# Coopération multi-robots par insertion incrémentale de plans

Thèse soutenue au LAAS - Toulouse par

Frédéric ROBERT

# PLAN

A) Objectifs
B) Approche de la coopération multi-robots
C) Le paradigme d'insertion incrémentale de plan
D) Gestion des déplacements d'un parc de robots mobiles
E) Le système décisionnel
F) Conclusion

# A) Objectifs

- La planification multi-robots consiste à trouver des techniques permettant de construire un plan multi-robots qui soit robuste et cohérent.
  - Il s'agit de permettre la cohabitation d'un grand nombre de robots mobiles non-holonomes effectuant des tâches de déplacement.
- L'environnement des robots est dynamique et ouvert à des perturbations extérieures au système.
- Pour pallier la dynamique de l'environnement, le paradigme d' insertion incrémentale de plan consiste à :
  - Planifier de manière incrémentale (par opposition à une planification complète des missions).
  - Traiter les interactions entre les mouvements des robots dans des planifications locales (et non globales).
- Chaque agent « robot » planifie incrémentalement ses déplacements, en tenant compte des plans élaborés par les autres.

# A) Objectifs (Suite)

- Un agent maintient un PGP « Plan Global Partiel » de son plan, complété par des informations servant au contrôle de la planification distribuée.
- Un PGP contient les buts (objectifs), le plan d'action, une description des interactions avec les autres agents et l'état courant.

Les deux principaux aspects de la coopération multi-robots sont :

1) La décomposition et l'allocation des tâches.
2) La gestion des interactions et des interdépendances.

- Construction mettant en phase la planification des interactions et des interdépendances entre les robots (avec le contexte d'exécution courant) et organisant la reprise efficace des éventuels échecs d'exécution.
- Distribution de certains traitements afin de réduire la complexité des traitements élémentaires et d'améliorer ainsi les performances en terme de temps de réponse ⟶ des solutions plus élaborées et plus efficaces.

# B) Approche de la coopération multi-robots

L'architecture retenue :

- Une station centrale assure la décomposition des missions globales du système en missions élémentaires.
- Elle répartit ces missions individuelles entre les différents robots du système.
- Les robots reçoivent donc de temps en temps de nouveaux buts ou missions qu'ils réalisent de manière autonome, en coopérant avec les autre robots du système.
- Chaque agent « délibère » pour planifier en consultant les autres.
- Les délibérations nécessaires à la réalisation de ces buts et la gestion des interactions entres les robots et avec l'environnement sont menées de manière distribuée et ainsi prises en charge par les robots eux-mêmes.

# B) Approche de la coopération multi-robots(Suite)

## L'organisation générale :

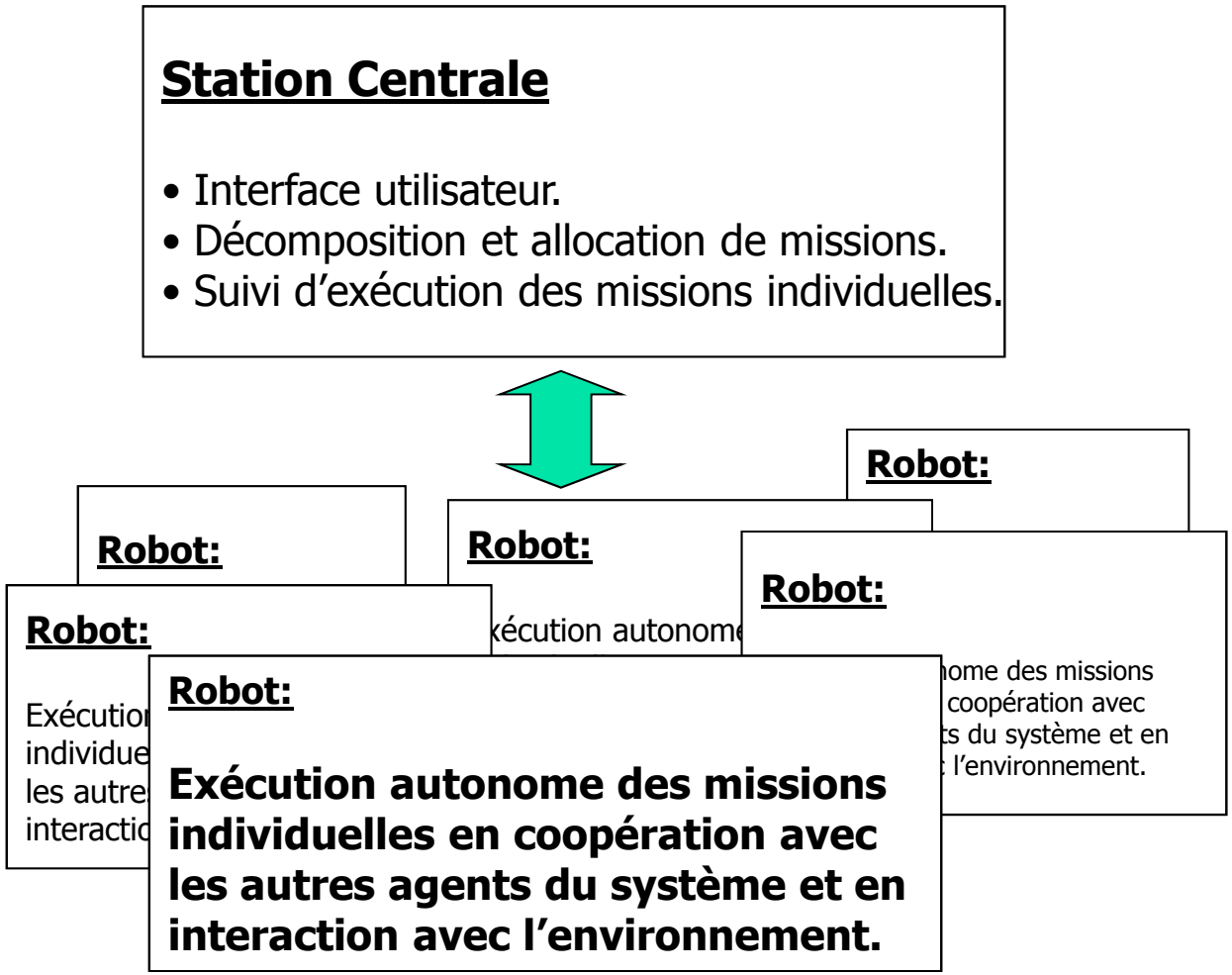
- Les mises à jour sont faites sur la base de compte-rendus d'exécution retournés par les robots.
- Tous les robots jouent un rôle équivalent et disposent des mêmes types de connaissances *a priori*.
- L'information à faire circuler concerne exclusivement le contexte d'exécution courant.

## Les communications :

Le système offre deux modes de communication

- Un mode point à point, permettant de s'adresser directement à un agent.
- Un mode diffusion, autorisant la publication d'informations à tous les agents du système.

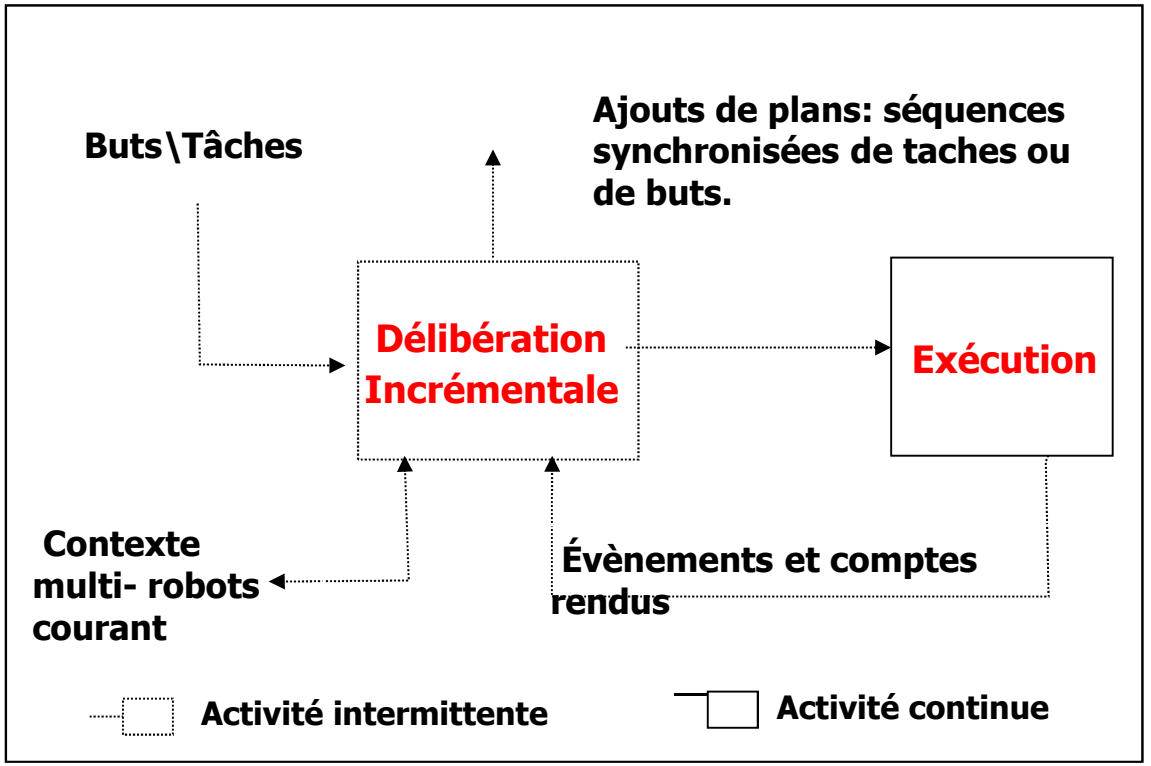
# B) Approche de la coopération multi-robots(Suite)




**Organisation générale du système**

# B) Approche de la coopération multi-robots (suite)

- Il est difficile de prévoir avec exactitude l'évolution d'une tâche.
- Dans ces conditions, aucune planification ne peut produire de plan raisonnablement robuste se projetant au delà d'un temps très court.
- Besoin d'entrelacer les délibérations et les exécutions.




**Entrelacement entre délibération et exécution**

# B) Approche de la coopération multi-robots (suite)

- Un robot planifie ses interactions avec autrui, en confrontant son plan ou son but avec les plans en cours d'exécution des autres robots.
  - Il valide son plan sans modifier celui des autres, et peut alors l'exécuter.
  - Il ne valide pas l'ensemble de son plan mais procède par ajouts successifs.
    - ce qui lui permet de ne pas trop contraindre les autres robots voulant planifier.
  - Il délibère donc de manière incrémentale.
  - Il affine progressivement les tâches à réaliser et traite les interactions avec les autres robots de manière hiérarchique.

# Activité d'un agent : Exécution

**Boucle**

**/* supervision de l'exécution du plan */**

**pour chauqe (*tâche_en_cours*)**

**contrôler l'exécution de la tâche en cours**

**/* traitement des ajouts de plan */**

**pour chauqe (*ajout_plan*)**

**concaténer *ajout_plan* au plan**

**si ∃ ( *tâches* exécutables) lancer l'exécution de *tâches***

**/* traitement des événements d'exécution internes*/**

**pour chauqe (*évenement_exécution*)**

**concaténer *ajout_plan* au plan**

**si ∃ ( *tâches* en attente de *évenement_exécution*)**

**Lancer l'exécution de *tâches***

**si ∃ ( *sujet_délibération* en attente de *évenement_exécution*)**

**Délibérer (*sujet_délibération )***

**si ∃ ( *synchronisation* en attente de *évenement_exécution*)**

**Emettre (*évenement_synchro)***

**/* traitement des événements synchronisation */**

**pour chauqe (*évenement_ synchro*)**

**si ∃ ( *tâches* en attente de *évenement_ synchro)***

**Lancer l'exécution de *tâches***

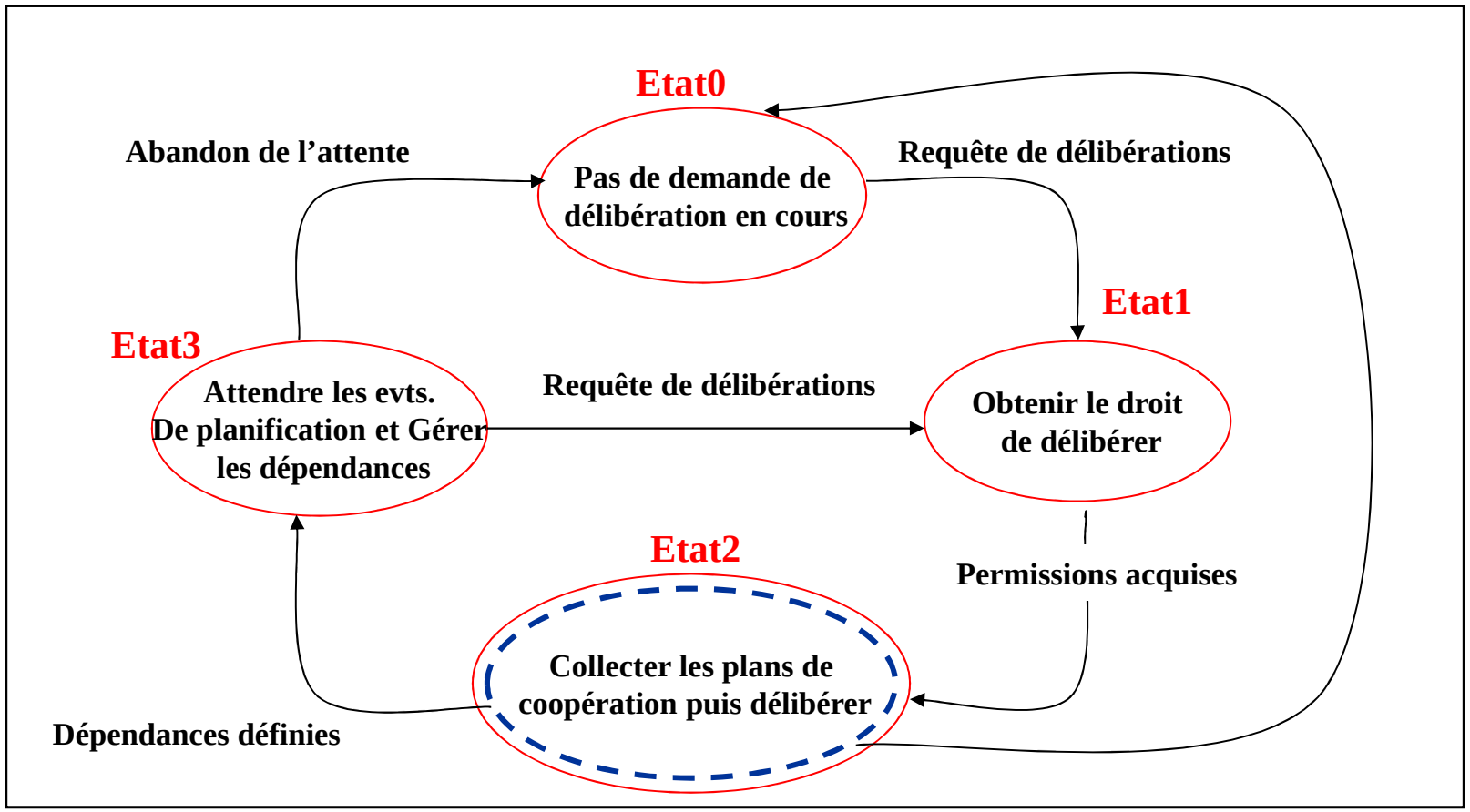
# Activité d'un agent : Délibérations

**pour chaque (*sujet_délibération*)**
- **Publier *requête*(s*ujet_délibération)***
- **Attendre le droit de délibérer**
- **Collecter les plns des autres robots**
- **Reconstituer le plan global courant**
- **Planifier**
- **si (*délibérations réussies*)**
  - **communiquer *informations_nouvelles***
  - **donner à exécuter *ajout_plan***
- **sinon**
  - **gérer *dépendances_nouvelles***
  - **attendre *informations_nouvelles***

# C) Le paradigme d'insertion incrémentale de plan

- Le paradigme est conçu pour des systèmes à buts concurrents.
- Il propose des mécanismes distribués autorisant à chacun d'élaborer ses plans en gérant les interactions avec les autres, au fur et à mesure qu'elles se présentent.
- Il organise les échanges d'information et l'exploitation de divers planificateurs.
- Il offre un cadre générique pour contrôler une délibération incrémentale et distribuée.
- Il permet de résoudre les conflits par affinement ou par proposition de tâches ou de ressources alternatives.
- Il permet de détecter les interactions ou les situations d'interblocage et garantit un contexte stable pour leurs solutions.
- Il permet de traiter de manière hiérarchique les tâches et les interactions qu'elles engendrent.
- Il permet de paralléliser le traitement des diverses interactions.

# C) Le paradigme d'insertion incrémentale de plan




Le protocole de délibération

# Formalisme de représentation

- Description du monde
  - Attributs d'état
  - Ressources
- Evolution du monde
  - Changements d'états
  - Utilisation des ressources
    - Disponibilité, consommation, production
- Tâches
- Un plan de tâches

# Description du monde : Attributs d'état

- Attribut d'état :
  - Représente une caractéristique du monde
  - A une valeur unique à un instant donné.
  - A la forme : $stat(x_1, .., x_n)$
    - *stat* : nom de l'attribut d'état
    - $(x_1, .., x_n)$ : paramètres de l'attribut
    - $D_{stat}$ : domaine des valeurs de l'attribut état

## Exemple

| Nom de l'attribut | Domaine de valeurs des paramètres | Domaine de valeur de l'attribut état |
|---|---|---|
| ***DANS*(Robot)** | $D_{robot}$**(R1, H2, etc..)** | $D_{DANS}$ **(Couloir, Salle, Couloir∧ Salle)** |

# Description du monde : Ressources

- Une ressource
  - Est un attribut particulier (coûts, quantités, etc.)
  - A un type (2 ressources sont de même type si elles peuvent être utilisées indifféremment par une même tâche)
  - A chaque type de ressource, est associé un attribut de ressource
    - *Res* : le type de ressource
    - $(x_1, .., x_n)$ : paramètres définissant une ressource dans le type.

    Ce sont des variables ayant comme domaines $(D_1, .., D_n)$
    - $D_{res}$ : domaine des valeurs possibles de la ressource *res*

# Exemples de Ressource

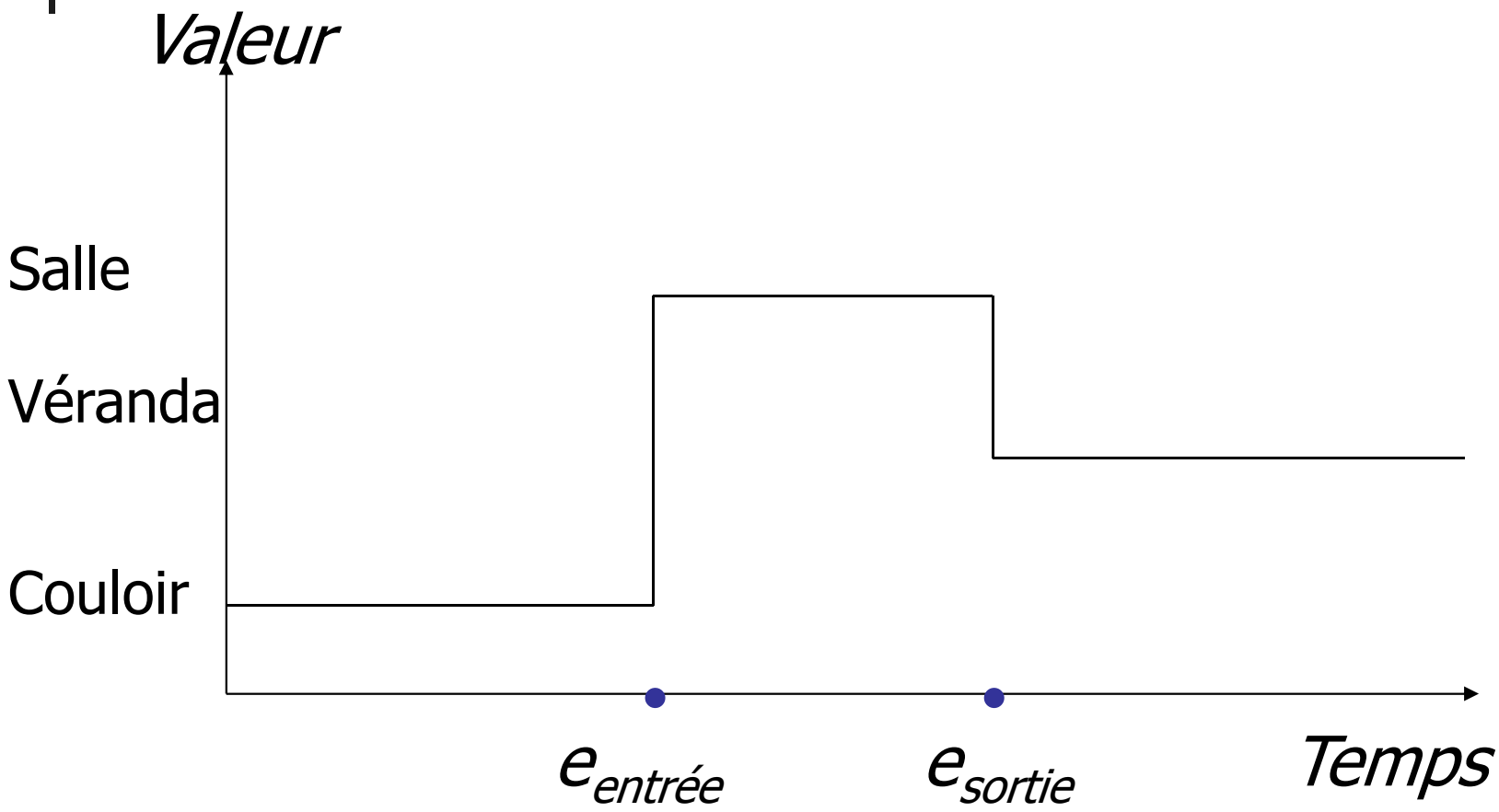
- Continue : énergie
- Discrète : étau
- Banalisée : une pile de disques
- Individualisée : le couloir de longueur 5 m
- Divisible : le robot prends 5 disques parmi 20 disponibles
- Indivisible : le robot utilise le marteau

# Evolution du monde : changements d'états

- 2 prédicats
  - Persistance d'un attribut $stat(x_1, .., x_n)$ à une valeur $V$ sur un intervalle temporel $]e_{deb}, e_{fin}[$
    - $hold(stat(x_1, .., x_n) : V (e_{deb}, e_{fin}))$
    - $hold(stat(x_1, .., x_n) : V (e_{deb}, ?))$
  - Changement d'état suite à un évenement $e_{changement}$
    - $change (stat(x_1, .., x_n) : (V_{initiale}, V_{finale}), e_{changement})$

$\exists\ e_{?1} \exists\ e_{?2} / \ e_{?1} < e_{changement} < e_{?2}$
$\wedge\ hold(stat(x_1, .., x_n) : V_{initiale} (e_{?1}, e_{changement}))$
$\wedge\ hold(stat(x_1, .., x_n) : V_{finale} (e_{changement}, e_{?2}))$

# Exemple : DANS(R2)



# Ressources

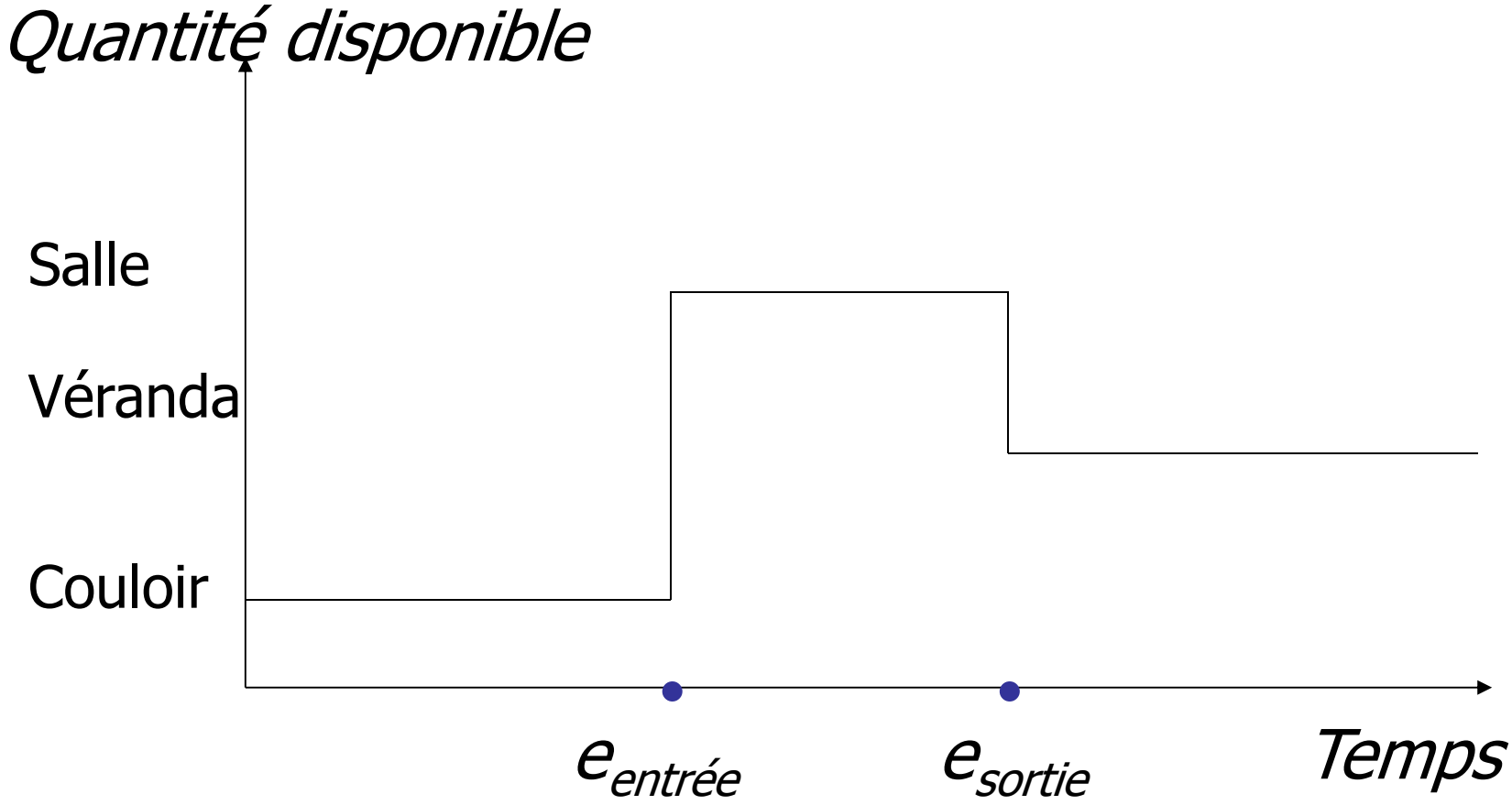
- Disponibilité
  - *available* (res$(x_1, .., x_n)$ : $q_{disponible}$, ($e_{deb}$ , $e_{fin}$))
- Consommation
  - *consume* (res$(x_1, .., x_n)$ : $q_{consomée}$ , $e_{conso}$))

  $\exists\, e_{?1} \exists\, e_{?2} \,/\, e_{?1} < e_{conso} < e_{?2}$

  $\wedge$ *availbale*$(res(x_1, .., x_n) : q_{disponible}\,(e_{?1}, e_{conso}))$

  $\wedge$ *available*$(res(x_1, .., x_n) : q_{disponible} - q_{consommée}(e_{conso}, e_{?2}))$
- Production
  - *produce* (res$(x_1, .., x_n)$ : $q_{produite}$ , $e_{prod}$))

# Exemple




*consume(Salle : 1, $e_{entrée}$) et produce(Salle : 1, $e_{sortie}$)*

# Tâches

- Action : tâche à affiner progressivement
- Description hiérarchique
- Une tâche :
  - Un événement début de tâche $e_{deb}$
  - Un événement fin de tâche $e_{fin}$
  - Un ensemble de sous-tâches $SubTasks\{t_i\}$
  - Des conditions d'insertion *InsertCond*

# Tâches (suite)

- InsertCond =

  $\{hold(stat(x_1, .., x_n) : V, (e_{?1}, e_{deb}) / e_{?1} < e_{deb})\}$
  $\cup \{hold(stat'(x'_1, .., x'_n) : V', (e_{?1}, e_i) / e_{?1} < e_{deb} \wedge e_{deb} < e_i < e_{fin})\}$
  $\cup \{available(res'(y_1, .., y_n) : q, (e_{?2}, e_{deb}) / e_{?2} < e_{deb})\}$
  $\cup \{hold(res(y'_1, .., y'_n) : q', (e_{?3}, e_i) / e_{?3} < e_{deb} \wedge e_{deb} < e_j < e_{fin})\}$

- Events = Changes $\cup$ Consumes $\cup$ Produces =

  $\{change(stat(x_1, .., x_n) : (V_i, V_f), e_{chgt}) / e_{deb} < e_{chgt} < e_{fin}\}$
  $\cup \{consume(res(y_1, .., y_n) : q_{consommée}, e_{conso}) / e_{deb} < e_{conso} < e_{fin}\}$
  $\cup \{produce(res'(y'_1, .., y'_n) : q_{produite}, e_{prod}) / e_{deb}) < e_{prod} < e_{fin}\}$

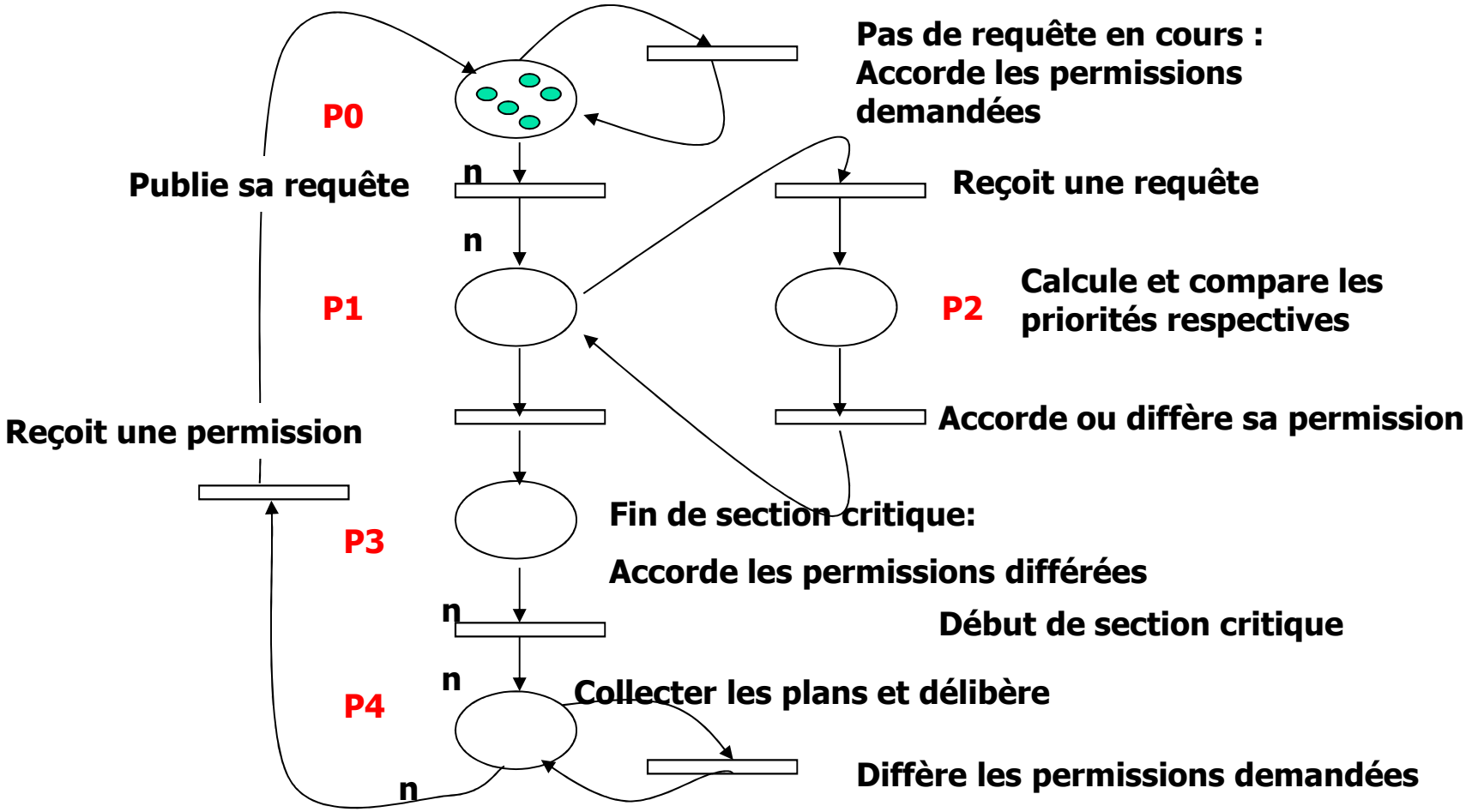
- Un ensemble de contraintes de précédence temporelle définissant un ordre partiel sur les différents événements
  - $Constraints = \{c_{ij}\}$ avec $c_{ij} \equiv e_i < e_j$

# C) Le paradigme d'insertion incrémentale de plan

Le droit de délibérer :

- Il faut garantir qu'a un instant donné, un seul robot effectue cette opération (Exclusion mutuelle).
- L'exclusion mutuelle est garantie par l'obtention du nombre exacte de permissions.
    - La permission de chaque robot est représentée par un jeton dans un réseau de Petri.
    - Les **n** jetons présents dans le réseau correspondent au nombres de robots dans le système.
- L'absence de famine est assurée par le maintien d'un ordre total sur les requêtes.
- Le déplacement d'un jeton est basé sur un algorithme distribué d'exclusion mutuelle.
- Il permet de collecter l'ensemble des permissions sans risque d'interblocage.

# C) Le paradigme d'insertion incrémentale de plan




**Obtention du droit de délibérer**

# C) Le paradigme d'insertion incrémentale de plan

Les délibérations :

- S'effectuent en section critique garantissant la cohérence des informations réunies.
- L'action de délibérer signifiera pour les robots :

1. Faire la synthèse d'un ajout de plan :

   *faite hors section critique, i.e : sans tenir compte du contexte d'exécution courant.*

2. Valider l'ajout de plan en prenant en compte les plans des autres:

   *chaque ajout de plan sera ensuite validé en tenant compte des plans exécutés par les autres robots.*

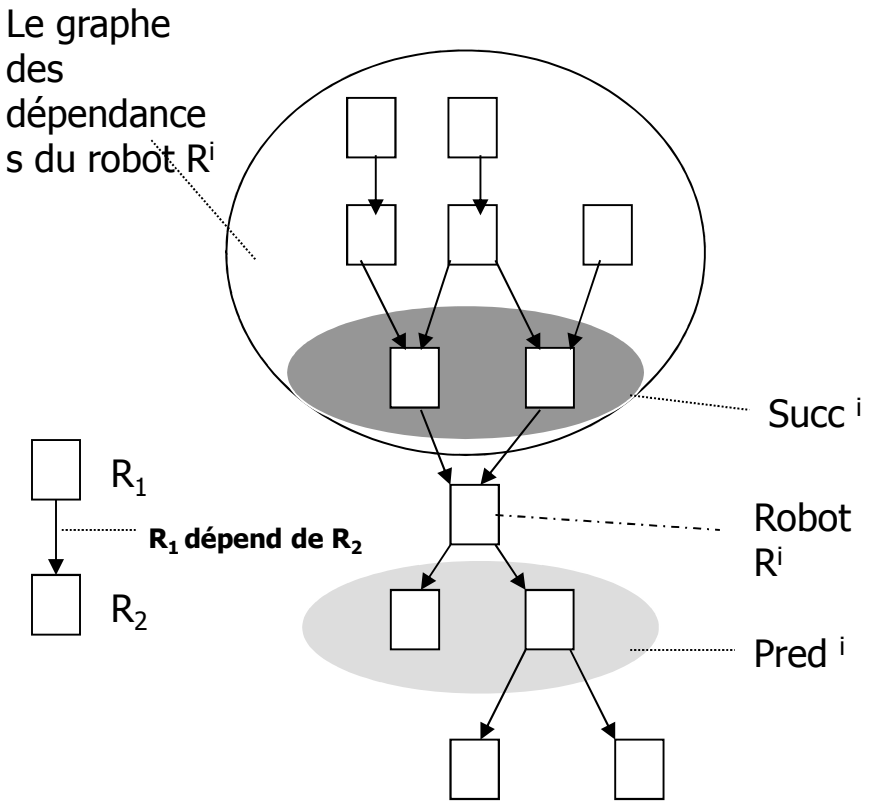
- Il est avantageux de prendre en compte les plans des autres robots, dès la phase de synthèse du plan.

# C) Le paradigme d'insertion incrémentale de plan

## La gestion des dépendances

- Afin de valider le principe général de l'insertion incrémentale de plans et de le rendre fiable et robuste, nous devons fournir aux robots les moyens de détecter d'éventuels interblocages dans les attentes d'événements de planification.

- Pour cela, chaque robot $R^i$ maintient, en permanence, un graphe $D_g^i$ des robots dépendants de lui en terme de planification.




**Les dépendances du robot $R^i$ et**

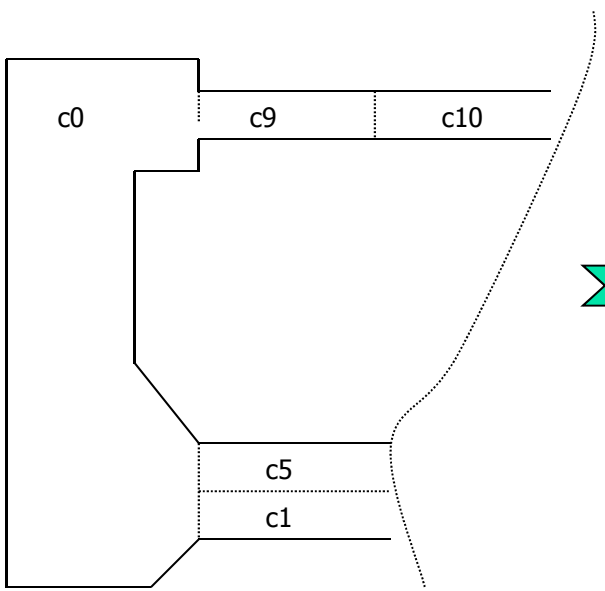
**Les dépendances sur le robot $R^i$**

# D) Gestion des déplacements d'un parc de robots mobiles

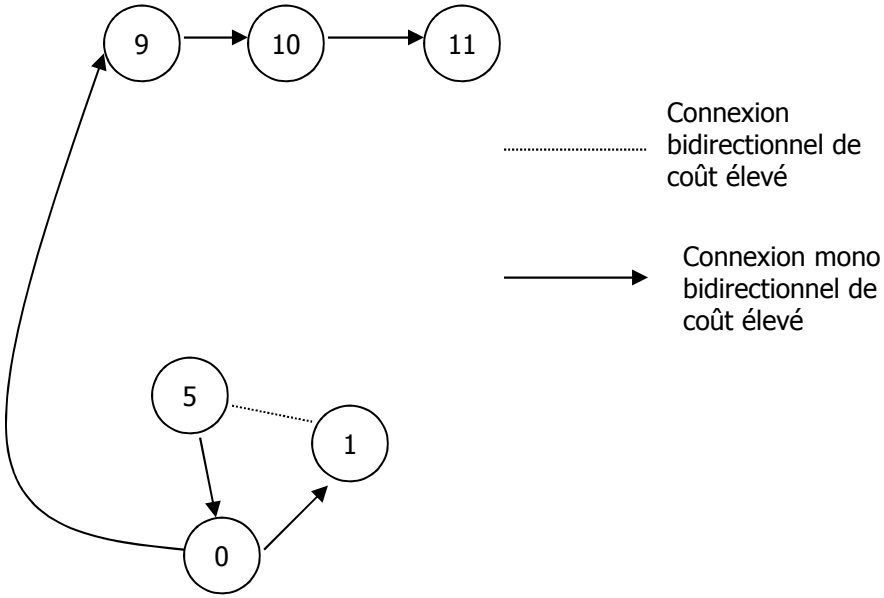
- Les conflits autour de la ressource « espace » constituent la principale source des interactions générées par les tâches de déplacement.
- Un robot ayant une position but à atteindre, doit planifier son mouvement et en synchroniser l'exécution avec les autres robots.
- Nous sommes ici exactement dans le domaine des systèmes à buts concurrents pour lesquels le paradigme d'insertion incrémentale de plan a été conçu.
- Deux modes de coopération sont organisés entre eux comme deux niveaux d'affinement de la tâche de déplacement :
    1. Un niveau cellule
    2. Un niveau trajectoire
- Les robots coopèrent d'abord au niveau cellule et en cas de nécessité ils affinent leurs synchronisations en coopérant au niveau des trajectoires.

# D) Gestion des déplacements d'un parc de robots mobiles (suite)

- Chaque robot est doté d'un planificateur de trajectoires.
- Un graphe topologique, indiquant les connections entre les cellules est fourni.




**Décomposition d'un environnement en cellules**




**Organisation des cellules en un graphe topologique**

# D) Gestion des déplacements d'un parc de robots mobiles (suite)

Un robot voulant se déplacer dans son environnement :

- cherche un chemin vers son but dans le graphe topologique des cellules.
- planifie ensuite sa trajectoire et détermine avec précision les cellules utilisées dans son mouvement.
- choisit la portée de l'ajout de plan en fonction du type d'entité impliquée et publie une requête de délibération portant sur les cellules utilisées dans l'ajout de plan.
- collecte les plans de coopération nécessaires, une fois rassemblé l'ensemble des autorisations requises.
- Il peut alors reconstituer un « plan global local » et tenter d'y insérer son ajout de plan.

# E) Le système décisionnel

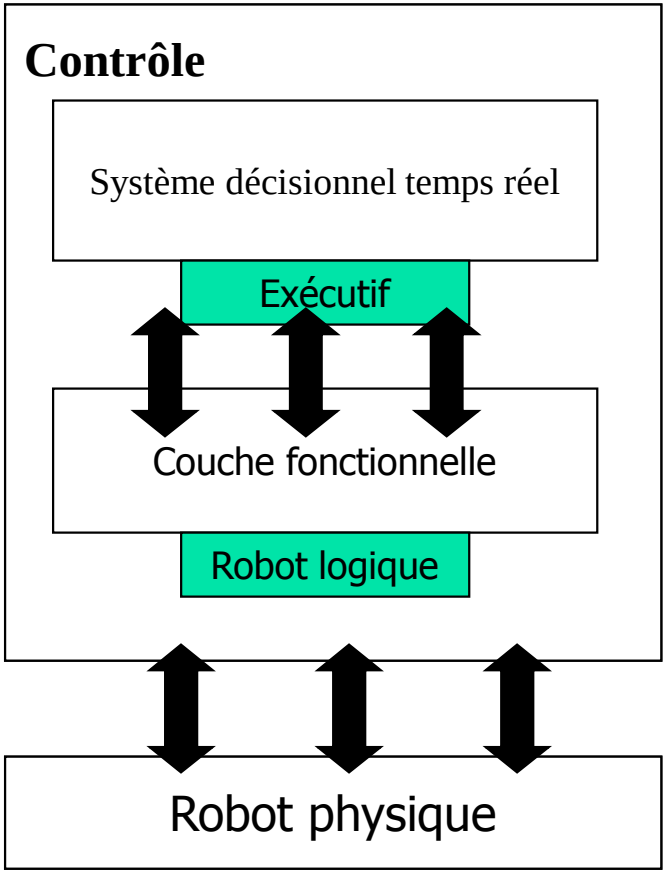
## 1) l'architecture de contrôle d'un robot

- Un système décisionnel « temps réel », en charge des aspects délibération et contrôle d'exécution.
- Une couche fonctionnelle, constituée d'un substrat de modules intégrant et organisant les fonctions élémentaires du robot.
- Une interface entre les deux est assurée par un exécutif.

### Les rôle du système décisionnel

Le robot doit être capable:

- de délibérer
- de superviser l'exécution des plans



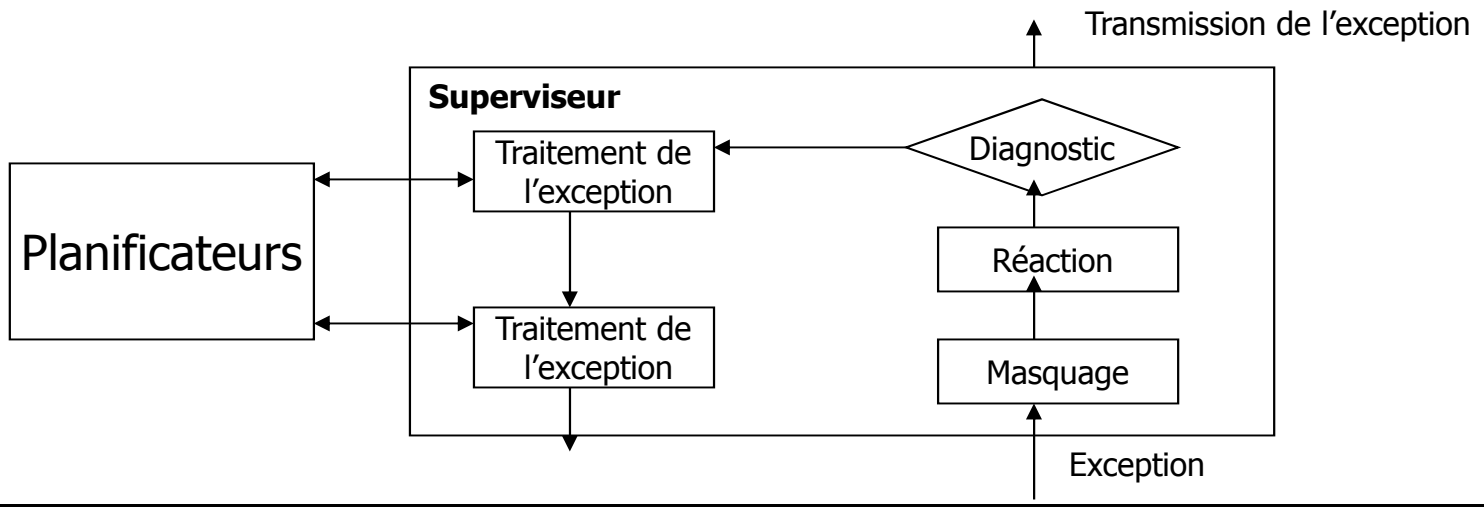

**L'architecture de contrôle d'un robot**

# E) Le système décisionnel (suite)

## 2) Un système décisionnel coopératif

- L 'affinement d'une mission ou d'un plan de tâches consiste à mener les délibérations nécessaires pour les traduire en des plans de tâches exécutables.
- Le contrôle de l'exécution du plan consiste à donner les tâches du plan exécutable à réaliser, en respectant le séquencement établi lors des délibérations.

## 3) Prise en compte des événements exceptionnels



# F) Conclusion

- L'intérêt de cette approche est d'aboutir à un système continuellement capable d'accepter de nouvelles mission, d'intégrer l'arrivée de nouveaux robots ou le départ d'autres.
- Grâce au paradigme d'insertion incrémentale de plans, un grand nombre de robots autonomes peuvent planifier et exécuter différents types de tâches en partageant le même environnement et les mêmes ressources.
- L'insertion incrémentale de plans permet aux robots d'affiner progressivement leurs interactions et d'en planifier les détails au fur et à mesure de l'avancement des tâches.
- L'approche adoptée présente l'intérêt de permettre une coopération par adaptation incrémentale et locale des plans, tout en garantissant le traitement centralisé du problème global.
- Cette méthode est robuste, car elle permet d'éviter les interblocages au niveau de l'exécution.