APUNTES AIN

TEMA 2.- AGENTES INTELIGENTES: CONCEPTOS FUNDAMENTALES

1.-DEFINICIÓN DE AGENTE

La principal característica de los agentes es que son **autónomos**, es decir, son capaces de actuar independientemente. Un **agente** es un sistema informático capaz de actuar autónomamente en algún entorno con el fin de alcanzar los objetivos que se le han delegado. Consideremos que un agente es como una entidad que está en continua interacción con su entorno.

Una mejor definición de **agente** sería la siguiente: un agente es un sistema informático, **situado** en algún entorno, que **percibe** el entorno (entradas sensibles de su entorno) y a partir de tales percepciones **determina** (mediante técnicas de resolución de problemas) y ejecuta acciones (de forma **autónoma** y **flexible**) que le permiten alcanzar sus **objetivos** y que pueden **cambiar** el entorno.

Cualquier proceso computacional dirigido por el objetivo debe de ser capaz de interaccionar con su entorno de forma **flexible** y **robusta**. Para ser flexible, el agente ha de ser **reactivo**, **proactivo** y **social**.

Un sistema **reactivo** es aquel que mantiene una constante interacción con su entorno y responde (a tiempo para que la respuesta sea útil) a los cambios que ocurren en él.

Queremos que los agentes hagan cosas por nosotros, por ello, adoptan un comportamiento dirigido por el objetivo. La **proactividad** es la capacidad de generar e intentar conseguir objetivos, no solamente dirigidos por eventos, es decir, tomar la iniciativa reconociendo oportunidades.

La **sociabilidad** o **capacidad social** en agentes es la capacidad de interactuar con otros agentes mediante **cooperación**, **coordinación** y **negociación**. La **cooperación** es la capacidad de trabajar juntos como un equipo para conseguir un objetivo compartido. La **coordinación** es la capacidad de gestionar las interdependencias entre actividades. La **negociación** es la capacidad de alcanzar acuerdos sobre temas de interés común.

Existen dos **conceptos** de agente:

- **Débil**:
    - Autonomía.
    - Proactividad.
    - Reactividad.
    - Sociabilidad.
- **Fuerte**:
    - Concepto débil.
    - Movilidad: habilidad de trasladarse en una red de comunicación informática.
    - Veracidad: no comunica información falsa intencionadamente.
    - Benevolencia: no tiene objetivos contradictorios y siempre intenta realizar la tarea que se le solicita.

    - Racionalidad: tiene unos objetivos específicos y siempre intenta llevarlos a cabo.
    - Aprendizaje/adaptación.

Un agente tiene siempre las propiedades débiles de agencia y puede tener las propiedades fuertes.

2.-ENTORNOS DE AGENTE

En entornos complejos, un agente no tiene control completo sobre su entorno, sólo tiene un control parcial. El **control parcial** significa que el agente puede influir sobre el entorno con sus acciones. Una acción ejecutada por un agente puede fallar o tener el efecto deseado. En conclusión, los entornos no son deterministas y los agentes deben estar preparados para posibles fallos.

Las **propiedades** de los entornos de agente son:

- **Accesible** (observable) vs **inaccesible** (parcialmente observable). Un entorno observable es aquel en el que el agente puede obtener información completa, exacta y actualizada del estado del entorno (los sensores perciben todos los datos que son relevantes para la toma de decisiones).
- **Determinista** vs **estocástico**. Un entorno determinista es aquel en el que cualquier acción tiene un único efecto garantizado, no hay incertidumbre sobre el estado resultante de la ejecución de una acción.
- **Episódico** vs **secuencial**. En un entorno episódico el desempeño/actuación de un agente depende de un número discreto de episodios, no existiendo enlaces (relación) entre el desempeño de un agente en escenarios distintos. Cada episodio consiste en la percepción del agente y la realización de una única acción posterior. Es muy importante saber el siguiente episodio no depende de las acciones que se realizaron anteriormente y es que en los entornos episódicos la elección de la acción en cada episodio depende sólo del episodio en sí mismo.
- **Estático** vs **dinámico**. Un entorno estático es aquel en el que se puede asumir que no se producen cambios excepto los provocados por la ejecución de acciones del agente. Un entorno dinámico es aquel que tiene otros procesos que operan en él, y que por lo tanto se producen cambios que están fuera del control del agente.
- **Discreto** vs **continuo**. Un entorno es discreto si en él hay un número fijo y finito de acciones y percepciones.

El tipo de entorno determina el tipo de agente. El mundo real es parcialmente observable, estocástico, secuencial, dinámico, continuo y multiagente.

3.-AGENTES COMO SISTEMAS INTENCIONALES

Un sistema intencional de primer orden tiene creencias y deseos, pero no creencias y deseos sobre creencias y deseos. Un sistema intencional de segundo orden es más sofisticado, tiene creencias y deseos sobre creencias y deseos, tanto los de los demás cómo los suyos mismos.

Atribuir creencias, albedrio, intenciones, conciencia, habilidades, o deseos a una máquina es correcto cuando tal atribución expresa la misma información sobre la máquina que expresa sobre una persona.

Cuanto más sepamos sobre un sistema, menos necesitamos confiar en explicaciones anímicas e intencionales de su comportamiento. Pero con sistemas muy complejos, esta explicación mecanicista no es factible. Es por eso que son necesarias abstracciones, como la **actitud intencional**.

Los **conceptos intencionales** son **herramientas de abstracción** que nos proporciona una forma cómoda y familiar de describir, explicar y predecir el comportamiento de los sistemas complejos.

La **caracterización de agentes** nos proporciona una forma familiar, no técnica, de comprender y explicar agentes.

Las **representaciones anidadas** nos brindan la posibilidad de especificar sistemas que incluyen representaciones de otros sistemas.

Con los **agentes**, proporcionamos una descripción de alto nivel del objetivo delegado, y dejamos que el mecanismo de control deduzca qué hacer, sabiendo que actuará según una teoría integrada de agencia racional.

4.-ARQUITECTURAS ABSTRACTAS PARA AGENTES INTELIGENTES

Asumimos que el **entorno** pude estar en uno cualquiera de los estados de un conjunto finito de estados instantáneos discreto (E={s1,s2,…}). Se asume que los agentes tienen disponible un repertorio (conjunto finito) de posibles **acciones** que transforman el estado del entorno (Ac={α1,α2,…}).

La **ejecución**, r, de un agente en un entorno representa la interacción entre el agente y su entorno, es una secuencia de estados de entorno y acciones intercaladas.

R es el conjunto de todas las secuencias finitas posibles sobre E y Ac.

$\mathbf{R^{Ac}}$ es el subconjunto de estas secuencias que finalizan con una acción.

$\mathbf{R^{E}}$ es el subconjunto de estas secuencias que finalizan con un estado del entorno.

Los entornos pueden ser modelado mediante una función de transición de estado que representa el comportamiento del entorno:

$$\tau: \mathcal{R}^{\mathcal{A}} \rightarrow \mathcal{P}(E)$$

donde $\mathcal{P}(E)$ es el conjunto de partes de E.

Esta función tiene como argumento una secuencia estado-acción (r) que finaliza en una acción, y devuelve como resultado un conjunto de estado del entorno, es decir, representa el efecto que las acciones de un agente tienen sobre el entorno. Su significado es que como resultado de ejecutar la última acción de la secuencia el entorno puede encontrarse en cualquiera de los estados $\tau(r)$.

Debemos observar que los entorno son dependientes de la historia y no son deterministas. Si $\tau(r)$=0 no hay posible estado sucesor de r, es decir, la ejecución ha finalizado. Asumimos que eventualmente todas las ejecuciones finalizan. Un entorno Env es una tripleta <E, s0, τ> donde E es el conjunto de estados del entorno, s0 es el estado inicial y τ es función de transición.

Necesitamos un modelo de agente que habite en un sistema. Asumimos un modelo donde el agente decide que acción ejecutar basándose en su historia. Un agente puede ser visto como una función que relaciona ejecuciones con acciones:

$$Ag: \mathcal{R}^E \rightarrow Ac$$

donde $\mathcal{R}^E$ es una ejecución que representa la historia previa.

Asumimos que los entornos son implícitamente estocásticos, sin embargo, los **agentes son deterministas**.

Un sistema es un par que contiene un agente y un entorno. Cualquier sistema tendrá asociado un conjunto de posibles ejecuciones. Denotaremos el conjunto de posibles ejecuciones de un agente **Ag** en un entorno **Env** por $\mathcal{R}$(Ag, Env), que contiene solo ejecuciones que han finalizado.

Dos agentes Ag1 y Ag2 tienen un comportamiento equivalente en un entorno Env si y solo si $\mathcal{R}$(Ag1, Env) = $\mathcal{R}$(Ag2, Env), es decir, si tienen un comportamiento equivalente en todos los entornos.

Un **agente reactivo** decide que acción ejecutar sin tener en cuenta su historia considerando solamente el presente. Formalmente el comportamiento de un agente reactivo se representa mediante una función:

Acción: E -> A

Para cualquier agente reactivo hay un modelo de agente como el definido previamente, lo inverso no es generalmente cierto.

La función **percibir** modela la capacidad del agente para percibir su entorno, mientras que la función **actuar** modela el proceso de toma de decisión del agente.

La **percepción** es el resultado de la función:

Percibir: E -> P

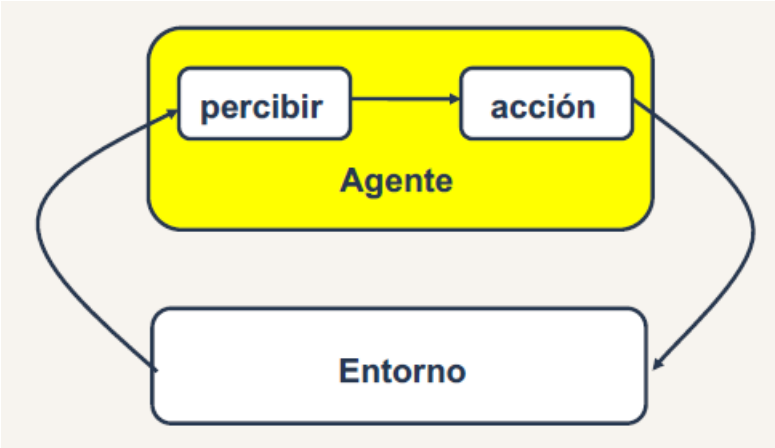
donde P es un conjunto (no vacío) de percepciones, que relacionan estados del entorno con percepciones y E es el conjunto de los estados percibidos diferentes. Un agente sería omnisciente si |E| = |S|.

La decisión del agente es el resultado de la función **acción**:

Acción: P* -> A

que relaciona secuencias de percepciones (P*) con acciones.

Dos estados diferentes s1∈S y s2∈S (s1≠s2) son indistinguibles si percibir(s1) = percibir(s2).



Los **agentes con estado** tienen una estructura de datos interna que es utilizada para recordar información sobre la historia y estado del entorno. Tienen la misma función de percepción. La función acción-selección es definida ahora como un mapping entre estados internos y acciones:

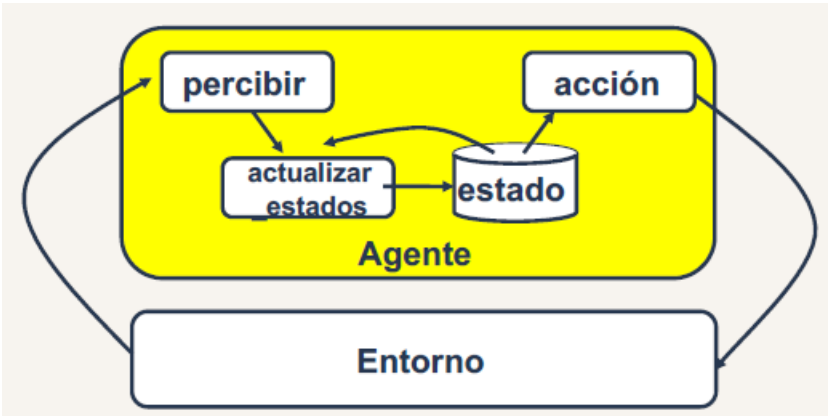
Acción: I -> Ac

donde I es el conjunto de todos los estados internos del agente.

Se introduce una nueva función de siguiente estado que define un estado interno a partir de un estado interno y las percepciones:

Actualizar_estados: I x P -> I

Esta función determina el nuevo estado interno a partir del estado interno en que se encuentre el agente y las percepciones que perciba en su entorno.



5.-¿CÓMO DECIRLE A UN AGENTE LO QUE TIENES QUE HACER?

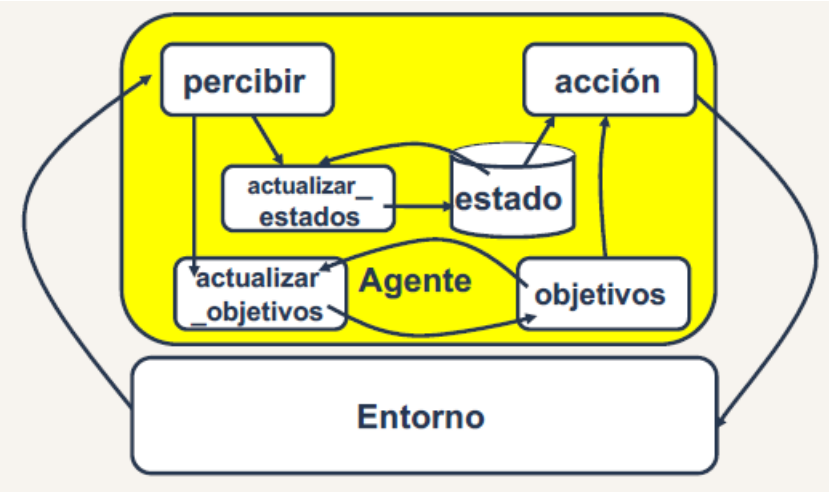
Los **agentes con objetivos** tienen la misma estructura y funciones que los agentes con estado. Se introduce un nuevo conjunto, el conjunto de objetivos G. Se introduce una nueva función para actualizar los objetivos del agente a partir de las percepciones del mismo:

Actualizar_objetivos: G x P -> G

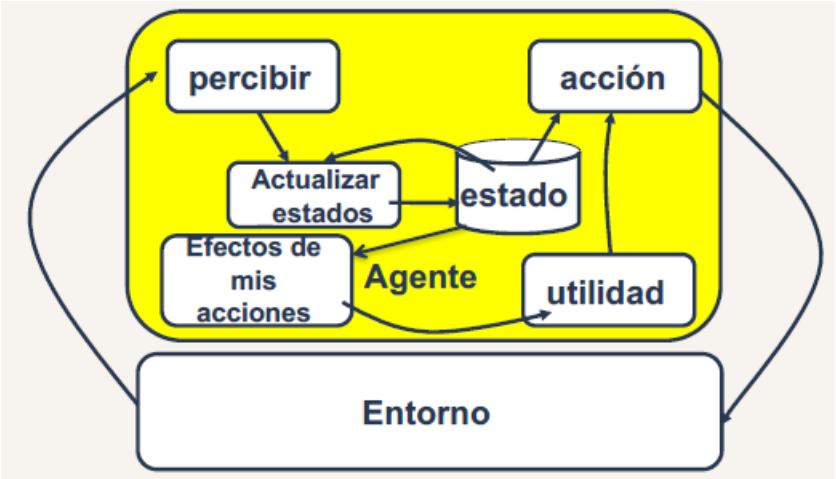
Y se redefine la función acción:

Acción: I x G -> Ac

La acción seleccionada por el agente dependerá del estado interno y de los objetivos que el agente quiera alcanzar.



Un **agente racional** emprende aquella acción que maximice el valor esperado de la medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones. Para diseñar un agente racional, debemos especificar el **entorno de trabajo**.



Una utilidad es un valor numérico que representa cuan bueno es un estado, cuando mayor es la utilidad, mejor es el estado. La tarea del agente es alcanzar estados que maximicen la utilidad. Una especificación de tarea es una función:

$$u: E \rightarrow \mathbb{R}$$

que asocia un número real a cada estado del entorno.

Al asignar utilidades a estados locales, es difícil precisar una visión a largo plazo. Una solución es no asignar una utilidad a estados individuales sino a las ejecuciones del agente:

$$u: R \rightarrow \mathbb{R}$$

Esta aproximación tiene inherentemente una visión a largo plazo. Otras variaciones implican incorporar las probabilidades de los diferentes estados emergentes.

La **utilidad esperada** es denotada por P(r|Ag,Env), es decir, la probabilidad de que ocurra la ejecución r cuando el agente Ag está situado en el entorno Env. Observar que se tiene que cumplir que la utilidad esperada del agente Ag en el entorno Env es:

$$\sum_{r \in R(Ag,Env)} P(r|Ag,Env) = 1$$

$$EU(Ag,Env) = \sum_{r \in R(Ag,Env)} U(r) * P(r|Ag,Env)$$

El **agente óptimo** Ag_{opt} en un entorno Env es aquel que maximiza la utilidad esperada:

$$Ag_{opt} = \underset{Ag \in AG}{\operatorname{argmax}}\, EU(Ag,Env)$$

En promedio podemos esperar que el agente óptimo lo haga mejor.

Denotamos por AG_m el conjunto de los agentes que pueden ser implementados en una máquina m:

$$AG_m = \{Ag | Ag \in AG \; y \; Ag \; puede \; ser \; implementado \; en \; m\}$$

El conjunto de agentes óptimos limitados, Ag_{bopt}, con respecto a m es entonces:

$$Ag_{bopt} = arg \max_{Ag \in AG_m} EU(Ag,Env)$$

Un caso especial de asignación de utilidades a historias es asignar 0 (falso) o 1 (verdadero) a una ejecución. Si se asigna un 1 a una ejecución, entonces el agente tiene éxito en esa ejecución, en cualquier otro caso falla. Llamamos a esto especificaciones predicado de tareas. Denotaremos las especificaciones de predicado de tarea mediante ψ:

$$\psi: \mathcal{R} \rightarrow \{0,1\}$$

Un **entorno de tarea** es un par <Env, ψ>. Denotaremos por$R_\psi(Ag,Env)$ el conjunto de todas las ejecuciones de un agente Ag en un entorno Env que satisface ψ:

$$R_\psi(Ag,Env) = \{r | r \in R(Ag,Env) \; y \; \psi(r) = 1\}$$

Entonces decimos que una agente Ag tiene éxito en un entorno de tarea si:

$$R_\psi(Ag,Env) = R(Ag,Env)$$

Sea P(r|Ag, Env) la probabilidad de que la ejecución r ocurra si el agente Ag está situado en el entorno Env. Entonces la probabilidad $P(\psi|Ag,Env)$ de que ψ sea satisfecho por el agente Ag en el entorno Env será:

$$P(\psi|Ag,Env) = \sum_{r \in R_\psi(Ag,Env)} P(r|Ag,Env)$$

Los dos tipos de tareas más comunes son:

- **Tareas de logro**: conseguir el estado de sucesos ψ. Es especificada por un conjunto G de estados buenos o objetivos. El agente tiene éxito si se garantiza que alcanza al menos uno de estos estados.
- **Tareas de mantenimiento**: mantener el estado de sucesos ψ. Es especificada por un conjunto B de estados malos. El agente tiene éxito en un entorno determinado si consigue evitar todos los estados de B.

La **síntesis de agentes** es una programación automática. El objetivo es tener un programa que tome un entorno de tarea y, a partir de este entorno de tarea, automáticamente genere un agente que tenga éxito en este entorno:

$$syn: \tau\varepsilon \rightarrow (AG \cup \{null\})$$

El algoritmo de síntesis es:

- **Sólido** si cada vez que genera un agente este tiene éxito en el entorno de tarea que se le ha pasado como entrada:

$$syn(< Env, \psi >) = Ag \Rightarrow \mathcal{R}(Ag, Env) = \mathcal{R}_{\psi}(Ag, Env)$$

- **Completo** si está garantizado que genera un agente, siempre que exista un agente que tendrá éxito en el entorno de tarea que se le ha pasado como entrada.

$$\exists Ag \in AG / \mathcal{R}(Ag, Env) = R_{\psi}(Ag, Env) \Rightarrow syn(< Env, \psi >) \neq null$$

TEMA 3.- RAZONAMIENTO PRÁCTICO

1.-ARQUITECTURA DE AGENTE

Un agente es un sistema informático capaz de acción autónoma flexible. Existen tres tipos de arquitectura de agente:

- **Simbólico/lógico**.
- **Reactivo**.
- **Híbrido**.

2.-AGENTES DE RAZONAMIENTO DEDUCTIVO

Los **agentes de razonamiento deductivo** son una aproximación clásica para construir agentes, lo que implica verlos como un tipo particular de sistema basado en el conocimiento, y usar las metodologías de ese tipo de sistemas asociadas para darles soporte. Se define un agente deliberativo o arquitectura de agente deliberativa como aquella que:

- Contiene un modelo del mundo simbólico explícitamente representado: BD de **creencias** especificadas usando fórmulas de lógica de predicados de primer orden.
- Toma de decisiones vía razonamiento simbólico: conjunto de **reglas de deducción**.

Hay dos **problemas** clave para construir un agente así:

1. **El problema de transducción**: traducir el mundo real en una descripción simbólica adecuada (**percibir la información relevante**), a tiempo para que dicha descripción sea útil.
2. **El problema de representación/razonamiento**: cómo representar simbólicamente la información sobre entidades y procesos complejos del mundo del mundo real, y cómo obtener agentes para razonar con esta información a tiempo para que los resultados sean útiles.

Ninguno de estos problemas está completamente resuelto. El problema es la complejidad de la manipulación de símbolos, ya que la mayoría de los algoritmos basados en búsqueda son altamente intratables.

La toma de decisiones usando lógica de primer orden es indecible, incluso usando lógica proposicional. Las soluciones típicas son:

- Simplificar la lógica.
- Usar representaciones simbólicas, no lógicas.
- Pasar el énfasis del razonamiento de la ejecución al diseño.

Otros problemas son que la aproximación lógica presentada implica añadir y borrar cosas de una BD (no monotonía). Eso no es lógica de primer orden (lógica modal). Los primeros intentos de crear un agente planificador utilizaron deducción lógica de primer orden para resolver el problema.

La **Programación Orientada a Agentes** es un nuevo paradigma de la programación, basado en una visión social de la computación. Se trata de programar agentes en términos de nociones

intencionales (creencia, compromiso e intención). La motivación es usar actitudes intencionales como mecanismo de abstracción para representar propiedades de sistemas complejos.

Según Shoham, un sistema de AOP completo tendrá tres componentes:

- Una **lógica** para especificar agentes y describir sus estados mentales.
- Un **lenguaje de programación** interpretado para programar agentes.
- Un proceso de **agentificación**, para convertir aplicaciones en agentes

La relación entre lógica y lenguaje de programación es la semántica.

AGENT0 se considera el primer lenguaje de AOP (saltando la lógica).

3.-AGENTES REACTIVOS

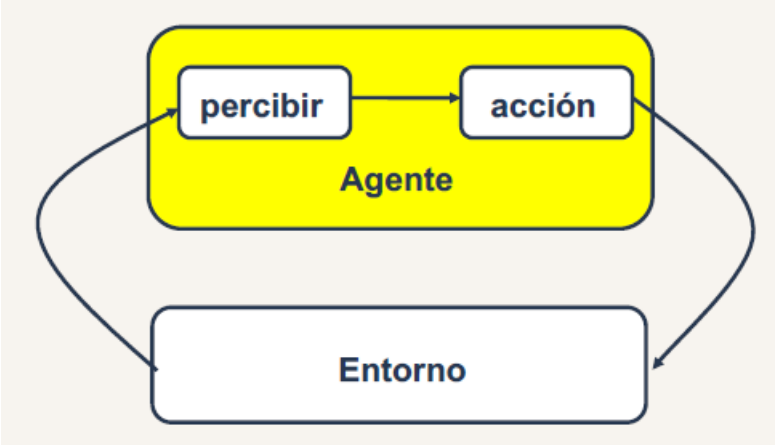
El **comportamiento inteligente** es producto de **interacción** con el entorno y **emerge de comportamientos simples**.

La **arquitectura de subsunción de Brooks** se basa en dos ideas:

- **Estar situado y materializado**: la inteligencia real está situada en el mundo, con un cuerpo, no como sistemas incorpóreos tales como demostradores de teoremas o sistemas expertos.
- **Inteligencia y emergencia**. El comportamiento inteligente surge como resultado de la interacción de un agente con su entorno. La inteligencia está en el ojo del observador, no es una propiedad aislada, innata.

A su vez se sustenta en dos **tesis** clave:

- **Inteligencia sin representación**: comportamiento inteligente que puede ser logrado sin representaciones explícitas del tipo que la IA simbólica propone.
- **Inteligencia sin razonamiento**: comportamiento inteligente que puede ser logrado sin razonamiento abstracto explícito del tipo que la IA simbólica propone.



Un **comportamiento** es un par **(c,a)** donde **c** pertenece a P y es un conjunto de percepciones llamadas la condición, y **a** pertenece a A y es una acción. El conjunto de todos los comportamientos es:

$$Beh = \{(c, a) | c \subseteq P \; y \; a \in A\}$$

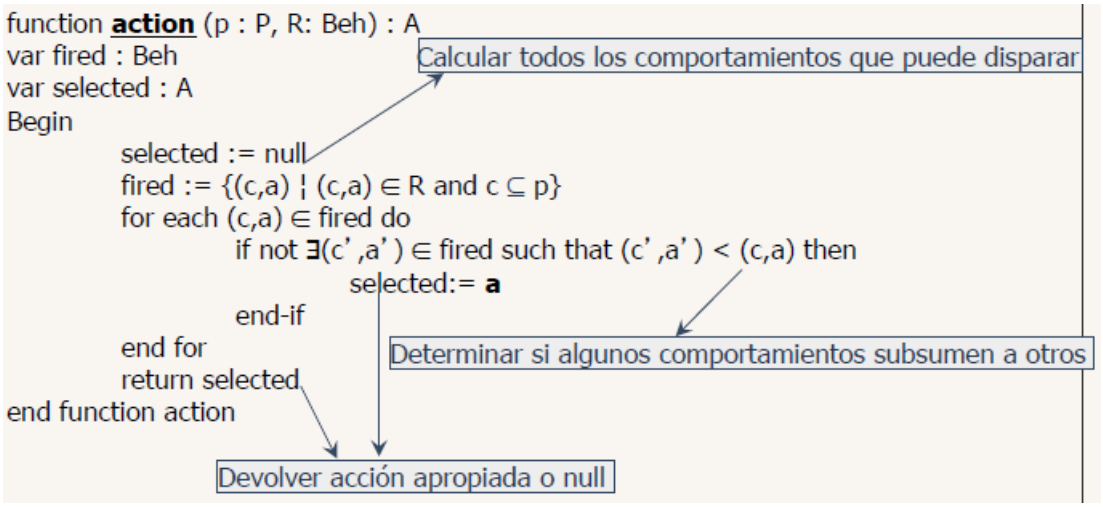
Un comportamiento (c,a) puede **dispararse** cuando el entorno esté en el estado $s \in S$ si $c \subseteq percibir(s)$. En un estado del entorno puede dispararse más de un comportamiento de Beh:

$$disparados = \{(c, a) | (c, a) \in Beh \; y \; c \subseteq percibir(s)\}$$

Como sólo se puede ejecutar una acción tenderemos que elegir un comportamiento de disparados. Los comportamientos compiten para controlar el agente. La selección de acciones se hace en base a una jerarquía. Las capas más bajan tienen precedencia sobre las más altas.

Asociamos a un conjunto de reglas de comportamiento de un agente $R \subseteq Beh$ una relación binaria (<) en el conjunto de comportamientos, es una relación de orden total en R.

La función de selección de acción se define como sigue:



4.-AGENTES HÍBRIDOS

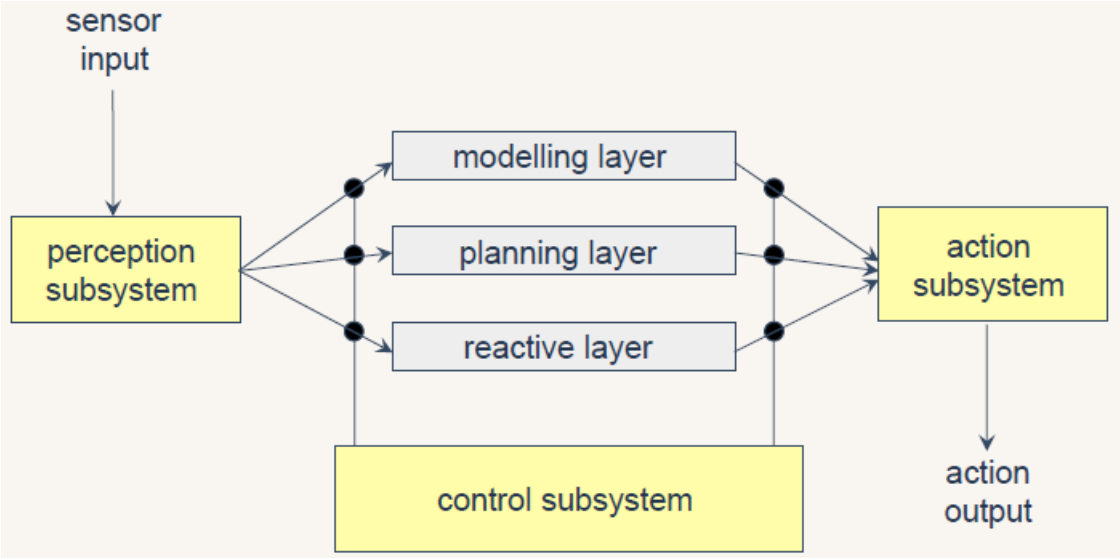
Un **agente híbrido** es un agente construido a partir de dos o más subsistemas:

- Uno **deliberativo**, conteniendo un modelo del mundo simbólico, que desarrolla planes y toma decisiones de la manera propuesta por la IA simbólica.
- Uno **reactivo**, que es capaz de reaccionar a eventos sin razonamiento complejo.

En su mayoría, se le da preferencia al componente reactivo sobre el deliberativo. Esta diseñado como una arquitectura en capas:

- **Capas horizontales**: capas directamente conectadas a la entrada sensorial y la salida de acciones. Son subsistemas de percepción y acción. Tienen tres capas que producen actividad:
  - **Capa reactiva**: respuesta inmediata a cambios en el entorno.
  - **Capa de planificación**: comportamiento pro-activo.
  - **Capa de modelado**: representa las diversas entidades en el mundo. Predice conflictos y genera nuevos objetivos para resolver estos conflictos.

  También posee un **subsistema de control**, un conjunto de reglas de control para decidir qué capa debería tener control sobre el agente.



Sus **ventajas** son la simplicidad conceptual, si necesitamos que un agente presente n tipos diferentes de comportamientos, implementamos n capas diferentes. Sus **desventajas** son la competición entre capas. Para ello se utiliza la **función mediador** que toma la decisión sobre quién tiene control.

- **Capas verticales**: la entrada sensorial y la salida de acciones son tratadas por una capa como mucho. Está basada en tres capas:
    - **Capa basad en el comportamiento**: comportamiento reactivo.
    - **Capa de planificación local**: para planificación diaria.
    - **Planificación co-operativa**: para interacciones sociales.

  Control entre capas:

    - **Activación de abajo a arriba**: si la capa no es competente para tratar con la situación, pasa el control a la de arriba.
    - **Ejecución de arriba a abajo**: cuando una capa más alta hace uso de las facilidades proveídas por una capa más baja para lograr uno de sus objetivos.

  El flujo de control de agentes empieza con una entrada perceptual que llega a la capa más baja. Si la capa reactiva puede tratar con esta entrada, lo hace, sino se hace una activación de abajo a arriba. Si la capa de planificación local puede manejar la situación, lo hace, sino se hace una activación de abajo a arriba.

  Sus **ventajas** son que la complejidad de interacciones entre capas es reducida. Sus **desventajas** son su flexibilidad, es decir, es no tolerante a fallos.

5.-AGENTES DE RAZONAMIENTO PRÁCTICO

El **razonamiento práctico** es el razonamiento dirigido por las **acciones**, el proceso de descubrir que hacer. El razonamiento práctico se distingue del **razonamiento teórico** es dirigido por las **creencias**.

La **arquitectura BDI** para agentes con recursos incluye:

- Análisis de objetivos.
- Sopesar distintas alternativas.
- Interacción entre estas dos formas de razonamiento.

Algunos conceptos básicos son:

- **Beliefs**: información que tiene el agente sobre el mundo.
- **Desires**: estado de los asuntos que el agente desearía alcanzar.
- **Intentions**: deseos (o acciones) que el agente se ha comprometido a lograr.

Esta arquitectura ha sido particularmente convincente por:

- **Componente filosófico**: basado en una teoría de acciones racionales en humanos.
- **Arquitectura software**: ha sido implementado y usado con éxito en un gran número de complejas situaciones.
- **Componente lógico**: el modelo ha sido formalizado rigurosamente en una familia de lógicas BDI.

El **razonamiento práctico humano** es la suma de la deliberación con el razonamiento guiado por los objetivos. La **deliberación** es decidir qué estados se desean alcanzar (objetivos). El resultado de esta deliberación son las **intenciones**. El **razonamiento guiado por objetivos** es decidir cómo lograr esos estados. El resultado de este razonamiento es un **plan** o **secuencia de acciones**.

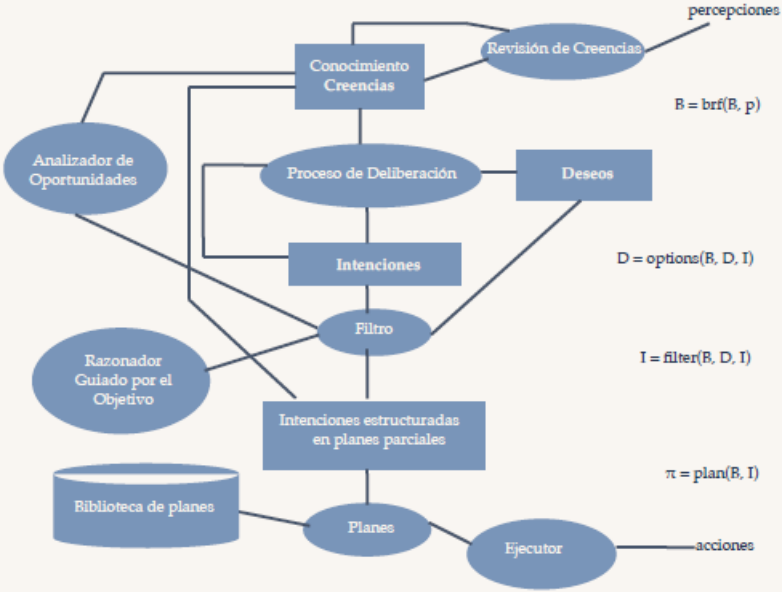
Los agentes necesitan determinar formas de lograr las intenciones. A su vez, éstas son como un filtro para adoptar otras intenciones que no entren en conflicto. Los agentes siguen el éxito de sus intenciones, y están inclinados a volver a intentarlas si fallan. También creen que sus intenciones son posibles y que las conseguirán bajo ciertas circunstancias, pero no creen que no lograrán sus intenciones. Los agentes no tienen que planificar todos los efectos colaterales de sus intenciones. Este problema se conoce como **efecto colateral** o problema del **todo en el mismo paquete**. Las **intenciones** son mucho más fuertes que los meros **deseos**.

La idea detrás de un agente planificador es dotar a éste de:

- Representación de objetivo / intención a lograr.
- Representación de acciones que puede realizar.
- Representación del entorno.
- Hacerle generar un **plan** para lograr el objetivo.

Esencialmente esto es **programación automática**.

5.1.-ARQUITECTURAS BDI



El bucle de control del agente se basa en:

1. Observar el mundo.
2. Actualizar el modelo interno del mundo.
3. Deliberar sobre qué intenciones lograr.
4. Crear un plan y ejecutarlo.

El problema de los procesos de deliberación y razonamiento guiado por objetivos es que no son instantáneos, ya que tienen un **coste temporal**. Si el agente comienza a deliberar en t0, comienza el razonamiento guiado por objetivos en t1, y comienza a ejecutar el plan en t2. El tiempo para deliberar es t1–t0. El tiempo para el razonamiento guiado por los objetivos es t2–t1.

La deliberación es **óptima** si cuando selecciona alguna intención a lograr, ésta es la mejor opción para el agente. Así en el instante t1, el agente ha elegido una intención a lograr que hubiera sido óptima si se hubiese logrado en t0. Pero, a menos que el tiempo de deliberación sea muy pequeño, el agente corre el riesgo de que la intención seleccionada no sea ya óptima en el momento en el que el agente se fija en ella. Esto es **racionalidad calculada**. La deliberación es sólo la mitad del problema, el agente todavía tiene que determinar cómo lograr la intención.

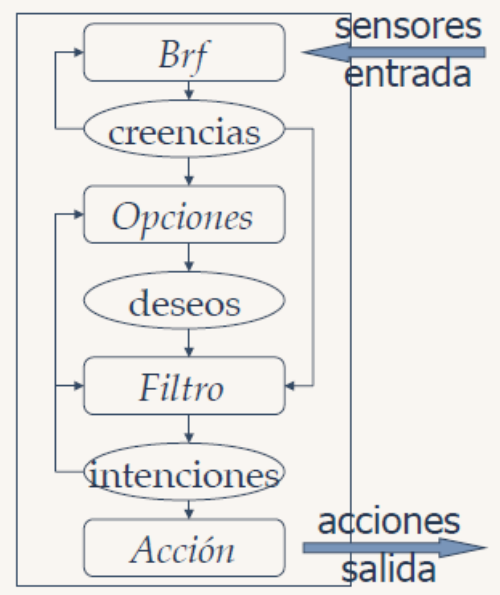
Un agente así tendrá un comportamiento global óptimo en las siguientes circunstancias:

1. Cuando la deliberación y el razonamiento guiado por el objetivo toman muy poco tiempo.
2. Cuando se garantiza que el mundo permanecerá estático mientras el agente está deliberado y realizando su razonamiento basado en objetivos, con lo que las suposiciones sobre la elección de qué intención lograr y qué plan usar para lograr la intención permanecen válidas hasta que el agente ha completado la deliberación y el razonamiento dirigido por los objetivos.
3. Cuando una intención que es óptima si se logra en el instante t0 se garantiza que permanece siendo óptima hasta el instante t2.

Un agente para deliberar comienza intentando entender qué **opciones** tiene disponibles, después **eligen entre ellas**, y **se compromete** con alguna. Las opciones elegidas son entonces, intenciones.

La función **deliberar** puede ser descompuesta en dos componentes funcionalmente distintos:

1. **Generación de opciones**: el agente genera un conjunto de posibles alternativas. Se representa la generación de opciones por una función, **opciones**, que a partir de las creencias e intenciones actuales del agente determina un conjunto de opciones (**desires**).
2. **Filtrado**: el agente elige entre las alternativas que compiten y se compromete a lograrlas. Para seleccionar entre opciones que compiten, un agente usa una función de filtrado.



Las **estrategias de compromiso** más comunes son:

- **Compromiso ciego o fanático**: un agente comprometido ciegamente continuará manteniendo una intención hasta que crea que la ha logrado.
- **Compromiso inquebrantable**: un agente inquebrantable continuará manteniendo una intención hasta que crea que o bien ha logrado la intención, o bien ya no es posible lograrla.
- **Compromiso sin prejuicios**: un agente sin prejuicios mantendrá una intención mientras la crea posible.

Un agente se compromete tanto a los **fines** como a los **medios**, por ende, se debería **replanificar** siempre que un plan vaya mal (compromiso ciego).

```
Agent Control Loop Version 4
1.
2.   B := B0;
3.   I := I0;
4.   while true do
5.        get next percept ρ;
6.        B := brf(B, ρ);
7.        D := options(B, I);
8.        I := filter(B, D, I);
9.        π := plan(B, I);
10.       while not empty(π) do
11.            α := hd(π);
12.            execute(α);
13.            π := tail(π);
14.            get next percept ρ;
15.            B := brf(B, ρ);
16.            if not sound(π, I, B) then
17.                 π := plan(B, I)
18.            end-if
19.       end-while
20.  end-while
```

Esta implementación **nunca para a considerar si sus intenciones son apropiadas o no**, por lo que deberíamos determinar si las intenciones han tenido éxito o si son imposibles (compromiso inquebrantable).

```
Agent Control Loop Version 5
2.   B := B0;
3.   I := I0;
4.   while true do
5.        get next percept ρ;
6.        B := brf(B, ρ);
7.        D := options(B, I);
8.        I := filter(B, D, I);
9.        π := plan(B, I);
10.       while not (empty(π)
                  or succeeded(I, B)
                  or impossible(I, B)) ) do
11.            α := hd(π);
12.            execute(α);
13.            π := tail(π);
14.            get next percept ρ;
15.            B := brf(B, ρ);
16.            if not sound(π, I, B) then
17.                 π := plan(B, I)
18.            end-if
19.       end-while
20.  end-while
```

Nuestro agente ahora reconsidera sus intenciones cada vez que está en el bucle exterior, cuando:

- Ha **ejecutado completamente un plan** para lograr sus intenciones actuales.
- Cree que **ha logrado sus intenciones actuales**.
- Cree que sus **intenciones ya no son posibles**.

Esto permite a un agente **reconsiderar** sus intenciones, pero debería reconsiderar intenciones después de ejecutar cada acción (compromiso de mente abierta).

```
Agent Control Loop Version 6
1.
2.   B := B_0;
3.   I := I_0;
4.   while true do
5.        get next percept ρ;
6.        B := brf(B, ρ);
7.        D := options(B, I);
8.        I := filter(B, D, I);
9.        π := plan(B, I);
10.       while not (empty(π)
                  or succeeded(I, B)
                  or impossible(I, B)) do
11.           α := hd(π);
12.           execute(α);
13.           π := tail(π);
14.           get next percept ρ;
15.           B := brf(B, ρ);
16.           D := options(B, I);
17.           I := filter(B, D, I);
18.           if not sound(π, I, B) then
19.               π := plan(B, I)
20.           end-if
21.       end-while
22.  end-while
```

La reconsideración de intenciones es muy **costosa**, por lo que surge el siguiente dilema:

- Un agente que no para a reconsiderar sus intenciones muy a menudo continuará intentando lograr sus intenciones incluso después de que esté claro que no pueden ser logradas, o de que ya no haya ninguna razón para lograrlas.
- Un agente que **constantemente** reconsidera sus intenciones puede dedicarle realmente demasiado poco tiempo para lograrlas, y por lo tanto corre el riesgo de realmente no lograrlas nunca.

La solución es incorporar un componente explícito de **control meta-nivel**, que decida si debe o no reconsiderarlas.

```
Agent Control Loop Version 7
1.
2.   B := B_0;
3.   I := I_0;
4.   while true do
5.        get next percept ρ;
6.        B := brf(B, ρ);
7.        D := options(B, I);
8.        I := filter(B, D, I);
9.        π := plan(B, I);
10.       while not (empty(π)
                  or succeeded(I, B)
                  or impossible(I, B)) do
11.           α := hd(π);
12.           execute(α);
13.           π := tail(π);
14.           get next percept ρ;
15.           B := brf(B, ρ);
16.           if reconsider(I, B) then
17.                D := options(B, I);
18.                I := filter(B, D, I);
19.           end-if
20.           if not sound(π, I, B) then
21.                π := plan(B, I)
22.           end-if
23.       end-while
24.  end-while
```

Las interacciones posibles entre control meta-nivel y deliberación son:

Situation number	Chose to deliberate?	Changed intentions?	Would have changed intentions?	*reconsider*(...) optimal?
1	No	—	No	Yes
2	No	—	Yes	No
3	Yes	No	—	No
4	Yes	Yes	—	Yes

El coste de **reconsiderar** es mucho menor que el coste del proceso de deliberación mismo. Existen dos tipos diferentes de **estrategias de reconsideración**:

- Agentes **atrevidos**: nunca paran a reconsiderar intenciones.
- Agentes **cautos**: después de cada acción, se paran a reconsiderar.

El **dinamismo** en el entorno es representado por el ratio **de cambio en el mundo**, **ϒ**. Si **ϒ** es baja, los agentes atrevidos funcionan bien comparados con los cautos. Si **ϒ** es alta, los agentes cautos tienden a superar a los agentes atrevidos.

5.2.-JASON

El lenguaje que interpreta es una extensión de AgentSpeak. Los elementos fundamentales del lenguaje son:

- Creencias (beliefs).
- Objetivos (goals).
- Planes (plans, intentions).

Cada agente tiene una base de creencias, que son a su vez, una colección de literales representados como predicados. Las anotaciones son detalles asociados a una creencia que aportan elegancia al lenguaje y facilitan el manejo de la base de creencias.

Existen anotaciones que tienen un significado especial para el intérprete. En particular la anotación **source**. Hay tres tipos de fuentes de información para los agentes:

- **Información perceptual**: aquella que percibe del entorno, **source(percept)**.
- **Comunicación**: aquella que proviene de otro agente del sistema, **source(id agente)**.
- **Notas mentales**: creencias que provienen del propio agente, **source(self)**.

StrongNegation: se denota con "~". Expresa que el agente cree explícitamente que algo es falso.

Reglas: permiten inferir nueva información a partir de conocimiento que se tiene.

Existen dos **tipos de objetivos** en JASON:

- **Achievement goals(operador !)**: expresan un estado del mundo que el agente desea conseguir.
- **Test goals(operador ?)**: usados normalmente para recuperar información de la base de creencias.

Un **plan** tiene **tres partes**:

- **Triggering event**: cambios en las creencias u objetivos del agente.
- **Context**: determinan si un plan es aplicable.
- **Body**: sucesión de acciones que comporta el plan. Se separan por ";". Estas instrucciones pueden ser:
    - **Actions**: acciones externas que se denotan por un predicado. Proporcionan un "feedback".
    - **Achievement goals**: subobjetivos que deben ser alcanzados para que el plan continúe su ejecución. Si en lugar de emplear "!" se emplea "!!", el plan no suspenderá su ejecución.
    - **Test goals**: para recuperar información de la BB o verificar si el agente cree algo.
    - **Mental notes**: sirven para añadir, modificar o eliminar nuevas creencias.
    - **Internal actions**: son acciones que no modifican el entorno. Se diferencian de acciones del entorno por el carácter.
    - **Expressions**.
    - **Plan labels**.

Un plan puede fallar por tres causas principales:

- Falta de planes relevantes o aplicables para un achievement goal.
- Fallo de un test goal.
- Fallo de una acción.

Cuando un plan falla se genera un evento -!g(goal deletion event) si se generó por la adición de un achievement goal o test goal. El plan que se dispare por el fallo se añade a la pila de intenciones del plan que ha fallado.

Para cada creencia se anota su origen (id, self, perception). Cada agente posee una mailbox M y una función de selección de mensajes de M.

TEMA 4.- CAPACIDAD SOCIAL

1.-COMUNICACIÓN

La **comunicación** es el acto o proceso por el que una o varias personas (emisores) transmiten un mensaje con información a otra u otras (receptoras).

El **código** es el conjunto de signos y reglas con que el emisor ha formado su mensaje.

El **canal** es el medio físico o el soporte tecnológico que hace que el mensaje llegue del emisor al receptor.

El **mensaje** es la información que el emisor le transmite al receptor.

El **contexto** es el lugar y el momento en el que se produce la comunicación.

Si dos agentes tienen que comunicarse en un dominio o contexto determinado es necesario que compartan la terminología que utilizan para describir el dominio. Una **ontología** es una especificación de un conjunto de términos que tiene por objeto proporcionar una base común de comprensión sobre algún dominio.

El propósito de **Knowledge Sharing Effort** es el desarrollo de técnicas, metodologías y herramientas software para la compartición y reutilización del conocimiento entre sistemas a lo largo de las etapas del ciclo de vida del software. Los componentes fundamentales para la interacción eficaz de agentes de software son:

1. Un lenguaje común.
2. Una comprensión común del conocimiento intercambiado.
3. Una habilidad para intercambiar todo lo relativo a (1) y (2).

Compartir conocimiento entre agentes requiere la capacidad de comunicarse, la capacidad de comunicarse requiere un lenguaje de comunicación común:

- **Sintaxis**: KIF (Knowledge Interchange Format). Lenguaje para el intercambio de conocimiento.
- **Semántica**: Ontolingua. Lenguaje para la definición de Ontologías.
- **Pragmática**: KQML (Knowledge Query and Manipulation Language). Lenguaje para la comunicación entre agentes. Proporciona interoperabilidad de alto nivel entre agentes en un entorno distribuido.

2.-LENGUAJE DE COMUNICACIÓN

La comunicación es la base para las interacciones y la organización social de los agentes. Hay interacciones cuando la dinámica de un agente está perturbada por las influencias de otros. Éstas son el motor de los sistemas multi-agente. Existen distintas **formas de interaccionar**:

- Acciones sobre el entorno.
- Pizarra compartida.
- Paso de mensajes.

Las capacidades de comunicación son lo bloques básicos con los que construir mecanismos de coordinación, cooperación y negociación entre agentes.

En el **sistema de pizarra**, la pizarra es una zona de trabajo común que permite a los agentes compartir todo tipo de información. Un sistema multiagente puede tener varias pizarras con distintos agentes registrados en cada una. No hay comunicación directa entre agentes. Los sistemas más avanzados incorporan nuevos conceptos:

- **Moderador**: agente especializado con conocimiento de control y de evaluación que publica en la pizarra los subproblemas a resolver y decide a qué agentes se asignan de entre aquellos que se han ofrecido a resolverlos.
- **Despachador**: agente que avisa a los agentes afectados por algún cambio producido en la pizarra.

Los sistemas de pizarra constituyen un método muy flexible de comunicación para la resolución distribuida de problemas. Son independientes de la estrategia de cooperación que se vaya a utilizar y no afectan a la arquitectura de los agentes individuales. Sin embargo, la estructura central de la pizarra representa cada vez más un inconveniente ya que todos los agentes distribuidos por una red se ven obligados a acceder al dispositivo central donde se encuentra la pizarra.

El **sistema de paso** de mensajes es un método de comunicación muy flexible que se establece directamente entre dos agentes. Los mensajes pueden ser de muy distinto tipo, permitiendo así la implementación de muy diversas estrategias de interacción entre agentes. Es necesario establecer **protocolos de comunicación** que especifiquen el proceso de comunicación, el **formato de los mensajes** y el **lenguaje de comunicación**. Todos los agentes deben conocer la semántica del lenguaje de comunicación. Realmente el significado de un mensaje depende del estado mental de los agentes involucrados y de la relación entre ambos. Un mensaje producirá cambios inicialmente en el estado mental del receptor y eventualmente en el resto del entorno.

La función **denotativa** del lenguaje es determinar la verdad o falsedad de una frase. Los actos del habla hacen referencia a la función **conativa**, ya que designan las acciones intencionales en el curso de una conversación.

Quien habla no declara solamente sentencias ciertas o falses, sino que realiza actos de habla: peticiones, sugerencias, promesas, amenazas, etc. Cada declaración es un acto de habla. Identificar estos actos es imprescindible para una correcta comunicación. Los tipos de actos de habla son:

- **Actos asertivos**: dan información sobre el mundo.
- **Actos directivos**: para solicitar algo al destinatario.
- **Actos de promesa**: comprometen al locutor a realizar ciertas acciones en el futuro.
- **Actos expresivos**: dan indicaciones del estado mental del locutor.
- **Actos declarativos**: el mero hecho de la declaración es la realización de un acto.

Los componentes del acto de habla son:

- **Locución**: modo de producción de frases utilizando una gramática y un léxico.
- **Ilocución**: acto realizado por el locutor sobre el destinatario mediante la declaración (utterance).
    - **Fuerza ilocutoria (F)**: información, pregunta, orden, petición, promesa, oferta.
    - **Contenido proposicional (P)**: objeto de la fuerza ilocutoria.
- **Perlocución**: efectos que pueden tener los actos ilocutorios en el estado del destinatario y en sus acciones, creencias y juicios.

Todo acto del habla consiste en una fuerza F aplicada a una proposición P.

Una declaración no es verdadera o falsa, sino tiene éxito o fracasa. Un acto de habla puede fallar:

- En su enunciación, porque no llegue el mensaje o llegue corrompido o el destinatario no lo entiende.
- En su interpretación, por el destinatario.
- En su consecución final.

2.1.-KQML (KNOWLEDGE QUERY MANAGEMENT LANGUAGE)

En la comunicación intervienen los siguientes elementos:

- **El protocolo de interacción**. Estrategia de alto nivel seguida por el agente software para controlar la interacción con otros agentes. Desde esquemas de negociación hasta esquemas más simples.
- **El lenguaje de comunicación**. Es el medio a través del que se intercambian los actos de la comunicación. Indica si el contenido de la comunicación es una información, una respuesta o algún tipo de consulta.
- **El protocolo de transporte**. Mecanismo de transporte utilizado en la comunicación.

KQML asume un modelo de agentes, entidades de alto nivel con capacidades cognitivas, que tienen una descripción de nivel intencional. Los agentes residen en el **nivel de conocimiento**. Los mensajes KQML comunican una actitud sobre el contenido que llevan. Las primitivas del lenguaje se llaman performativas y cada mensaje KQML representa un acto de habla. El lenguaje KQML está divido en tres capas:

- **La capa de contenido**. Relacionada con el contenido del mensaje y puede expresarse en cualquier lenguaje. KQML sólo tiene interés en identificar el inicio y el final del contenido.
- **La capa de mensaje**. Es el núcleo del lenguaje KQML. Determina los tipos de interacción que puede realizar un agente e identifica el protocolo, la ontología y el acto del habla.
- **La capa de comunicación**. Identifica las características del mensaje que describen los parámetros de bajo nivel de la comunicación.

Los mensajes KQML representan un acto de habla o performativa y constan de una lista de pares atributo-valor (el primer elemento es el identificador del acto del habla, el resto son pares atribut-valor).

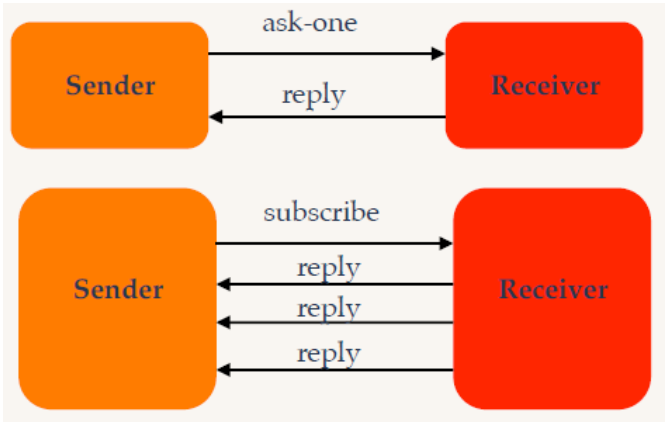
Palabras reservadas:

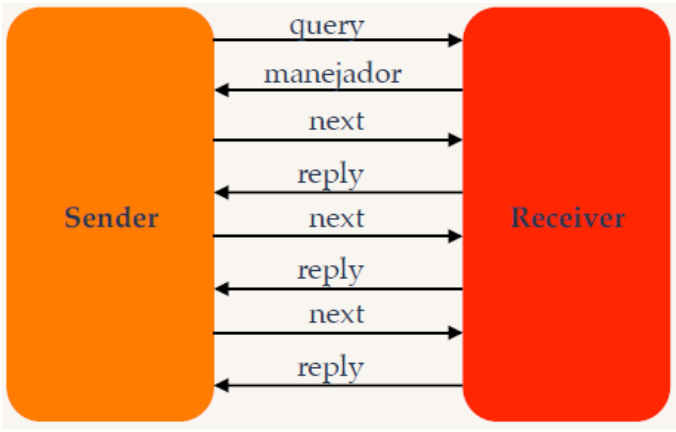
Atributo	Significado
:content	La información
:sender	Emisor del mensaje
:receiver	Receptor del mensaje
:language	El nombre del lenguaje de representación empleado en el atributo :content
:ontology	El nombre de la ontología utilizada en el atributo : content
:reply-with	Etiqueta para la respuesta (si es que el emisor la espera)
:in-reply-to	La etiqueta esperada en la respuesta

Performativas:

Nombre	Significado
tell	S comunica una información que se encuentra en su BC
evaluate	S quiere que R evalúe la expresión
reply	S comunica a R una respuesta esperada
ask-if	S quiere saber si la expresión se encuentra en la BC de R
ask-about	S quiere conocer todo el conocimiento de R relacionado con la expresión
ask-one	S quiere que R conteste a una pregunta
stream-about	Versión en respuesta múltiple de ask-about
achieve	S quiere que R convierta en verdadera la expresión en su BC
standby	S quiere que R esté preparado para responder a una *performative*
ready	S está preparado para responder a R
next	S quiere que R le devuelva la siguiente respuesta (de una consulta múltiple)
advertise	S está preparado para responder a una determinada *performative*
register	S puede responder a *performatives* de un determinado agente
broadcast	S quiere que R envíe la *performative* a todas los agentes conectados
recommend-one	S quiere el nombre de un agente que pueda contestar a una *performative*
recruit-one	S quiere que R le proporcione el nombre de otro agente que conteste a la *performative*

Protocolos:





Los mensajes recibidos por un agente JASON tienen la siguiente estructura:

<emisor, fuerza_ilocutoria, contenido>

La forma general de la acción interna predefinida para la comunicación es:

.send(receptor, fuerza_ilocutoria, contenido_proposicional)

También se puede hacer:

.broadcast(fuerza_ilocutoria, contenido_proposicional)

El agente s ejecuta .send(r, performativa, contenido):

- **tell**: s pretende que r crea (lo que s cree) que el literal del contenido es verdadero.
- **untell**: s pretende que r no crea (lo que s cree) que el literal del contenido es verdadero.
- **achieve**: s pide a r que invente alcanzar un estado en el que el literal del contenido del mensaje sea verdadero (s delega un objetivo en r).
- **unachieve**: s pide a r que abandone el objetivo de alcanzar un estado donde el contenido del mensaje sea cierto.
- **askone**: s quiere saber si el contenido del mensaje es verdadero para r.
- **askAll**: s quiere todas las respuestas de r a una pregunta.
- **tellHow**: s informa a r de un plan (que s conoce).
- **untellHow**: s pide a r que ignore un determinado plan.
- **askHow**: s desea obtener todos los planes de r que son relevantes para el evento de disparo especificado en el contenido del mensaje.

TEMA 5.- SISTEMAS MULTIAGENTE

1.-INTRODUCCIÓN

Un **sistema multiagente** está constituido por un conjunto de agentes (dos o más) que interactúan los unos con los otros. En el caso más general, los agentes actúan en representación de usuarios que tienen diferente objetivos y motivaciones. Para interactuar con éxito, requieren capacidades para cooperar, coordinarse y negociar con cada uno de los otros.

Los sistemas multiagente funcionan de la siguiente manera:

1. El agente monitoriza la actividad del usuario:
    a. Lee/escucha la conversación del usuario.
    b. Reconoce patrones en la conversación.
    c. Deduce información y objetivos en función de la experiencia pasada.
2. El agente persigue satisfacer sus objetivos:
    a. Toma decisiones.
    b. Puede descomponer objetivos en subobjetivos.
    c. Ejecuta tareas.
3. Para cumplir sus objetivos necesita la colaboración con otros agentes:
    a. Negociación.
    b. Delegación.
    c. Coordinación.
4. Los agentes necesitan servicios de localización de agentes:
    a. Páginas blancas.
    b. Páginas amarillas.
5. Comunicación con el usuario:
    a. Interfaces avanzadas.
    b. Información implícita a partir de experiencias pasadas o preferencias del usuario.
    c. Gestión de diálogos.

La tecnología de SMA nos permite modelar sistemas reales complejos y con características claramente distribuidas. Se puede ver como una organización computacional consistente de varios “roles” interactuando.

Se debe identificar los diferentes subsistemas que forman parte del sistema global, y las posibles interacciones y dependencias entre ellos. Hay que tener en cuenta el punto de vista interno (un agente) y el punto de vista externo (varios agentes).

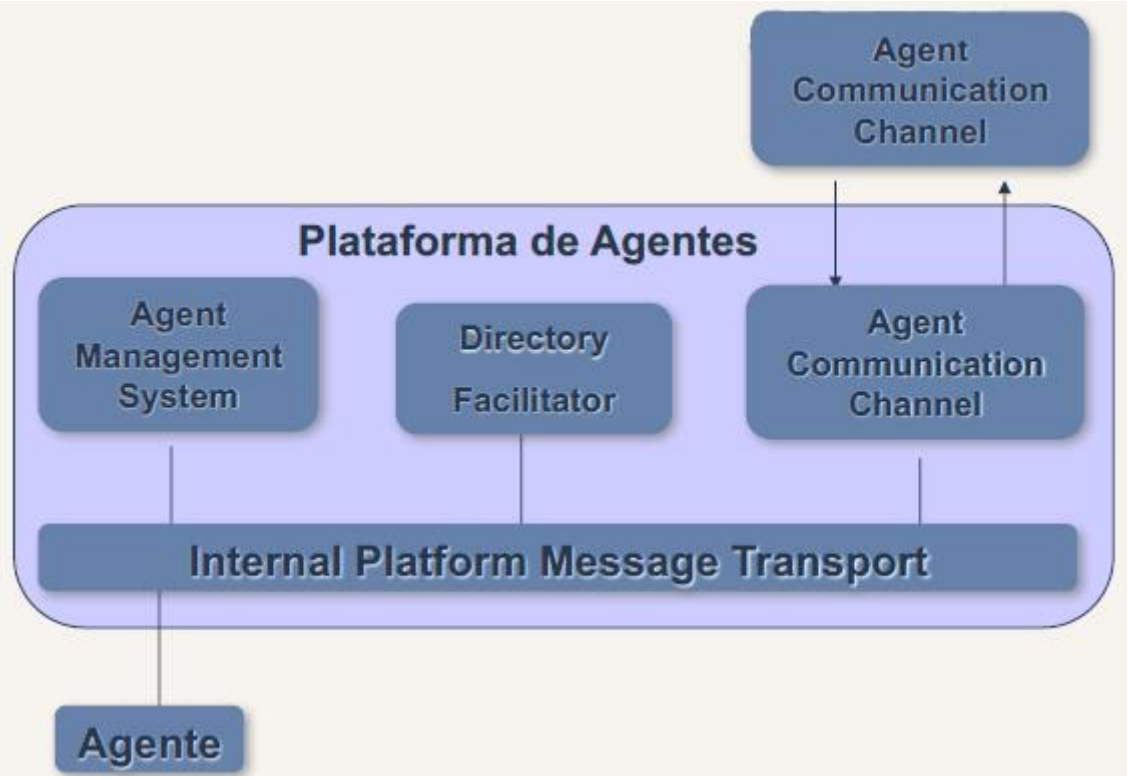
Las **ventajas** que presentan los SMA son:

- **Modularidad**: se reduce la complejidad al trabajar con unidades más pequeñas, permitiendo una programación más estructurada.
- **Eficiencia**: la programación distribuida permite repartir las tareas entre los agentes, es decir, paralelismo.
- **Fiabilidad**: que un elemento del sistema deje de funcionar no tiene que significar que el resto también lo tenga que hacer. Se puede conseguir más seguridad replicando servicios críticos y tener así redundancia.
- **Flexibilidad**: se pueden añadir y eliminar agentes dinámicamente.

2.-MODELO CONCEPTUAL DE PLATAFORMA FIPA

FIPA (Foundation for Intelligent Physical Agents) es una organización de estándares para agentes y sistemas multiagente.

Su modelo conceptual sería el siguiente:



La **plataforma de agentes** es la infraestructura en la cual los agentes pueden ser desarrollados y usados. Hardware y software.

El **agent management system** es el elemento de gestión principal. Maneja el estado de la plataforma y el estado de los agentes de la misma. Ofrece los siguientes servicios:

- Creación, destrucción y control del cambio de estado de los agentes.
- Supervisar los permisos para que nuevos agentes se registren.
- Control de la movilidad de los agentes.
- Gestión de los recursos compartidos.
- Gestión del canal de comunicación.
- Servicio de nombres (ANS) o páginas blancas (nombre - dirección).

El **directory facilitator** es el servicio de páginas amarillas. Los agentes se registran indicando los servicios que ofrecen.

El **agent communication channel** ofrece envío de mensajes entre agentes de la misma o distinta plataforma de forma asíncrona.

El **internal platform message transport** es la infraestructura de comunicaciones que permite que dos agentes se comuniquen.

3.-PLATAFORMA DE AGENTE

Las **plataformas de agentes** sirven para ejecutar sistemas multiagente.

El middleware es un software de **conectividad** que ofrece un **conjunto de servicios** que hacen posible el funcionamiento de aplicaciones distribuidas sobre **plataformas heterogéneas**. Funciona como una capa de **abstracción de software distribuida**, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El middleware nos **abstrae de la complejidad y heterogeneidad de las redes de comunicaciones** subyacentes, así como de los sistemas operativos y lenguajes de programación, **proporcionando una API para facilitar la programación** y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipos de servicios de middleware.

La **plataforma** proporciona un **alto nivel de abstracción** para el desarrollo de determinadas aplicaciones. Es un conjunto de **herramientas** que facilitan el desarrollo. También es una **API** adaptada al dominio de las aplicaciones.

Las dos plataformas de agentes más usadas son:

- **JADE** (Java Agent DEvelopment framework): marco de trabajo para el desarrollo de SMA y aplicaciones acordes con el estándar FIPA en Java.
- **Magentix2**: ofrece soporte al desarrollo de SMA abiertos. Sus características son:
    - Agentes programados en cualquier lenguaje de programación.
    - Soporte para múltiples arquitecturas de agentes.
    - Interacción entre agentes transparente.
    - Soporte para protocolos de interacción dinámicos.
    - Agentes BDI.
    - Organizaciones de agentes.

  Un agente Magentix2 tiene tres métodos principales: **init**, **execute** y **finalize**. Éstos se ejecutan en el siguiente orden:

  1. Init: el código se ejecuta al principio de la ejecución del agente.
  2. Execute: es el método que contiene el código principal del agente.
  3. Finalize: estemétodo se ejecuta justo antes de que el agente finalice y sea destruido.

  Solo es obligatorio incluir código en el método execute.

TEMA 6.- NEGOCIACIÓN

1.-INTRODUCCIÓN

La **negociación** es un proceso por el cual una **decisión conjunta** es tomada por dos o más partes. Las partes primero verbalizan demandas contradictorias y se **mueven** hacia un acuerdo mediante un proceso de **concesión**, haciendo o buscando nuevas alternativas.

La negociación también puede definirse como una forma de **interacción** en la que un grupo de agentes o personas con intereses en **conflicto** y un deseo de **cooperar**, intentan alcanzar un acuerdo mutuamente satisfactorio en la división de una serie de recursos limitados.

Características de la negociación:

- Situación de conflicto, diferentes preferencias de los participantes.
- Interacción entre dos o más partes.
- Proceso de búsqueda conjunta.
- Inherente sentimiento de cooperación y competición.
- Existe prácticamente en cualquier dominio.

La **negociación automática** surge como rama de la inteligencia artificial. Es demasiado costoso buscar la solución óptima, por lo que se buscan soluciones cercanas al óptimo a menor coste. Se suelen evitar las siguientes asunciones:

- Conocimiento perfecto.
- Recursos computacionales ilimitados.
- Proceso no acotado en el tiempo.

Los primeros trabajos de la negociación automática datan de la época de los 80.

Existen diferentes **tipologías de las negociaciones**:

- En base al número de participantes en el proceso:
    - 1 vs 1 (regateo, comprador-vendedor).
    - 1 vs n / n vs 1 (subasta, equipos de negociación).
    - n vs n (group decisión making, ollaborative design, equipos de negociación).
- Según el número de atributos:
    - Monoatributo (típicamente el precio).
    - Multiatributo.
- Según el tipo de relación a establecer:
    - Corto plazo.
    - Largo plazo (alianza estratégica).
    - Ninguna (resolución de disputas).
- De acuerdo a la intervención de terceras partes:
    - Negociaciones no mediadas.
    - Negociaciones mediadas.
- Según la naturaleza social del conflicto:
    - Interpersonal.
    - Intragrupo.
    - Intergrupo.

- En base a los tipos de solución:
    - **Distributiva**: hacer ganar valor a una de las partes supone pérdidas en otra de las partes.
    - **Integrativa**: existe potencial para encontrar una situación donde todas las partes salen ganando.
    - **Mixta**: valores tanto distributivos como integrativos.
- Cardinalidad de los atributos:
    - Binarios.
    - Discretos (ordenables, no ordenables).
    - Continuos.

Las **tareas básicas** que debe llevar a cabo un **agente negociador** son:

- Preliminares:
    - Conflicto social: identificar una situación de conflicto y su naturaleza.
    - Participantes en la negociación:
        - Identificar posibles oponentes.
        - Identificar potenciales compañeros.
        - Identificar competidores.
- Modelo de pre-negociación:
    - Estructurar la información personal:
        - Definir atributos a negociar.
        - Caracterizar las preferencias sobre los atributos.
        - Definir límites y objetivos.
    - Análisis de los oponentes:
        - Obtener o inferir información sobre los límites y objetivos de los oponentes.
        - Historial de negociación de los oponentes.
        - Estrategias usadas por los oponentes.
    - Definir el protocolo y seleccionar la estrategia inicial. El **protocolo** es el conjunto de reglas de interacción que deben seguir los participantes. En la **estrategia** defines qué decisiones se van a tomar, y cómo y cuando van a ser estas tomadas.
- Modelo de negociación:
    - Intercambio de ofertas y intercambio de feedback sobre las mismas.
    - Argumentación / Intercambio de información.
    - Aprendizaje.
    - Adaptación de la negociación.
- Renegociación e implementación del acuerdo:
    - Formalización del acuerdo en un contrato.
    - Renegociar los flecos no solucionados o adaptar clausuras a nuevas condiciones.
    - Monitorización e implementación del acuerdo.

Las **tareas básicas** que deben llevar a cabo los **equipos de negociación**:

- Identificación de situación de conflicto.
- Formación del equipo.
- Acordar y planificar los atributos de negociación.
- Acordar y planificar protocolos de negociación.
- Seleccionar la estrategia inicial y dinámica de equipo inicial.
- Negociar y adaptar el equipo.

2.-TOMANDO DECISIONES EN GRUPO: SOCIAL CHOICE

Un **social choice** es un **grupo** de personas o agentes, cada uno con sus propias preferencias, que tienen como objetivo tomar una **decisión conjunta**. El social choice estudia la forma de **agregar las preferencias** de las personas. Formalmente la cuestión es como combinar las preferencias para obtener resultados sociales.

Los métodos estudiados en social choice surgen de forma natural en la sociedad. Además, ha sido estudiado por las matemáticas y la teoría de juegos principalmente.

Nociones básicas:

- Un conjunto de agentes N = {1,2,…,n} o votantes, estas son las entidades que expresarán sus preferencias.
- Un conjunto de posibilidades o candidatos O = $\{o_1,o_2,\dots,o_o\}$, si |O| = 2, tenemos una elección de parejas.
- Relaciones de preferencia $\succeq_i$:
  - Preferencia estricta $o_1 \succ_i o_2$.
  - Preferencia débil $o_1 \succeq_i o_2$.
  - Indiferencia $o_1 \sim_i o_2$.
- La relación de preferencia induce un orden de preferencias sobre O para cada uno de los agentes.
- F_N es la función de social choice o criterio del ganador.

Cada votante tiene sus preferencias sobre las distintas opciones (O), por lo que proporciona una ordenación sobre el conjunto de posibles opciones.

El **problema fundamental** de la teoría de social choice es que, dada una colección de órdenes de preferencia, una para cada votante, ¿cómo se puede combinarlas para obtener una decisión de grupo, que refleje lo más fielmente posible las preferencias de los votantes?

Existen dos variantes de **agregación de preferencias**:

- **Funciones de bienestar social**. Sea $\prod(O)$ el conjunto de ordenaciones de preferencia sobre O, una función de bienestar social determina, a partir de las preferencias de los votantes, un orden de preferencia social:

$$F_N: \underbrace{\prod(O) * \dots * \prod(O)}_{\text{n veces}} \to \prod(O)$$

- **Funciones de elección social**. Algunas veces solo necesitamos seleccionar una de las posibles opciones en vez de establecer un orden social. Una función de selección social determina, a partir de las preferencias de los votantes, una de las opciones:

$$F_N: \prod(O) * \ldots * \prod(O) \rightarrow O$$

n veces

Las **condiciones de Condorcet** son:

- Una opción o es ganadora Condorcet si para cualquier otra opción o', el número de agentes que o > o' es mayor o igual que el número de agentes que o' > o.
- Una opción o es perdedora Condorcet si para cualquier otra opción o', el número de agentes que o' > o es mayor o igual que el número de agentes que o > o'.
- $F_N(.)$ es Condorcet ganador si escoge como ganador al ganador Condorcet.
- $F_N(.)$ cumple el criterio Condorcet perdedor si excluye al Condorcet perdedor como ganador.

Las condiciones de Condorcet presentan un problema, la **paradoja Condorcet**, que viene a decir que existen situaciones en las que, sin importar el resultado que elijamos, una mayoría de los votantes no estarán contentos con el resultado elegido.

Existen dos **propiedades** claves en **procedimiento de votación**:

- La propiedad de Pareto.
- Independencia de alternativas irrelevantes.

El **criterio de Pareto** se basa en que si para todo votante i, $o >_i o'$, entonces $F_N(O) \neq o'$. En otras palabras, si todos prefieren a o más que a o', entonces o debe ser clasificado antes que o' en el resultado social.

Supongamos el espacio de votante particionado en dos subgrupos $N = \{N_1, N_2\}$. Denotamos a la elección social de N_1 como $F_{N1}(O)$ y a la elección social de N_2 como $F_{N2}(O)$. Entonces:

$$F_{N1}(O) \cap F_{N2}(O) \neq \emptyset \rightarrow F_N(O) = F_{N1}(O) \cap F_{N2}(O)$$

La **independencia de alternativas irrelevantes** (IIA) es que, si o_i se clasifica por encima de o_j en el resultado social, debe depender solamente de los ordenamientos relativos de o_i y o_j en los perfiles de los votantes.

Suponiendo que P es el perfil de preferencias de N y $F_N(O|P)=0$, y que P' es un perfil de preferencias basado en P donde la posición de o ha sido mejorada en al menos una posición, $F_N(.)$ es **monótono** si $F_N(O|P')=o$

La **votación** es el mecanismo de social choice por excelencia. Existen distintos tipos:

- Comparaciones por pares:
    - **Método de Dodgson**: elige el ganador condorcet, y en caso de que no exista busca el candidato más próximo al ganador condorcet, que sería el que requiere menor número de cambios en las preferencias para convertirse en ganador condorcet. Requiere el paso completo del ránking de los agentes. Se trata de un problema NP-duro.

- **Regla de Copeland**: busca la alternativa o cuyo índice Copeland IC(o) es mayor.

$$argmax\ IC(o) = \underset{o \in O}{\operatorname{argmax}} \sum_{\substack{o' \neq o \\ o' \in O}} p(o, o')$$

$$p(o, o') = \begin{cases} 1 & \#\ (o > o') > (o' > 0) \\ 0 & otherwise \end{cases}$$

  Pasa completamente de las preferencias.

- Posicionales:
  - **Pluralidad**: se basa en una función de elección social, que selecciona una única opción. Cada votante presenta sus preferencias. Cada candidato recibe un punto por cada orden de preferencia en que ocupa el primer lugar. El ganador es el que tiene mayor número de puntos. Es necesaria una regla para romper desempates. Solo requiere enviar la mejor opción. Existe una anomalía con la pluralidad, y es que puede salir una opción por ser la más votada, aunque haya una mayoría que prefiera a otra opción. En resumen, sólo considera al candidato mejor clasificado.
  - **Elecciones de mayoría secuenciales**: es una variante de la pluralidad, en el que los jugadores juegan en una serie de rondas, o bien una secuencia lineal o un árbol.
  - **Conteo Borda**: cada agente asigna una puntuación a cada candidato igual a su posición en su ránking de preferencias (k) descendente menos 1. Para cada candidato/opción se va contando la fuerza de la opinión a favor de dicho candidato. Si o_i aparece primero en un orden de preferencia, entonces se incrementa el conteo de o_i en k-1, a continuación, se incrementa el conteo para el siguiente candidato en ese orden de preferencia hasta que el candidato final en el orden de preferencia incrementa su conteo en 0. Después de hacer esto para todos los votantes, el candidato con mayor número de puntos es seleccionado como ganador. Es un criterio de semi-unanimidad. Requiere revelar todas las preferencias.
- Posicionales + comparación por pares.
  - **Pluralidad a dos rondas**: cada agente vota a su mejor candidato y los dos candidatos con mayor número de votos permanecen para una próxima votación, el resto son eliminados. Se vuelve a votar con solo los dos candidatos ganadores como opciones.
  - **Sistema de Hare**: cada agente vota a su mejor candidato y si uno de los candidatos recibió la mayoría de los votos, es ganador, sino, se elimina el candidato con menor número de votos y se repite el proceso hasta que uno de los candidatos gana por mayoría.
- Basados en agenda: elecciones por pares secuenciales lineales, pero el ganador depende del orden de la agenda.
  - **Proceso de enmienda**: los candidatos son enfrentados a pares hasta que una única opción queda como ganadora.
  - **Proceso sucesivo**: en orden, cada uno de los candidatos es enfrentado contra el resto. Si obtiene la mayoría es ganador. Sino, es eliminado del conjunto.

Los **grafos de mayorías** nos permiten ilustrar fácilmente como establecer una agenda para que gane un candidato. Se trata de un grafo dirigido con candidatos como vértices y un arco (i,j) si i

ganaría a j en una elección por mayoría simple. En un grafo, el ganador Condorcet sería el candidato que ganaría a todos los candidatos en una elección por parejas.

Este sería un resumen de la integración de criterios con sistemas:

Sistema	a	b	c	d	e
Enmienda	1	1	1	0	0
Sucesivo	0	1	1	0	0
Copeland	1	1	1	1	0
Dodgson	1	0	0	1	0
Pluralidad	0	0	1	1	1
Borda	0	1	1	1	1
Pluralidad 2 ron.	0	1	0	1	0
Hare	0	1	0	1	0

Abreviatura	Criterio
a	Ganador Condorcet
b	Perdedor Condorcet
c	Monotonía
d	Criterio Pareto
e	Consistencia

El **teorema de Arrow** es el resultado más importante de social choice. La función de bienestar W, da un ránking social de las alternativas. Los criterios justos y lógicos para Arrow son:

- **Criterio Pareto (P(W))**: si todos los agentes están de acuerdo en el orden entre candidatos, W debe seleccionar ese orden.
- **Independencia de alternativas irrelevantes (IIA(W))**: el orden entre dos candidatos debería depender solo del orden relativo de los candidatos dado por los agentes.
- **No dictatorial (¬D(W))**: no existe un solo agente cuyas preferencias siempre determinen el orden social.

No podemos encontrar un orden social que cumpla las tres propiedades a la vez. En muchas ocasiones no existe forma justa y lógica de agregar las preferencias individuales, es decir, no existe forma exacta de determinar la voluntad colectiva.

En el **teorema de Muller-Satterthwaite**, la función de elección social F, elige un ganador. Los criterios justos y lógicos son:

- **Criterio Pareto (P'(F))**.
- **Monotocidad (M(F))**.
- **No dictatorial (¬D'(W))**: F no es dictatorial si no existe un agente i tal que la elección de F siempre seleccione el mejor candidato para i.

En este caso, tampoco se pueden cumplir las tres condiciones a la vez.

En la **manipulación estratégica mediante votación táctica**, los votantes se benefician al tergiversar estratégicamente sus preferencias, es decir, la mentira.

Según el **teorema de Gibbard-Satterthwaite**, el único método de votación no manipulable que satisface la propiedad de Pareto para las elecciones con más de dos candidatos es una dictadura. Sin embargo, el teorema sólo nos dice que la manipulación es posible en principio, no da ninguna indicación de cómo tergiversar preferencias. Bartholdi, Tovey, y Trick demostraron que hay elecciones que son propensas a la manipulación, en principio, pero que la manipulación era computacionalmente compleja. El **Voto único transferible** es NP-duro de manipular.

3.-SUBASTAS

La asignación de recursos escasos entre un número de agentes es un tema central en sistemas multiagente. Los recursos pueden ser:

- Un objeto físico.
- Un derecho.
- Recursos computacionales.

Si el recurso no es escaso, no hay problemas para su asignación. Si no hay competencia por los recursos, entonces no hay problemas para su asignación.

Las **subastas** están relacionadas con los comerciantes y sus asignaciones de las unidades de un bien indivisible, y el dinero, que es divisible. En éstas se asume alguna asignación inicial. El intercambio es la libre modificación de la asignación de los bienes y dinero entre los comerciantes. En general tienden a presentar las siguientes características:

- Distribución de uno o más bienes discretos.
- 1 vendedor vs n compradores.
- Monoatributo.
- Mediadas o no.
- Distributiva.

Las **ventajas** de las subastas es que son un fenómeno usado socialmente, es decir, es conocido por usuarios y desarrolladores, y el protocolo de interacción es bien definido y cerrado.

Las **desventajas** son que no permite intercambiar información y que se suelte tratar solo negociaciones monoatributo.

Las subastas son estudiadas por el diseño de mecanismos, se buscan propiedades matemáticas interesantes como:

- Maximizar las ventas del vendedor.
- El bien es distribuido al comprador que más lo desea.
- Veracidad de los participantes.

3.1.-SUBASTAS POR UN ÚNICO BIEN

Las **subastas por un único bien** presentas las siguientes características:

- Un vendedor y n compradores.
- Negociación por un único bien o producto.
- Cada comprador valora en mayor o menor grado el producto.
- Los compradores prefieren pagar cuanto menos mejor.
- Los vendedores prefieren cobrar cuanto más mejor.

Hay de diversos tipos:

- **Subasta inglesa**. El vendedor ajusta un precio inicial para el producto, el mínimo para el vendedor. Los compradores pujan un valor mayor que el anterior y la puja suele ser

pública. La condición de terminación es un tiempo fijo o tras no recibir nuevas pujan durante un tiempo. El ganador paga el dinero que pujó.

- **Subasta japonesa**. Es de precio ascendente y el vendedor indica un precio inicial. A cada paso, todos los compradores indican de forma pública si están dispuestos a aceptar el precio. Los que no, abandonan. El vendedor incrementa el precio hasta que solo queda un comprador. El ganador paga el precio de la anterior ronda.
- **Subasta holandesa**. Es de precio descendente y el vendedor indica un precio máximo. Va descendiendo a cada paso el precio a cada tick de un reloj público. La subasta termina cuando alguno de los compradores indica que acepta la oferta. Es usado cuando se quiere vender rápido.
- **Subastas de sobre cerrado**. Realmente se trata de una familia de subastas. La puja no es pública, sino privada a un mediador. Todos los compradores pujan un valor al mediador (one-shot). El ganador es aquel comprador cuya puja es más alta. Hay varios tipos dependiendo de lo que paga el ganador:
    - De primer precio.
    - De segundo precio (Vickey auction).
    - De k-ésimo precio.

Una estrategia es dominante cuando esta es la más provechosa independientemente de la estrategia de los otros jugadores. Si vemos las subastas como un juego para los compradores:

- Cada participante es un agente.
- Cada agente tiene un máximo que desea pagar por el bien. Esta cantidad también representa su preferencia por el bien.
- Los agentes quieren simplemente maximizar su beneficio esperado, por lo que son neutrales al riesgo.
- Idealmente, el bien tendría que ir a aquel que lo valora más.

Estrategias dominantes en las subastas vistas:

- Subasta inglesa: pujar la unidad incremental mínima hasta que el resto de los agentes alcancen su máximo.
- Subasta japonesa: continuar en el proceso siempre que nuestro límite máximo lo permita. Decir la verdad es una estrategia dominante cuando nuestro máximo no depende de lo que hagan otros.
- Subasta holandesa: no tiene.
- Subasta de sobre cerrado de primer precio: no tiene.
- Subasta de sobre cerrado de segundo precio: decir la verdad y pujar por nuestro precio límite.

La subasta holandesa y la subasta de sobre cerrado de primer precio son estratégicamente equivalentes. La subasta japonesa y la de sobre cerrado de segundo precio también lo son cuando los valores máximos no dependen de lo que otros hacen.

La subasta que debemos escoger si somos el subastador es:

- Desde el punto de vista del beneficio teórico esperado: si los agentes son neutrales al riesgo y la función de valuación es independiente, da igual.
- Computacionalmente: los protocolos con estrategias dominantes son más baratos, pero hay que tener en cuenta el coste de las comunicaciones.
- Desde el punto de vista del beneficio real: la valuación de un objeto depende de lo que observamos, por lo que la subasta inglesa es mejor que la de sobre cerrado de segundo precio. Éstas últimas son mejores que la holandesa y la de sobre cerrado de primer precio.

Existen dos problemas conocidos son las subastas por un único bien:

- **Collusion**: los compradores pueden aliarse para pagar menos por un producto. Esto es dominante en la subasta inglesa y la de sobre cerrado de segundo precio.
- **Lying auctioneer**: el propio subastador puja por el producto para subir su valor real.

3.2.-SUBASTAS POR MÁS DE UNA UNIDAD

Existen diversos tipos:

- Subastas de paquetes de productos (bienes).
- Subasta de sobre cerrado:
    - Dos tipos de reglas:
        - Discriminatoria: cada uno paga lo que ofrece.
        - Uniforme: todos los ganadores pagan lo mismo. La cantidad más alta de los perdedores, o la más baja de los ganadores.
    - Tipos de puja:
        - Todo o nada: deseo de un número fijo de bienes a un precio.
        - Divisible: acepta una cantidad menor al mismo precio/unidad.
    - Los desempates se pueden hacer por cantidad, tiempo, etc.
- Subasta inglesa: los ganadores se deciden cunado se consigue vender todos los bienes de una. Presenta problemas similares a la subasta de sobre cerrado. La cantidad mínima incrementable de una ronda a otra son dos dimensiones.
- Subasta japonesa: en vez de decir si se sigue o no, los compradores dicen el número de unidades que están dispuestos a pagar a dicho precio. El número de unidades va descendiendo. La subasta termina cuando la demanda iguala o supera el número de unidades a vender.
- Subasta holandesa: el precio por unidad va descendiendo y los compradores indican cuantas unidades desean comprar a ese precio. Si la cantidad no es todas las unidades, se continua con la subasta.

3.3.-SUBASTAS COMBINATORIAS

Son subastas por múltiples bienes distintos o por múltiples unidades de múltiples bienes distintos.

Una modelización de este problema sería:

- Sea G={1,2,3,…,M} un conjunto de bienes a subastar.
- 1 vendedor.
- Los N compradores diferentes son un conjunto de agentes Ag={1,2,…,n}, en los que se captura las preferencias de cada agente i con la función de utilidad o valuación:

  $$U_i(.) = 2^G \mapsto \Re$$

  que significa que para cada posible conjunto de bienes $g \subseteq G$, $U_i(g)$ dice lo bueno que es g para i. Si $U_i(\emptyset) = 0$, entonces se dice que la función de valuación de i se normaliza.

- Otra idea útil es la libre disposición:

  $$g_1 \subseteq g_2 \; implica \; qué \; U_i(g_1) \leq U_i(g_2)$$

  en otras palabras, para un agente nunca es peor tener más cosas.

- Las pujas son por un conjunto de bienes variables y un precio. Un agente puede hacer muchas pujas distintas.
- Las pujas expresan en cierta manera preferencias por ciertos conjuntos de objetos:
  - Sustitubilidad: $U_i(G_1 \cup G_2) < U_i(G_1) + U_i(G_2)$
  - Complementariedad: $U_i(G_1 \cup G_2) > U_i(G_1) + U_i(G_2)$

Formalmente una asignación es una lista de conjuntos de $g_1,\ldots,g_n$, uno para cada agentes Ag_i con la condición de que:

$$g_i \subseteq G$$

$$\forall i, j \in Ag, tal \; que \; i \neq j, se \; cumple \; que \; g_i \cap g_j = \emptyset$$

No se asigna un mismo bien (producto) a más de un agente. El conjunto de todas las asignaciones G a los agentes Ag es:

$$alloc(G, Ag)$$

Una forma natural de determinar la asignación es **maximizar el bienestar social**, es decir, la **suma de las utilidades de todos los agentes**. Definimos una función de bienestar social de la siguiente forma:

$$F_N(g_1, \ldots, g_n, u_1, \ldots, u_n) = \sum_{i=1}^{n} u_i(g_i)$$

Dado un conjunto de bienes G y un conjunto de funciones de valuación (utilidad) $u_1,\ldots,u_n$ una por cada agente i ϵ Ag, el objetivo es encontrar una asignación:

$$g_1^*, g_2^*, \ldots, g_n^*$$

que maximice F_N, es decir:

$$g_1^*, g_2^*, \ldots, g_n^* = \underset{(g_1,\ldots,g_n) \in alloc(G,Ag)}{\operatorname{argmax}} F_N(g_1, \ldots, g_n, u_1, \ldots, u_n)$$

La condición de **optimalidad** se cumple cuando lo hace la función anterior. Para cumplir la condición de **correctitud**, ningún objeto debe ser distribuido más de una vez y a cada comprador solo se le asigna como mucho un subconjunto. Esta distribución de recursos se trata de un caso

especial del **problema de empacamiento de conjuntos**, que es NP completo. Se podría conseguir que cada agente i declarase su valuación u_i, pero esta valuación es lo que dice el agente, no necesariamente lo que es, y el agente puede mentir. En conclusión, basta con ver todas las posibles asignaciones y decidir cuál es la mejor.

Ver todas las posibles asignaciones supone un problema, la representación, ya que las valoraciones son exponenciales:

$$U_i(.) = 2^G \mapsto \Re$$

Una representación simple es poco práctico, pero buscar todas las posibles valoraciones es computacionalmente intratable.

El posible número de pujas para un comprador es exponencial con el conjunto de bienes a la venta. Esto presenta dos problemas:

- **Problemas de evaluación y selección de pujas a enviar**.
- **Problemas de representación**: lenguajes de expresión de pujas.

El **lenguaje de pujas** permite a los licitadores la construcción de las valoraciones de las ofertas que quieren mencionar. Se define de la siguiente manera:

- Ofertas atómicas (g,p) donde $g \subseteq G$.
- Un lote g' satisface una oferta (g,p) si $g \subseteq g'$. En otras palabras, un lote cumple una oferta si contiene por lo menos los productos de la oferta.
- Las ofertas atómicas definen las siguientes valoraciones:

$$u_\beta(g') \begin{cases} p \; si \; g' satisface \; (g,p) \\ 0 \; en \; cualquier \; otro \; caso \end{cases}$$

  Las ofertas atómicas por si solas no nos permiten construir valoraciones muy interesantes.

Para construir valoraciones más complejas, las ofertas atómicas pueden ser combinadas en ofertas más complejas, como las ofertas XOR:

$$\beta = (g_1, p_1) XOR \ldots XOR \; (g_k, p_k)$$

define la siguiente valoración u_β:

$$u_\beta(g') \begin{cases} 0 \; si \; g' no \; satisface \; ninguna \; oferta (g_i, p_i) \\ \max \{p_i | g_i \subseteq g'\} \; en \; cualquier \; otro \; caso \end{cases}$$

Las ofertas XOR son totalmente expresivas, es decir, pueden expresar cualquier función de valoración sobre un conjunto de mercancías. Para ello, es posible que necesitemos un número exponencial de ofertas atómicas. Sin embargo, la valoración de un lote se puede calcular en tiempo polinómico.

El problema básico es intratable, pero esto sucede en el peor caso, por lo tanto, es posible desarrollar aproximaciones óptimas que funcionan bien en muchos casos. También puede resultar adecuado renunciar al óptimo y utilizar:

- Heurísticas.

- Buscar algoritmos de aproximación. Una aproximación muy común es codificar el problema como un programa literal entero y utilizar un solucionador estándar, esto funciona frecuentemente en la práctica.

En general no sabemos si las valoraciones que nos proporcionan los agentes son veraces o falsas, pero podemos hacer estas valoraciones veraces en una subasta Vickrey a través del mecanismo Vickrey/Clarke/Groves o VCG. Este mecanismo es compatible con los incentivos, en los que decir la verdad es una estrategia dominante. La notación sería la siguiente:

- Una función de valoración indiferente ($u^0(G)=0$) para todo G.
- F_{N-i} es la función de satisfacción social sin i:

$$F_{N-i}(g_1, \dots, g_n, u_1, \dots, u_n) = \sum_{j \in Ag | j \neq i} u_j(g_j)$$

El **mecanismo VCG** se define como:

1. Cada agente declara una valoración u_i simultáneamente.
2. El mecanismo VCG calcula:

$$g_1^*, g_2^*, \dots, g_n^* = \underset{(g_1,\dots,g_n) \in alloc(G,Ag)}{\operatorname{argmax}} F_N(g_1, \dots, g_n, u_1, \dots, u_n)$$

   y es elegida la asignación:

$$g_1^*, g_2^*, \dots, g_n^*$$

3. El mecanismo calcula también, para cada agente i:

$$g_1^*, g_2^*, \dots, g_n^* = \underset{(g_1,\dots,g_n) \in alloc(G,Ag)}{\operatorname{argmax}} F_N(g_1, \dots, g_n, u_1, \dots, v^0, \dots, u_n)$$

   la asignación que maximiza la satisfacción social es la del agente que ha declarado u^0 (función de valoración indiferente) como su valoración

4. Cada agente i paga p_i, donde:

$$P_i = F_{N-i}(g'_1, \dots, g'_n, u_1, \dots, u^0, \dots, u_n) - F_{N-i}(g_1^*, \dots, g_n^*, u_1, \dots, u_i, \dots, u_n)$$

   en otras palabras, cada agente paga su coste, a los otros agentes, después de haber participado en la subasta.

Si yo hago una oferta mayor que mi valoración y gano, entonces yo acabo pagando lo que vale el bien para todos los demás, que es más de lo que vale para mí. Si oculto mi subasta, reduzco mi probabilidad de ganar, pero incluso si gano aún estoy pagando lo que todo el mundo piensa que vale el bien, por lo que no ahorro dinero reduciendo mis probabilidades de ganar. Por lo tanto, tenemos una estrategia dominante para cada agente que garantiza maximizar el bienestar social.

3.4.-INTERCAMBIOS

Los **intercambios** son subastas donde todos pueden actuar como compradores y vendedores.

Existen principalmente dos tipos:

- **Distribución de recursos**: subastas a dos bandas.
- **Agregación de información**: mercados de predicción.

En las **subastas a dos bandas**, hay muchos compradores y muchos vendedores de un bien. Cada comprador y cada vendedor hace una puja de compra/venta respectivamente con precio y número de unidades. Las pujas son almacenadas en un servidor central, definiendo dos tipos:

- Subastas a dos bandas continua.
- Subastas a dos bandas periódica.

Las ofertas de compra y venta son agrupadas de forma que se compra/venden las unidades deseadas. Si no es posible, el remanente se introduce como nueva oferta para próximas distribuciones. El agrupamiento de compra es posible si el precio de venta es menor o igual que el de compra.

El precio del producto es determinado mediante el algoritmo especificado por el mercado. Se encuentra entre el precio de la oferta de venta y el de la oferta de compra.

En los **mercados de predicción**, se busca agregar creencias de agentes en vez de distribuir bienes. Se abre un mercado con contratos (c1,c2,…,ck), donde ci es un compromiso a pagar al portador si el candidato i gana es vencedor. Los agentes compran y venden a su antojo. También valoran ci de acuerdo a la probabilidad que ellos creen que la opción i será la vencedora. Si la probabilidad de que i gane es mayor que su precio de venta, el agente querrá comprar contratos. Si la probabilidad de que i gane es menor que su precio de venta, el agente querrá vender sus contratos. Hay evidencias que apuntan a que los mercados de precisión son más efectivos que las encuestas.

4.-NEGOCIACIÓN BILATERAL Y MULTIPARTICIPANTE

En zero sum encounter no existe la posibilidad de lograr un acuerdo, pero en muchos otros escenarios, existe la posibilidad de un acuerdo mutuamente beneficioso sobre asuntos de interés común. Las capacidades de negociación y argumentación son fundamentales para la capacidad de un agente de alcanzar tales acuerdos.

La negociación se rige por mecanismos particulares o protocolos. El mecanismo define las “reglas del encuentro” entre agentes. El diseño de éstos contempla que satisfagan ciertas propiedades deseables, como la Pareto eficiencia.

Las **similitudes** entre las **subastas** y el **social choice** son:

- Nos permiten resolver conflictos en grupo y muliparticipante.
- Los protocolos usados son rígidos y no permiten mucho intercambio de información.
- Normalmente se trata de un solo atributo o bien un conjunto finito de opciones.

En cuanto a sus **diferencias**:

- Las **subastas** solo se ocupan de la asignación de bienes. Se requieren técnicas más ricas de alcanzar acuerdos.
- La **negociación** es el proceso de alcanzar acuerdos sobre asuntos de interés común.

Cualquier entorno de negociación tendrá cuatro componentes:

- Un **conjunto de negociación**: las posibles propuestas que los agentes pueden hacer.
- Un **protocolo**.
- **Estrategias**, una por cada agente, que son privadas.
- Una **regla** que determina cuando se ha alcanzado un acuerdo y cuál es el trato del acuerdo.

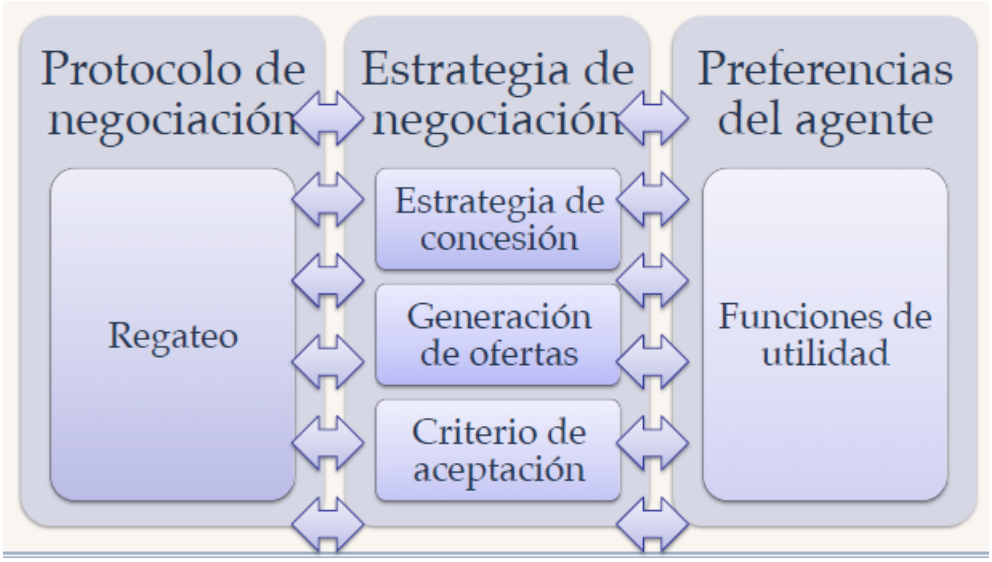
Frecuentemente la negociación ocurre en una serie de rondas con propuestas en cada ronda. Hay que encontrar múltiples soluciones a la complejidad inherente de la negociación como:

- El número de ofertas posibles es exponencial al número de soluciones.
- Es difícil comparar las ofertas a través de las múltiples soluciones.

Al final concluimos que no hay mucho que diferencia las subastas de la negociación.

4.1.-NEGOCIACIÓN BILATERAL

La **negociación bilateral y multilateral** posee múltiples atributos y valores, por lo que los espacios de búsqueda son muy grandes. En éstos, es complicado alcanzar un acuerdo óptimo en dichos espacios. A su vez, no nos valen protocolos one-shot para alcanzar un acuerdo válido, ya que requerimos intercambiar más información u ofertas.



El **regateo** también es conocido como el protocolo de intercambio de ofertas de Rubinstein. Se trata de un protocolo uno a uno que se divide en ronda. Cada ronda está constituida de 1 oferta por uno de loa agentes, 1 contraoferta del otro agente. En cualquier momento uno de los agentes puede abandonar el proceso. Asumimos que las negociaciones no son infinitas, cada agente tiene una deadline privada para terminar la negociación.

La **función de utilidad** es una función matemática abstracta que encapsula las preferencias o gustos del agente. Esta información es personal y privada para cada uno de los agentes. A su vez, determina cuánto le gusta al agente una determinada oferta. El objetivo es que el acuerdo final maximice esta función. La función de utilidad puede depender de factores distintos.

Las **funciones de utilidad lineales** son el modelo de preferencia multiatributo más común:

$$U_{ai}(X) = w_{i,1}V_{i,1}(x_1) + w_{i,2}V_{i,2}(x_2) + \cdots + w_{i,N}V_{i,N}(x_N)$$

$$U_{ai}(.) \in [0,1], V_{i,j}(.) \in [0,1]$$

$$\sum_{j}^{N} w_{i,j} = 1$$

Las reglas del protocolo no significan que siempre se alcanzará un acuerdo. En este caso, hablamos de conflicto deal o oferta conflicto. Se adoptan las siguientes suposiciones básicas:

- El desacuerdo es el peor resultado.
- Los agentes buscan maximizar su utilidad.

Un jugador racional siempre elige la estrategia que maximiza su utilidad esperada. Podemos asumir que todos los jugadores son racionales y el juego y la racionalidad de los jugadores es un conocimiento compartido por todos. De esta asunción surgen dos tipos de soluciones:

- Soluciones centradas en el individuo:
  - **Estrategia dominante**.
  - **Equilibrio de Nash**.
- Soluciones sociales:
  - **Optimalidad de Pareto**.
  - **Bienestar social**.

Una estrategia $s \in S_i$ es dominante para el jugador i si independientemente de lo que haga el oponente, i no puede estar mejor haciendo cualquier otra cosa. Un perfil estratégico $(s_1, s_2, \dots, s_n)$ está en **equilibrio de estrategias dominantes** (ESD) si s_i es la estrategia dominante de i.

Un perfil estratégico $(s_1, s_2, \dots, s_n)$ está en **equilibrio de Nash** si para cada i y cada $s' \in S_i$, $U_i(s_1, \dots, s_i, \dots, s_n) \geq U_i(s_1, \dots, s', \dots, s_n)$.

Todo juego en que cada jugador tiene un conjunto finito de estrategias tiene al menos un equilibrio de Nash de estrategias mixtas.

Un resultado $o \in O$ es un **óptimo de Pareto** si no hay otro $o' \in O$ tal que $U_i(o') \geq U_i(o)$ para todos los $i \in N$ y $U_i(o') > U_i(o)$ para algún $i \in N$.

El **bienestar social** de un resultado $o \in O$ se mide de dos maneras fundamentales:

- Suma de las utilidades de los agentes:

$$W(o) = \sum_{i \in N} U_i(o)$$

- Producto de las utilidades de los agentes:

$$W(o) = \prod_{i \in N} U_i(o)$$

El resultado que maximiza el bienestar es:

$$o* = \underset{o \in O}{\operatorname{argmax}} W(o)$$

Como tenemos un número infinito de equilibrios de Nash, el concepto de solución de NE es demasiado débil para ayudarnos. Podemos obtener resultados únicos si tenemos en cuenta el tiempo. Una forma estándar de modelar esta impaciencia es descontar el valor del resultado:

- Cada agente tiene un $\delta_i, i \in \{1,2\}$, donde $0 \leq \delta_i \leq 1$.
- Cuando más próximo a 1 está δ_i más paciente es el agente.

Si al agente i se le ofrece x entonces el valor del trozo es $\delta_i^k x$ en el instante k.

Un juego de una ronda sigue siendo un ultimátum game. Un juego de 2 rondas significa que el agente 2 puede jugar como antes, pero si lo hace así, solo obtendrá δ_2. En el caso general, el agente 1 hace la propuesta que ofrece al agente 2 y que el agente 2 sería capaz de imponer en la segunda ronda:

- El agente 1 obtiene:

$$\frac{1-\delta_2}{1-\delta_1\delta_2}$$

- El agente 2 obtiene:

$$\frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}$$

La **aproximación heurística** es un enfoque más sencillo ya que aproxima el valor que varía para los jugadores. Algunas aproximaciones comunes son:

- **Lineal**: incremento lineal desde el precio inicial en el instante de inicio hasta el precio de reserva en el plazo máximo.
- **Boulware**: aumento muy lento hasta la proximidad del plazo máximo y luego un aumento exponencial.
- **Concesión**: incremento exponencial inicial hasta la proximidad del precio de reserva y luego no cambia mucho.

Para evaluar la bondad de un modelo de negociación utilizamos los siguientes criterios:

- Criterios computacionales: número de rondas de negociación.
- Criterios de eficiencia económica:
    - Utilidad individual.
    - Utilidad conjunta.
    - Distancia al punto Pareto óptimo más cercano (acuerdo Pareto eficiente). Un acuerdo es **Pareto eficiente** si no se puede mejorar la utilidad de uno de los dos agentes sin empeorar la utilidad de otro.
    - Distancia al punto de regateo de Nash. El **punto de regateo de Nash** es el punto que maximiza la multiplicación de la utilidad de los dos agentes (utilidad conjunta).

En la **estrategia de concesión** la negociación es un proceso de concesión hacia un punto de acuerdo común. Empezamos con unas aspiraciones, y las vamos reduciendo a medida que

vemos que el acuerdo no es posible. Cómo, cuánto y cuándo concedemos es lo que marca la estrategia de concesión, es decir, nuestro nivel de aspiración actual.

La fórmula para la estrategia de concesión es:

$$p_{ai}(t) = \begin{cases} IP^{ai} + s_{ai}(t) * \left(RP^{ai} - IP^{ai}\right) a_i = b_i \\ RP^{ai} + \left(1 - s_{ai}(t)\right) * \left(IP^{ai} - RP^{ai}\right) a_i = s_i \end{cases}$$

En la **estrategia de concesión temporal** se concede a medida que avanzan las rondas de la negociación:

$$s_{ai}(t) = \left(\frac{t}{T_{ai}}\right)^{\frac{1}{\beta_{ai}}}$$

$$\beta_{ai} = 1 = lineal$$

$$0 < \beta_{ai} < 1 = boulware$$

$$1 < \beta_{ai} < \infty = conceder$$

La estrategia de concesión basada en el comportamiento es lo que se conoce como el Tit-for-Tat o Toma y Daca. Consiste en conceder lo que creemos que ha concedido el otro:

$$s_{a_i}(X_{b\to a}^{t-1}, X_{b\to a}^{t-\delta}, X_{b\to a}^{t-\delta+1}) = \min(1, \max(RU_{a_i}, \frac{1-U_{a_i}(X_{b\to a}^{t-\delta+1})}{1-U_{a_i}(X_{b\to a}^{t-\delta})} U_{a_i}(X_{b\to a}^{t-1}))) \quad \text{Relativo}$$

$$s_{a_i}(X_{b\to a}^{t-1}, X_{b\to a}^{t-\delta}, X_{b\to a}^{t-\delta+1}) = \min(1, \max(RU_{a_i}, U_{a_i}(X_{b\to a}^{t-1}) - (U_{a_i}(X_{b\to a}^{t-\delta+1}) - U_{a_i}(X_{b\to a}^{t-\delta})))) \quad \text{Absoluto}$$

$$s_{a_i}(X_{b\to a}^{t-1}, X_{b\to a}^{t-\delta}, X_{b\to a}^{t-1}) = \min(1, \max(RU_{a_i}, \frac{1-U_{a_i}(X_{b\to a}^{t-1})}{1-U_{a_i}(X_{b\to a}^{t-\delta})} U_{a_i}(X_{b\to a}^{t-1}))) \quad \text{Promediado}$$

En la **generación de ofertas**, se generan ofertas en base a la utilidad marcada por la estrategia de concesión. Puede haber desde 0 hasta infinitas ofertas con un nivel de utilidad dado.

En el caso de **funciones de utilidad lineales** se debe:

- Hacer la oferta lo más satisfactoria para el oponente dentro de lo posible.
- Intentar averiguar qué atributos son más interesantes para el oponente:
    - El oponente puede darnos información parcial sobre sus preferencias.
    - Usar algoritmos para intentar aprender aproximadamente los pesos dados a cada atributo.
        - Observar en qué atributos se concede más y en cuáles menos.
        - Aplicar otras técnicas avanzadas de IA y Aprendizaje.

Para todo esto podemos utilizar una **función heurística de similaridad**, es decir, escoger la oferta con el nivel de aspiración actual que más se parece a la anterior. Puede ser utilizando dos bases distintas de similaridad:

- **Similaridad en base a distancia euclídea** (independiente del dominio).
- **Similaridad en base a criterios de similaridad difusa** (requiere información de expertos e información sobre los pesos del oponente).

También podemos hacer uso de **algoritmos genéricos**, que realizan operaciones de cruce y mutación sobre ofertas del oponente y propias, o basándonos en **modelos estándar** para determinados perfiles de agente.

Los **criterios de aceptación** se pueden basar en muchas cosas, por ejemplo, en condiciones del entorno o en conocimiento sobre el oponente. El **criterio racional** es uno de los más aceptados:

$$accept_{ai}(X,t) = \begin{cases} accept\ U_{ai}(X) \geq s_{ai}(t+1) \\ reject \qquad\qquad otherwise \end{cases}$$

Un **dominio orientado a la tarea (Task Oriented Domain -TOD)** es una tripleta:

(T, Ag, c)

donde:

- T es el conjunto (finito) de todas las tareas posibles.
- Ag = {1,…,n} es el conjunto de los agentes participantes.
- $c: \mathcal{P}(T) \rightarrow \mathcal{R}$ define el coste de ejecutar cada subconjunto de tareas.

Un encuentro es una colección de tareas:

$$< T_1, \dots, T_n >$$

donde $T_i \subseteq T$ para cada $i \in Ag$.

Dado un encuentro $< T_1, T_2 >$ una oferta será una asignación de las tareas $T_1 \cup T_2$ a los agentes 1 y 2. El coste para i de la oferta $\delta =< D_1, D_2 >$ es $c(D_i)$ y lo denotaremos por $cost_i(\delta)$. La utilidad de la oferta δ del agente i es:

$$utility_i(\delta) = c(T_i) - cost_i(\delta)$$

La oferta conflicto, $\ominus$, es la oferta $< T_1, T_2 >$ constituida por las tareas asignadas originalmente. Observar que $utility_i(\ominus) = 0$, para todo $i \in Ag$. La oferta δ es racional individualmente si da una utilidad positiva.

El conjunto de ofertas sobre las que negocian los agentes son aquellas que son:

- **Individualmente racional**: los agentes no estarán interesados en ofertas que dan utilidad negativa, ya que preferirán la oferta conflicto.
- **Pareto eficiente**: los agentes siempre pueden transformar una oferta no Pareto eficiente en una oferta Pareto eficiente haciendo a un agente más feliz y que ninguno de los otros esté peor.

Las reglas del protocolo de concesión monótono son:

- La negociación se realiza en rondas.
- En la ronda 1, los agentes proponen simultáneamente una oferta del conjunto de negociación.
- Se alcanza un acuerdo si un agente considera que la oferta propuesta por los otros es al menos tan buena o mejor que su propuesta.
- Si no se alcanza un acuerdo entonces la negociación continua en otra ronda de propuestas simultáneas.
- En la ronda u+1, no se permite a ningún agente realizar una propuesta que sea menos preferida por los otros agentes que la oferta propuesta en el instante u.

- Si ningún agente realiza una concesión en alguna ronda (u>0), entonces la negociación acaba con la oferta conflicto.

Existen tres asunciones:

- La primera oferta de un agente es su oferta más preferida.
- El agente que debe ceder es el agente menos dispuesto a correr el riesgo de conflictos.
- Un agente debería conceder solo lo suficiente para cambiar el equilibrio de riesgos.

La disposición a correr riesgo de conflicto supone que has concedido mucho. Entonces:

- Tu propuesta está ahora cerca de la oferta conflicto.
- En el caso en que ocurra un conflicto, no estás mucho peor.
- Estás más dispuesto a arriesgar en un conflicto.

Un agente estará más dispuesto a arriesgar en un conflicto si la diferencia de utilidad entre su propuesta actual y la oferta conflicto es baja.

La **estrategia de Zeuthen** está en equilibrio Nash bajo la suposición que un agente está usando la estrategia y el otro no puede hacer nada mejor que utilizarla el mismo. Esto es de particular interés para el diseñador de agentes automáticos. Se elimina cualquier necesidad de secreto por parte del programador. La estrategia de un agente puede ser de conocimiento público, y ningún otro diseñador de agentes puede explotar la información eligiendo una estrategia diferente. De hecho, es deseable que la estrategia sea conocida, para evitar conflictos involuntarios.

Las **funciones de utilidad complejas** utilizan espacios de búsqueda. Hay que modelar este tipo de preferencias complejas como:

- Matrices de influencia.
- Restricciones hiperrectángulos.
- Grafos de utilidad.
- Restricciones hiper-cónicas.
- Redes o grafos CP.
- Independientes del tipo de preferencias.

Los modelos presentados hasta ahora son ciegos, solo se centran en el proceso actual de negociación. En la realidad podemos estar llevando a cabo varias negociaciones a la vez para obtener el mismo producto, las llamadas outside options. Los entornos son dinámicos y aparecen y desaparecen oportunidades.

Las **outside options** pueden influir en:

- **Utilidad de reserva**: ésta puede ser actualizada a la mejor oferta recibida de entre todos los oponentes.
- **Estrategia de concesión**: podemos conceder menos cuando hay más outside options.
- **Aceptación de ofertas**: podemos rechazar ofertas aceptables si consideramos que en un futuro vamos a recibir mejores.

La **negociación multilateral** puede entenderse como:

- La coordinación en paralelo de muchas negociaciones por los mismos productos.
- La coordinación en paralelo de negociaciones cuyos resultados están relacionados.
- Muchos agentes negociando a la vez para alcanzar un acuerdo conjunto. Puede verse como una mejora del social choice. Muchos modelos requieren mediador debido a la alta complejidad para alcanzar acuerdos.

TEMA 6 (II).- COORDINACIÓN

1.-INTRODUCCIÓN

El problema de coordinación de múltiples agentes surge en múltiples aplicaciones, tanto en la naturaleza como en sistemas artificiales.

Los actos comunicativos simples no son conversaciones. Un acto comunicativo es bien el originador de una conversación, bien la consecuencia de un acto comunicativo previo. Una conversación proporciona el potencial para la coordinación, cooperación y negociación.

La coordinación en SMAs trata sobre cómo los agentes se comportan **individual** y **socialmente** para que, por un lado, se satisfagan los **objetivos personales** y, por el otro, los **globales**. La coordinación es necesaria ya que los recursos son **limitados** y se utiliza para que:

- La **experiencia/pericia** esté repartida entre los agentes y evitar el caos.
- Se cumplan un conjunto de restricciones globales o se cumplan roles especializados.
- Exista interdependencia entre objetivos de unos y acciones de otros, es decir, eficiencia.

La coordinación se consigue mediante la distribución de los datos y el control, con lo que la coherencia global es difícil.

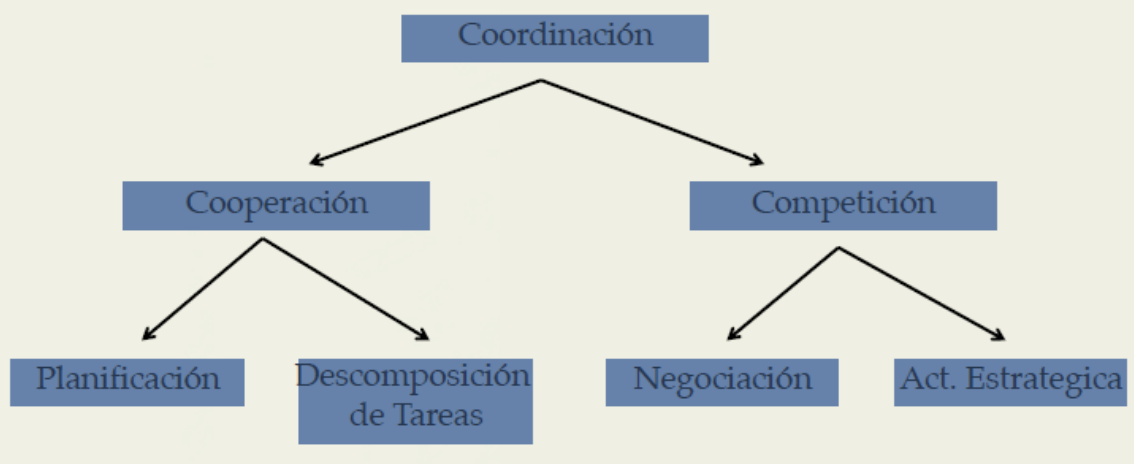
Para solucionar el problema de la coordinación se necesita modelar, en la fase de diseño, las posibles interdependencias que pueda haber entre agentes, sus acciones y objetivos.

Se utiliza un protocolo de interacción porque:

- Los agentes de un SMA actúan guiados por la consecución de sus objetivos individuales (**self-interested**), es decir, compiten. En estas situaciones los protocolos de interacción son orientados a maximizar los valores de la función de utilidad generados por las acciones de los agentes.
- Existen situaciones en la que comparten objetivos comunes (**benevolent**), es decir, cooperan. En éstas, los protocolos de interacción se enfrentan a:
    - Trabajar con objetivos comunes o tareas comunes.
    - Evitar conflictos en la medida de lo posible y mantener un flujo correcto de conocimiento y evidencias.

Si todo un sistema es nuestro (sistema cerrado), podemos diseñar agentes que se ayuden entre ellos siempre que se les requiera. En este caso se puede asumir que los **agentes** son **benévolos**, es decir, nuestro mejor interés es su mejor interés. La solución de problemas en los sistemas benévolos es la **Resolución Distribuida de Problemas Cooperativos**. La **benevolencia** simplifica enormemente el diseño de tareas.

Si los agentes representan individuos u organizaciones, no se puede realizar la asunción de benevolencia. Se asumirá que los agentes actuarán para conseguir sus propios intereses, posiblemente a expensas de los de otros, es decir, serán **agentes auto-interesados**. Tienen potencial para el conflicto o competición. A su vez puede complicar enormemente el diseño de tareas.



Existen dos modos principales de resolver problemas cooperativamente:

- **Compartir tareas**: se distribuyen los componentes de una tarea entre los agentes.
- **Compartir resultados**: se distribuyen la información.

En ambos casos, hay que tener en cuenta quién puede comunicarse con quién utilizando un **grafo de comunicaciones**.

Un grafo G = (V, E) se define por:

- $V = \{v_1,\ldots,v_N\}$, conjunto de vértices del grafo, es decir, los agentes.
- E: V x V, conjunto de arcos (v_i, v_j), en los que el agente v_i conoce y puede comunicarse con el agente v_j.

Un **grafo balanceado** tiene el mismo número de arcos de entrada que de salida, para todos los nodos.

Los **grafos AND/OR de tareas y objetivos** representan las dependencias entre objetivos y tareas que se necesitan para cumplir objetivos primitivos (nodos hoja del árbol). Algunas suposiciones son:

- Sean dos agentes, Agente1 y Agente2, han de cumplir los objetivos $G^1{}_0$ y $G^2{}_0$.
- $G_{in1,n2,\ldots,nm}$ es un subobjetivo del agente i, hijo nm-ésimo de $G_{in1,n2,\ldots,nm-1}$.
- El conjunto de superíndices a1, a2,…,an en un objetivo indica que los agente con esos índices deben cumplir todos los objetivos.

Las **interdependencias** se pueden producir entre nodos del grafo. Hay dos tipos de éstas, **débiles** o **fuertes**, dependiendo de si se debe satisfacer necesariamente para que el objetivo se cumpla o no. También pueden clasificarse en **unidireccional** o **bidireccional**, dependiendo del sentido de la dependencia. La naturaleza de las interdependencias determina el tipo de coordinación.

Para construir un árbol de este tipo, los pasos son los siguientes:

1. Definir el grafo de objetivos, incluyendo la identificación y la clasificación de dependencias.
2. Asignar regiones particulares del grafo a los agentes apropiados.
3. Controlar las decisiones acerca de qué áreas del grafo explorar.
4. Recorrer el grafo.
5. Informar de los recorridos con éxito.

Algunas de las actividades serán cooperativas y otras serán realizadas por un solo agente. Esto último se decide en la fase de diseño.

La **cooperación en un SMA** consiste en la actuación coordinada entre agentes de tal manera que unos colaboran en la resolución de tareas de otros interesada o desinteresadamente. Normalmente la cooperación es entre agentes benevolentes. Los agentes cooperan de forma natural al resolver un problema global, cada uno dedicado a su parcela.

Previo a toda negociación se necesita descomponer en tareas. Esto se puede hacer de la siguiente manera:

- En el diseño.
- Mediante planificación jerárquica.
- Inherente al problema.

Luego se pueden distribuir según los criterios tratando de:

- Evitar sobrecargar recursos críticos.
- Asignar tareas dependiendo de habilidades de los agentes.
- Conseguir solapamiento en responsabilidades para lograr coherencia global.
- Asignar tareas independientes a agentes con proximidad semántica o espacial para minimizar costes de comunicación y sincronización.
- Reasignar tareas, si es necesario, para completar tareas urgentes.

Los **mecanismos de distribución de tareas** son:

- **Descomposición de Tareas**: protocolo de red de contratos.
- **Planificación multi-agente**: los agentes encargados de la planificación, los que determinan a qué agentes se asignan qué tareas.
- **Estructura organizativa**: los agentes tienen responsabilidades fijas y, por lo tanto, tienen tareas fijas asignadas.

El Protocolo de Red de Contratos, **Contract Net**, sirve para que un agente contrate tareas a otros agentes. Se deben tener en cuenta las siguientes suposiciones:

- La negociación es un proceso local que **no** implica control centralizado.
- Existe un medio bidireccional para intercambiar información.
- Cada parte en la negociación evalúa la información desde su propia perspectiva.
- El acuerdo final se alcanza mediante selección mutua.

Existen dos roles diferentes:

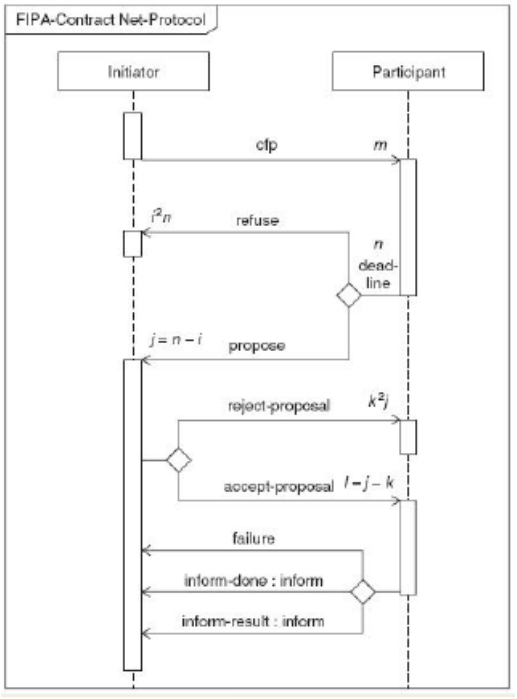
- **Manager**: inicia la negociación, monitoriza la ejecución y procesa los resultados.
- **Contractor**: se compromete con la realización, ejecuta la tarea y produce los resultados.

El **Sistema de Sensores Distribuidos** (DSS) es una red de nodos de **sensores** distribuidos por área geográfica amplia. No procesan datos, los adquieren. Hay otra red de nodos **procesadores**, que han de recibir datos de los nodos sensores. Éstos no procesan datos de los nodos sensores.

Cada manager debe elaborar una propuesta con la tarea usando un mensaje. Después, un contractor emite una oferta. Una vez que el manager ha decidido entre todas las ofertas, realiza la concesión de la realización de las tareas a aquellos nodos apropiados, enviando un mensaje.

FIPA ACL soporta los pasos del Contract:

- **cfp** (call for proposals): se usa para anunciar una tarea.
- **propose, refuse**: usados para hacer una propuesta o declinar el hacer una propuesta.
- **accept, reject**: usados para indicar la aceptación o rechazo de una propuesta.
- **inform, failure**: usados para indicar la finalización de una tarea o su fallo.



Uno de los métodos para coordinar agentes basándose en la compartición de resultados intermedios es el del Consenso:

$$x_i(k+1) = x_i(k) + \varepsilon \sum_{j \in N_i(k)} [x_j(k) - x_i(k)]$$

siendo N_i el conjunto de agentes conocidos por el agente i y:

$$\varepsilon \geq 0 \begin{cases} \varepsilon > 0 \; Conexión \; entre \; agente \; i \; y \; el \; j \\ \varepsilon = 0 \; No \; hay \; conexión \; entre \; agente \; i \; y \; el \; j \end{cases}$$

Si el grafo es uniformemente conectado para todo par de agentes, se asegura que puedan llegar a una sincronización. Si G es no dirigido, el estado final es la media aritmética.

Si se utiliza asincronía forzada, en cada iteración, cada agente selecciona un único enlace, y no todos como en el consenso, y sólo esos dos agentes se acercan entre ellos. Si el grafo está uniformemente conectado, se llega al consenso.

Para encontrar a los proveedores de servicios se usan **agentes intermediarios**, que en entornos abiertos permiten la coordinación entre agentes mediante servicios de localización y comunicación. Ponen en contacto proveedores con clientes. Existen dos tipos:

- **Broker**: realiza una función de interfaz entre los agentes que proporcionan servicios y los que los utilizan. Todas las comunicaciones pasan a través de él.
- **Matchmaker**: empareja solicitantes con proveedores, a diferencia del bróker, se limita a poner en contacto. No todas las comunicaciones pasan a través de él.