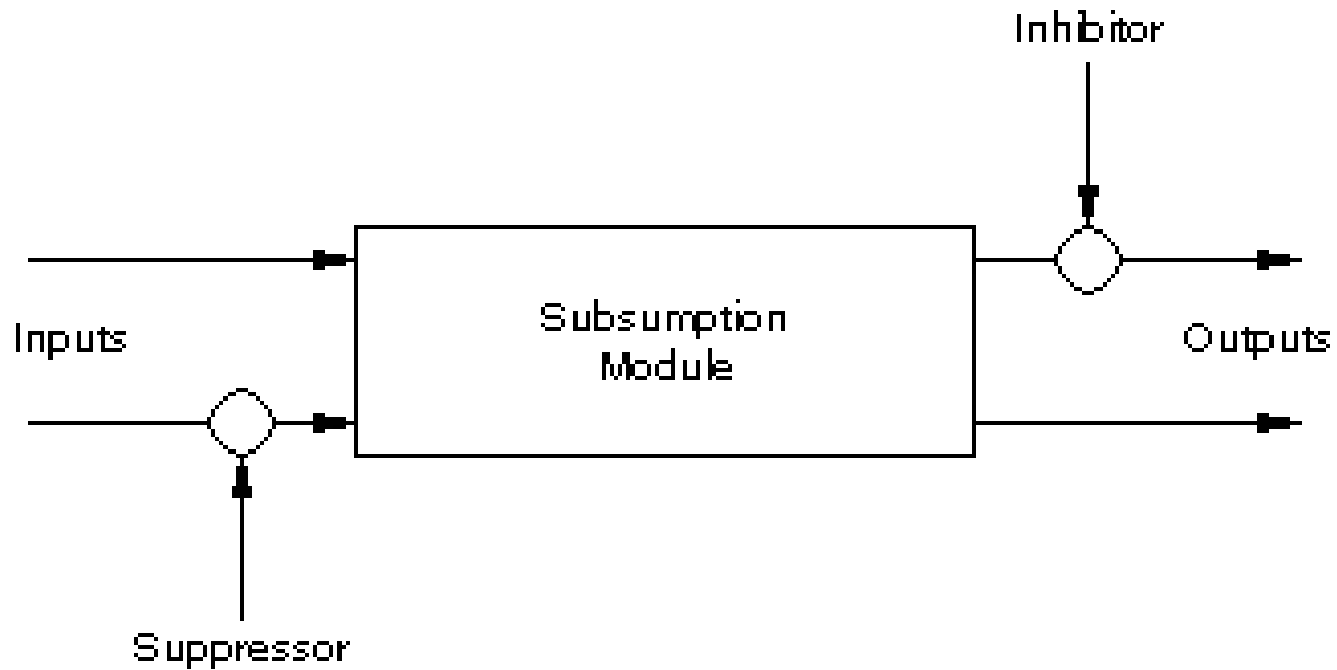
Capas

Metodología en capas.

Conecta sensores con actuadores en paralelo.

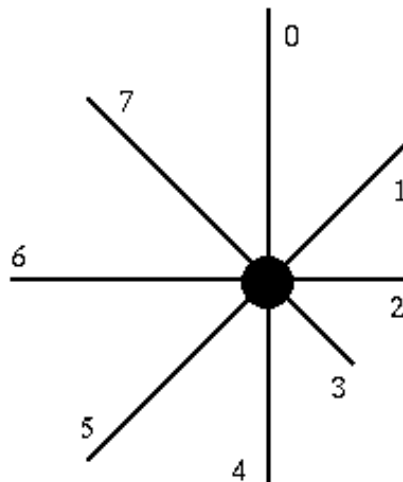


Interacción

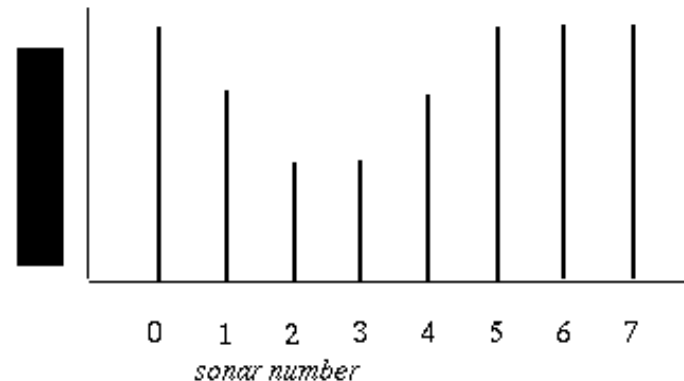


Gráfica polar

- La gráfica es egocéntrica.
- Existe una representación para almacenar la estructura de datos, pero no hay memoria ni razonamiento.

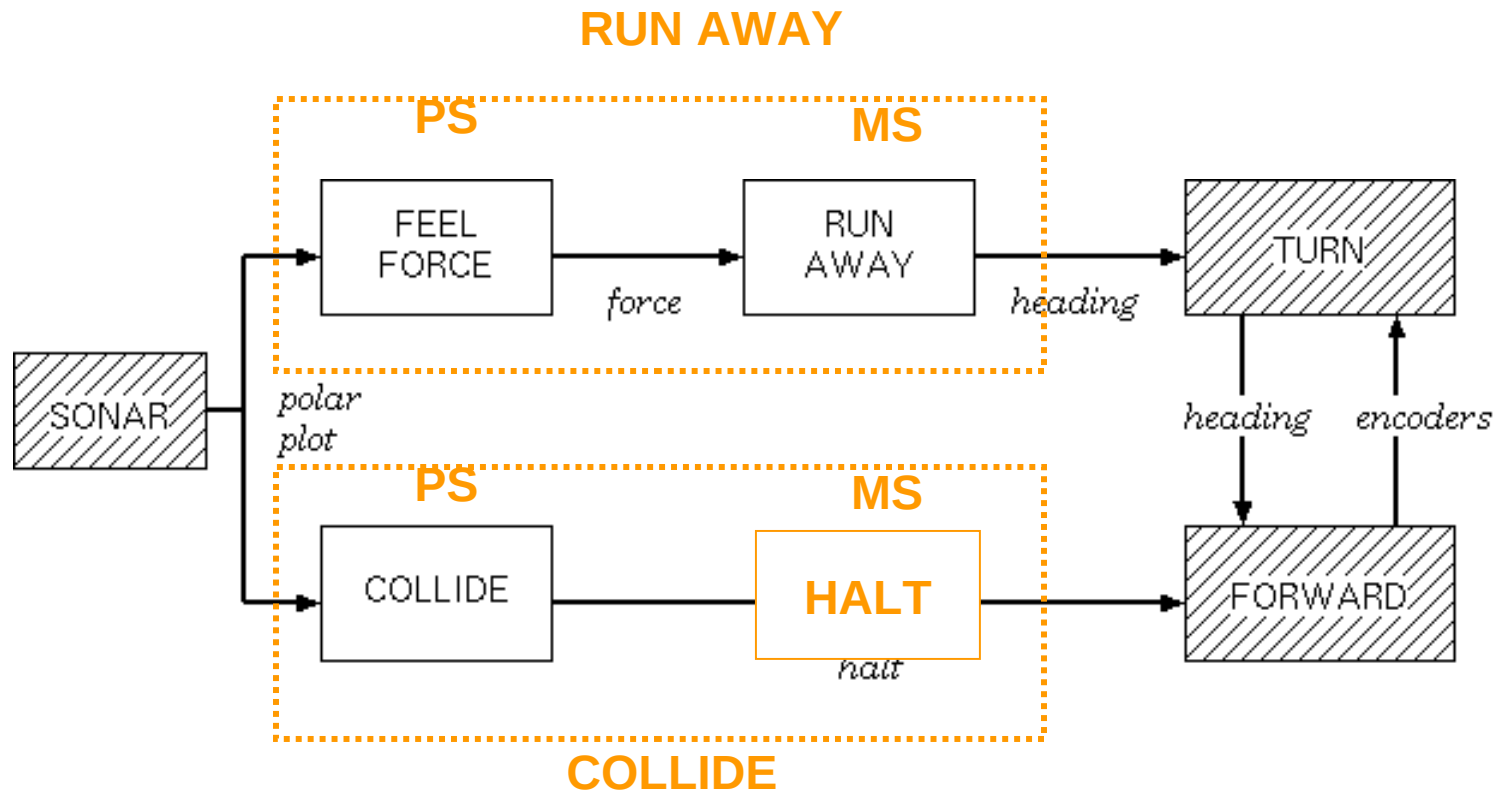


a.

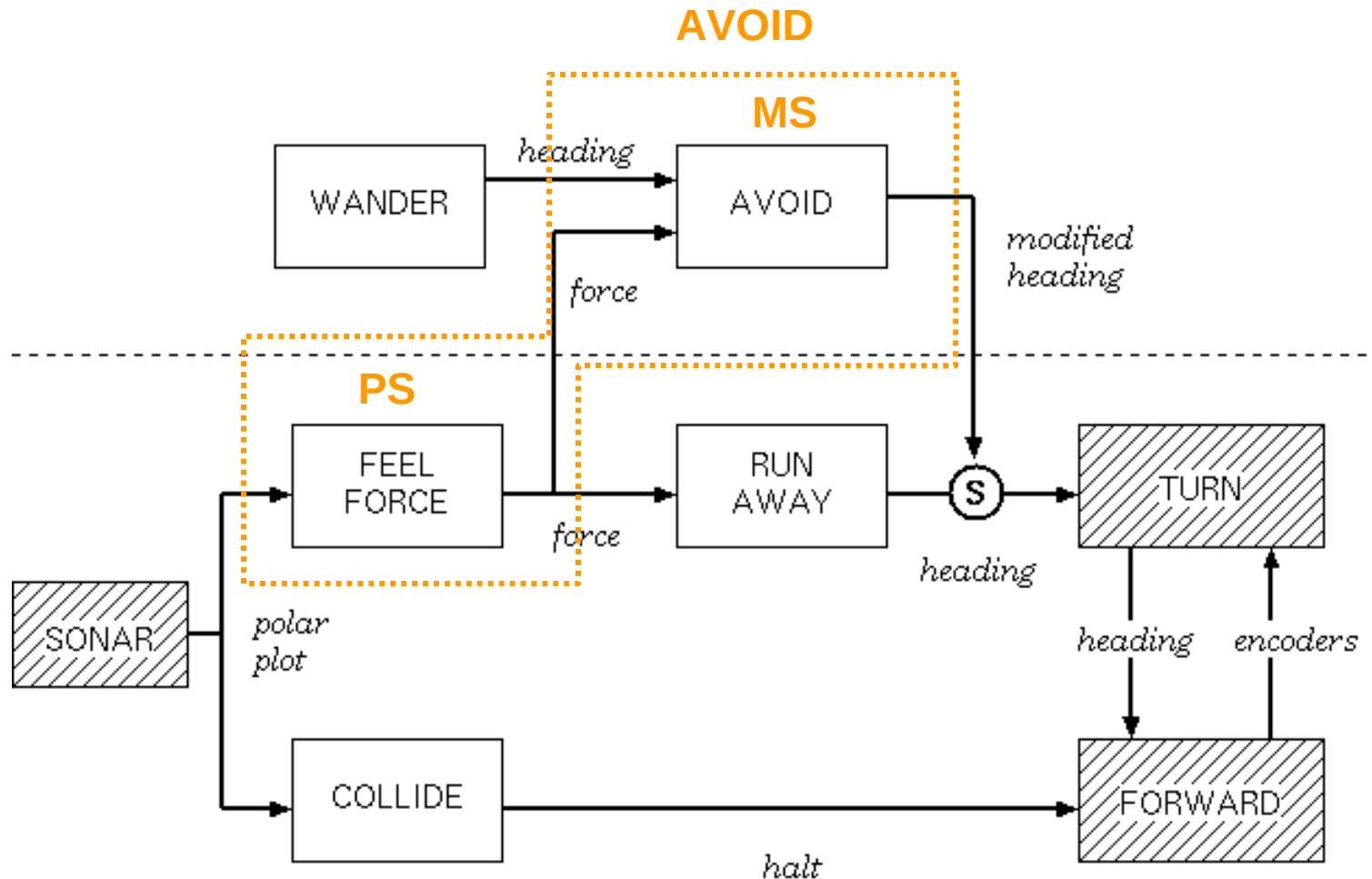


b.

Level 0: Runaway

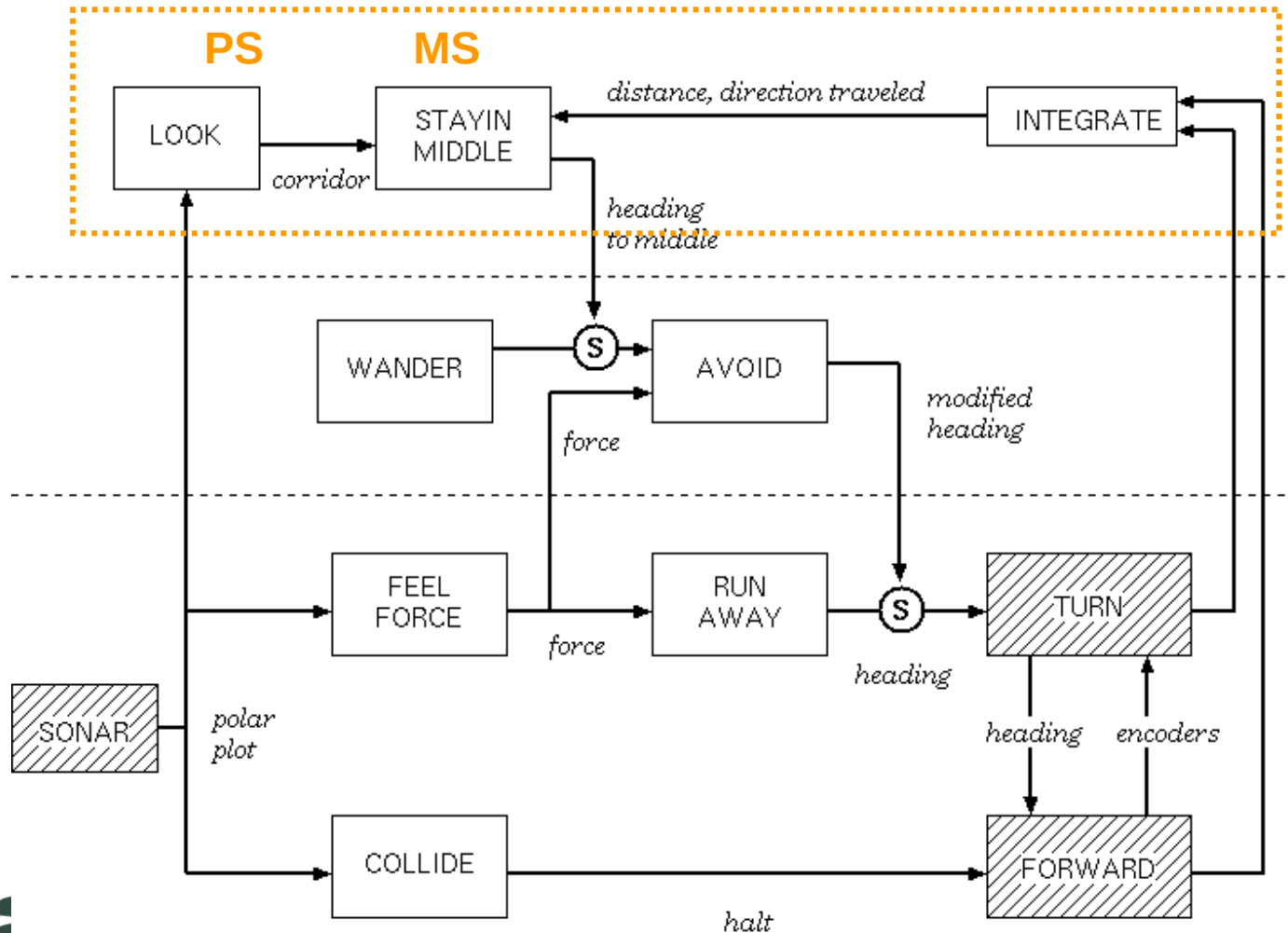


Nivel 1: Wander



Nivel 2: Follow-Corridors

STAY-IN-MIDDLE



Campos de potencial (R. Arkin)

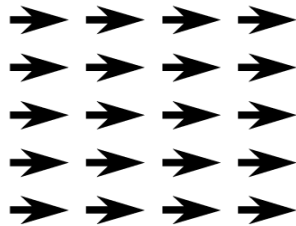
La acción a tomar debe representarse como un campo potencial.

El campo potencial puede verse como la fuerza que se realiza sobre el robot.

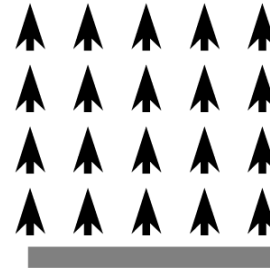
Un campo potencial es un array de vectores.

El array representa una región espacio.

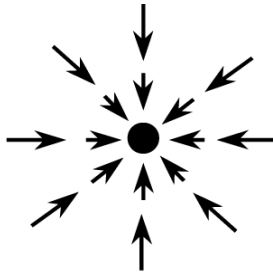
Cinco campos primitivos



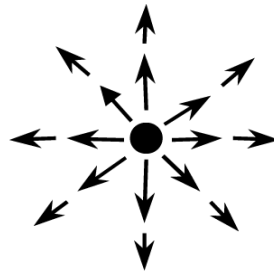
a



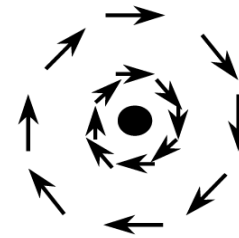
b



c



d

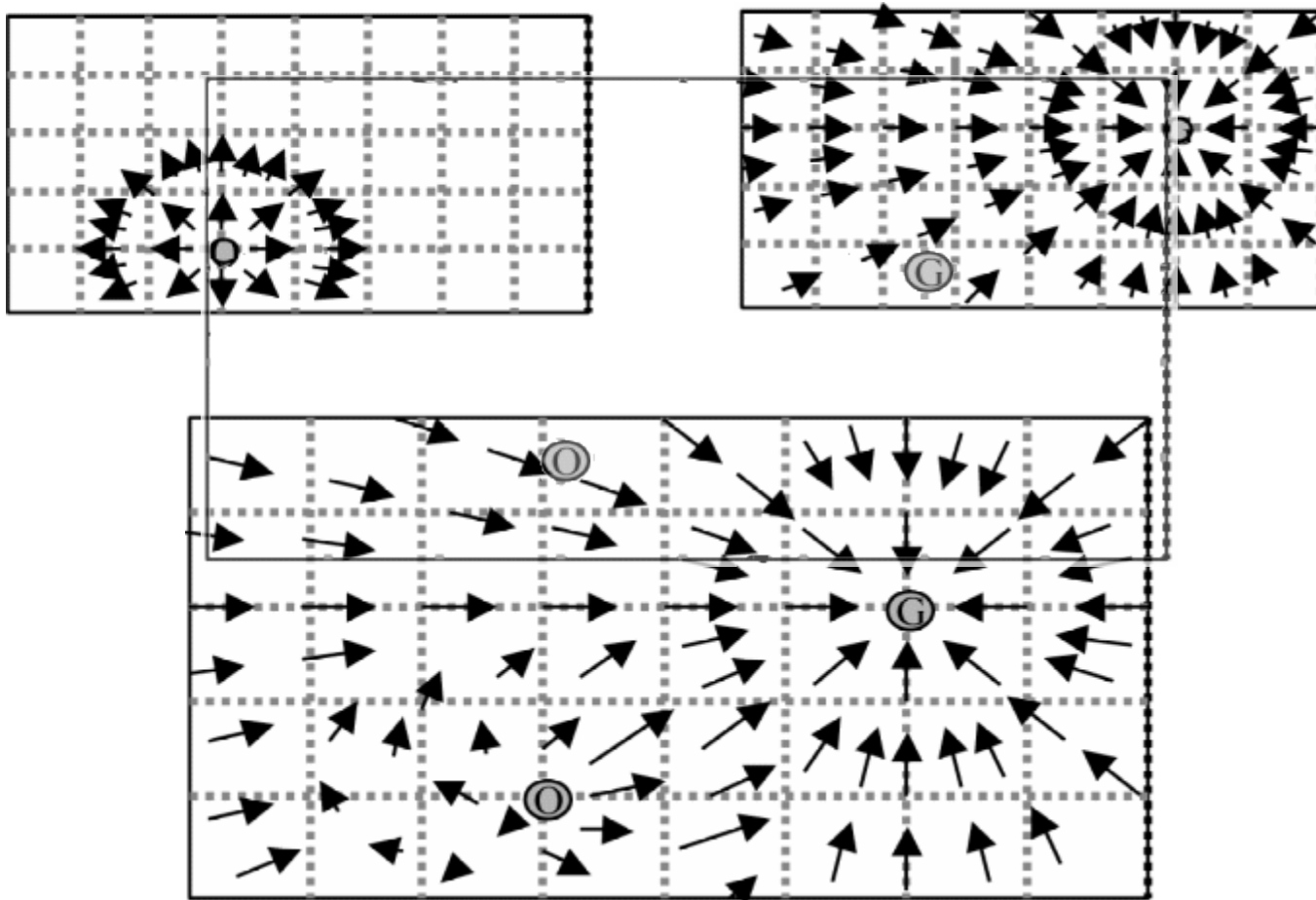


e

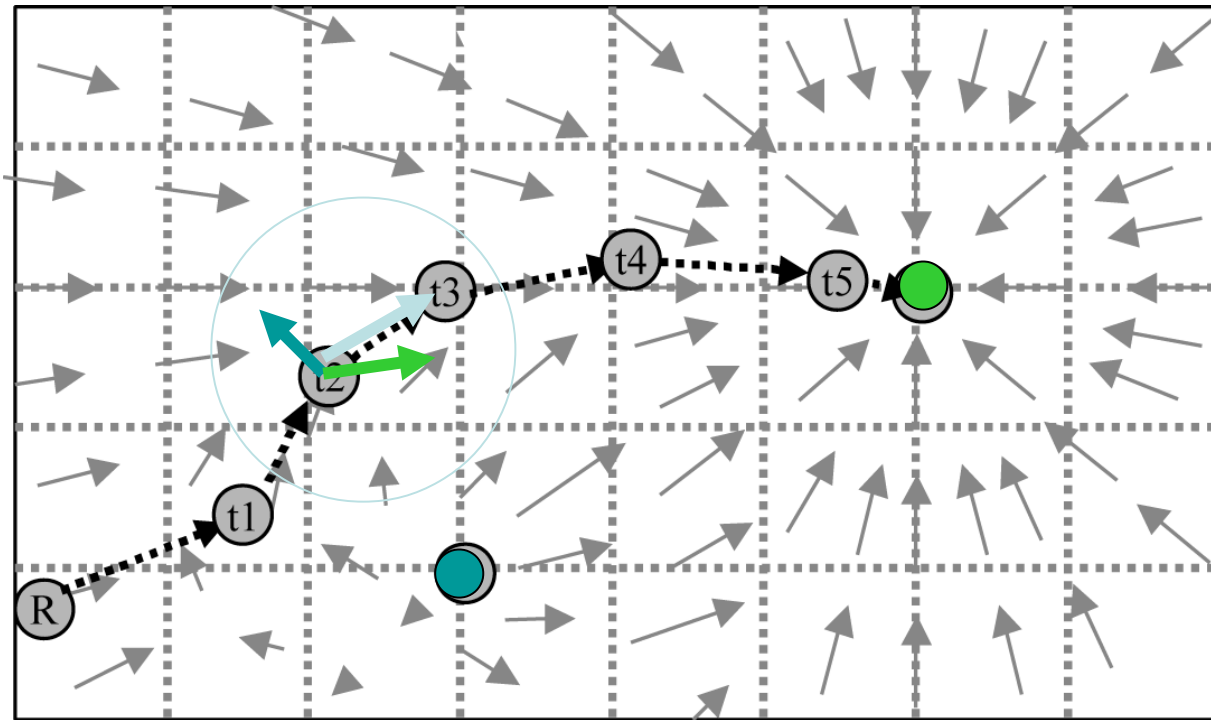
Campos y comportamientos

- Uniforme
 - Moverse en una dirección particular, seguir un corredor.
- Perpendicular
 - Evitar paredes
- Atracción
 - Moverse hacia el objetivo
- Repulsión
 - Evitar obstáculos
- Tangencial
 - Moverse a través de una puerta
- Aleatorio

Ir a una posición (1/2)

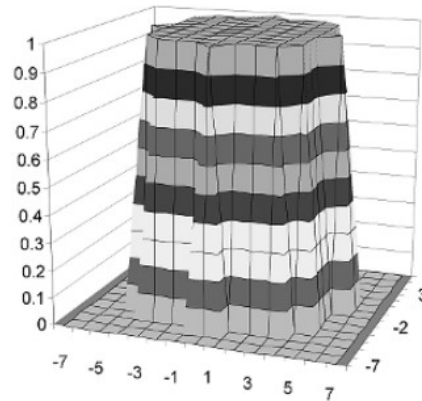


Ir a una posición (2/2)



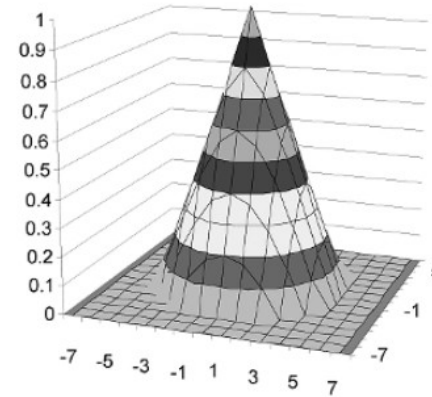
Profile de magnitud

Constant Magnitude



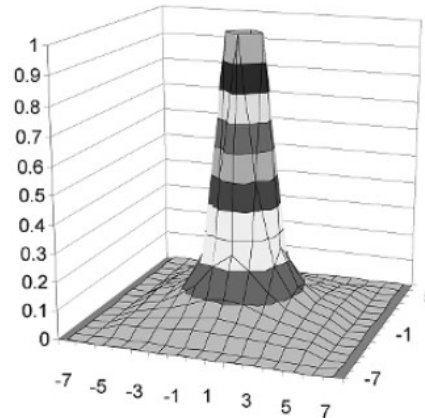
a.

Linear Dropoff



b.

Exponential dropoff



c.

Programación

- El robot computa el campo en cada ciclo sensado-actuación sin necesidad de almacenar información previa.
- Supongamos que tenemos un robot con un único sensor de distancia mirando hacia adelante. La fórmula para un campo repulsivo es:

$$V_{direction} = -180$$

$$V_{magnitud} = \begin{cases} (D - d)/D & \text{if } d \leq D \\ 0 & \text{otherwise} \end{cases}$$

Programación

```
typedef struct {
    double magnitude;
    double direction;
} vector;

vector repulsive(double d, double D) {
    if (d <= D) {
        outputVector.direction = -180;
        //turn around!
        outputVector.magnitude = (D-d)/D;
        //linear dropoff
    } else {
        outputVector.direction=0.0
        outputVector.magnitude=0.0
    }
    return outputVector;
}
```

```
vector runaway( ){
    double reading;
    reading=readSonar();
    //perceptual schema
    vector=repulsive (reading,
                      MAX_DIST);

    //motor schema
    return Voutput;
}

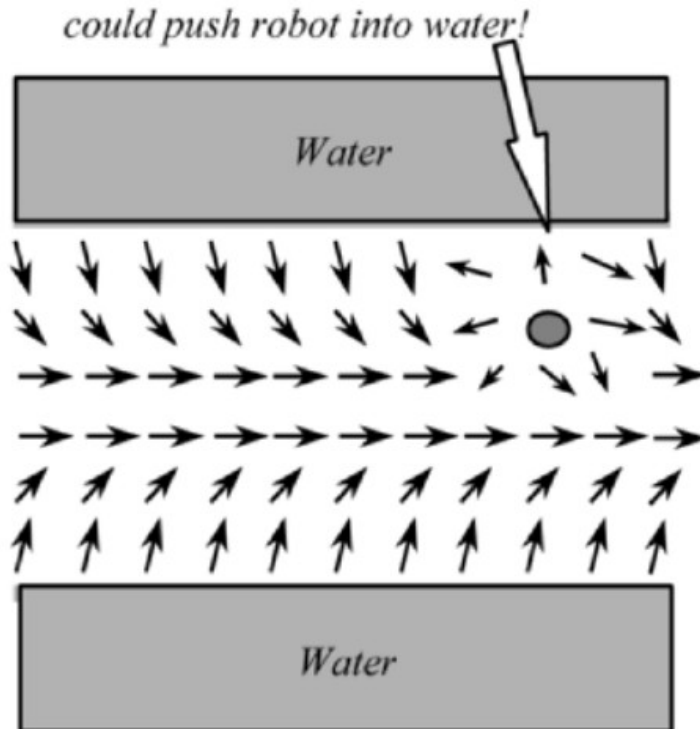
while (robot==ON) {
    Vrunaway=runaway();
    // motor schema
    turn(Vrunaway.direction);
    forward(Vrunaway.magnitude*
           MAX_VEL);
}
```

Ventajas

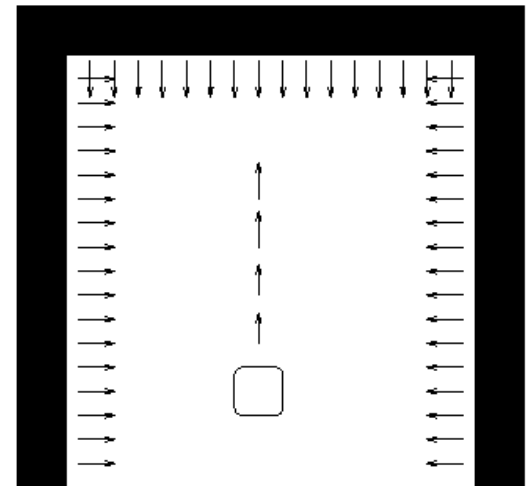
- Se puede visualizar gráficamente el comportamiento.
- Simple de implementar y combinar los comportamientos en lenguajes de programación.
- Mecanismo de combinación fijo.

Desventajas

- Múltiples campos pueden sumar cero.
- Movimientos espasmódicos.



box canyon problem



El Paradigma Híbrido

Organización

Introducción

Organización

Responsabilidades

Elementos

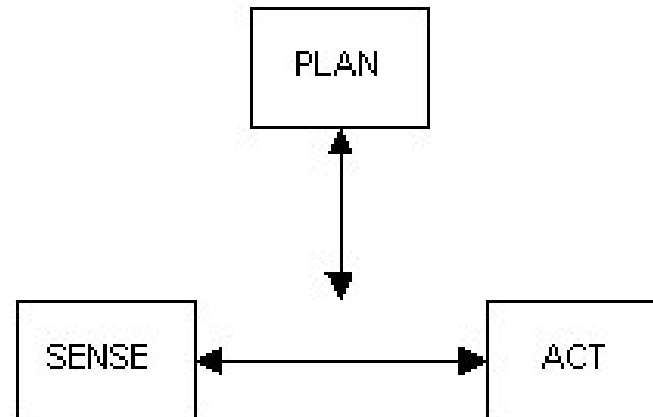
Arquitecturas

Introducción

Híbrido (1990's)

Bajo este paradigma el robot primero planifica como descomponer la tarea en subtareas y luego cuales son los comportamientos adecuados para realizar las subtareas.

Luego se ejecutan los comportamientos adecuados para cada subtarea.



Introducción

El paradigma Híbrido

Primitiva robótica	Entrada	Salida
Planificar (PLAN)	Información (sensorial o cognitiva)	Directivas
Sensar - Actuar (S-A)	Datos de los sensores	Comandos a los actuadores

Introducción

Permite controlar robots en tiempo real utilizando procesadores comerciales de bajo costo.

En 1990 los diseñadores vuelven a poner a la planificación en los robots.

En los sistemas reactivos la emergencia del comportamiento se ve como un arte y no como una ciencia.

Las arquitecturas híbridas son la solución general por varias razones:

- Usan técnicas de procesamiento asíncrono donde los módulos deliberativos funcionan independiente de los módulos reactivos.

- La modularidad y la inclusión del paradigma reactivo le dan la posibilidad de utilizar solo una porción de la arquitectura.

Introducción

La planificación cubre un gran horizonte y requiere conocimiento global.

La planificación y disponer de un modelo del mundo utilizan algoritmos costosos computacionalmente.

Organización

Reactor: parte reactiva de la arquitectura.

Deliberador.

Paradigmas en el tiempo

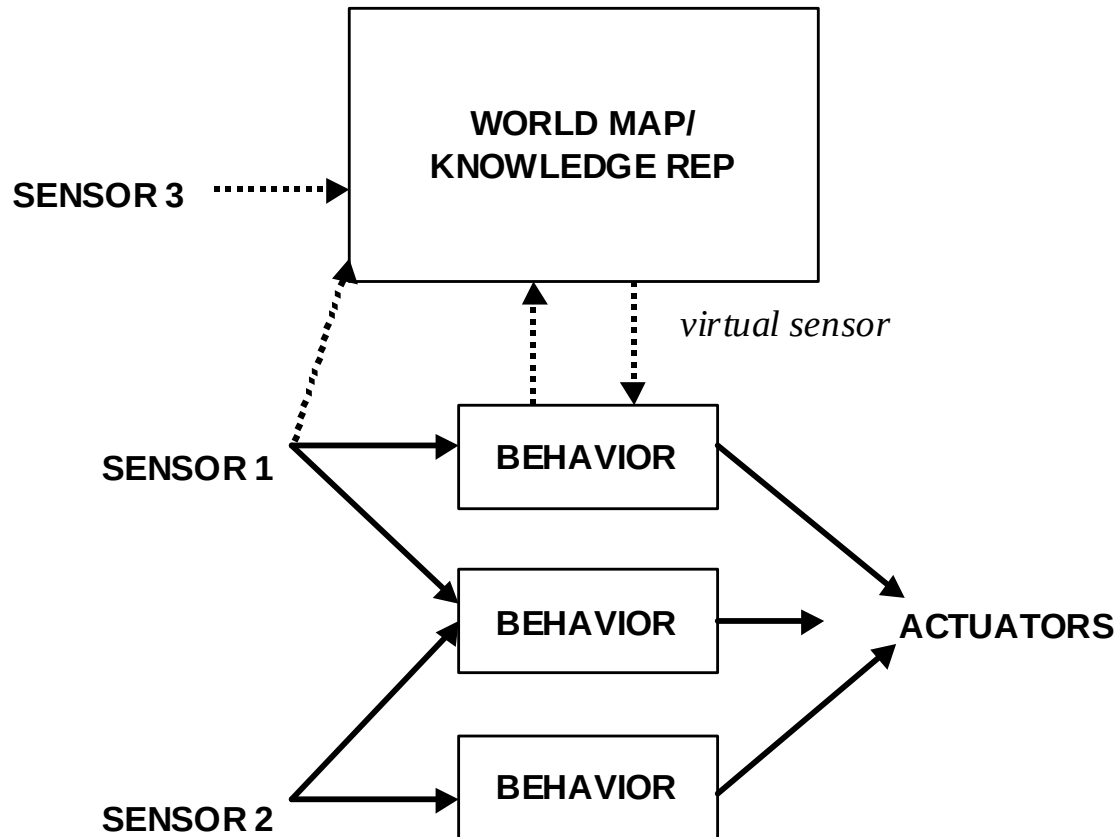
Reactivo, existe en el presente.

Deliberativo

Puede razonar sobre el pasado.

Puede proyectarse en el futuro.

Organización



Elementos

Secuenciador

Manejador de recursos

Cartógrafo

Planificador de Misión

Monitor de rendimiento y resolver de problemas.

AuRA - Autonomous Robot Architecture

Ron Arkin, Georgia Institute of Technology.

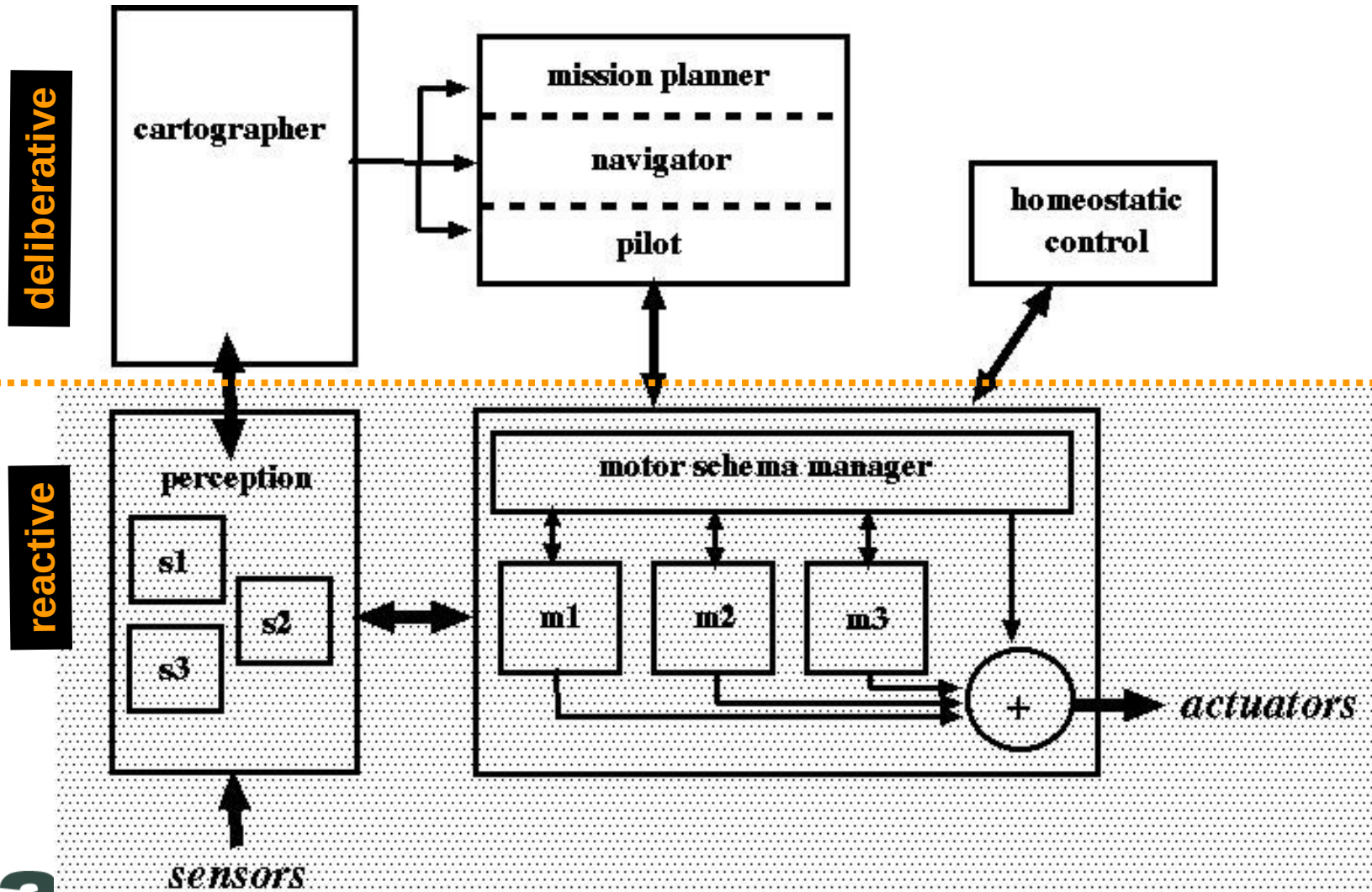
Es de las primeras arquitecturas híbridas desarrolladas.

Se desarrollo al mismo tiempo que la propuesta de Brooks.

Muchos aspectos de AuRA son motivados por la biología.

Se basa en la teoría de los esquemas.

Esquema de la arquitectura



Otras arquitecturas

SFX - Sensor Fusion Effects

3T

Saphira

Task Control Architecture

Resumen

P, S-A, la parte deliberativa usa modelos del mundo, la parte reactiva usa comportamientos.

Generalmente incluyen módulos para planificación de misión, cartógrafo, secuenciador, gestión de comportamientos y monitor de performance.

La parte reactiva tiende a usar ensamblaje de comportamientos.

Referencias

Murphy R. R., An Introduction to AI Robotics (Intelligent Robotics and Autonomous Agents), MIT Press, 2000.

Brooks, R., "A robust layered control system for a mobile robot". Robotics and Automation, IEEE Journal, 1986.

Shakey the Robot,

<http://www.sri.com/about/timeline/shakey.html>, Set 2010.

References on RCS,

http://www.eleceng.ohio-state.edu/nist_rcs_lib/ref_RCS.html

, Set 2010.

Preguntas