IFT608 / IFT702
Intelligence Artificielle

Contrôle de la recherche avec des formules LTL

Sujets

- Logique Temporelle Linéaire (LTL)
- Spécification des buts avec LTL
- Spécifications de règles de contrôle de recherche avec LTL.
- TLPLAN

Références

- Bacchus F. and Kabanza F. Using Temporal Logic to Express Search Control Knowledge for Planning. *Artificial Intelligence* , 116(1-2):123-191, 2000.

- Malik Ghallab, Dana Nau & Paolo Traverso (2016). *Automated Planning and Acting.* http://projects.laas.fr/planning/book.pdf

- Malik Ghallab, Dana Nau & Paolo Traverso (2004). *Automated Planning : Theory and Practice (Chapitre 10)* https://www.cs.umd.edu/~nau/planning/slides/

- http://www.cs.toronto.edu/tlplan/

RAPPEL

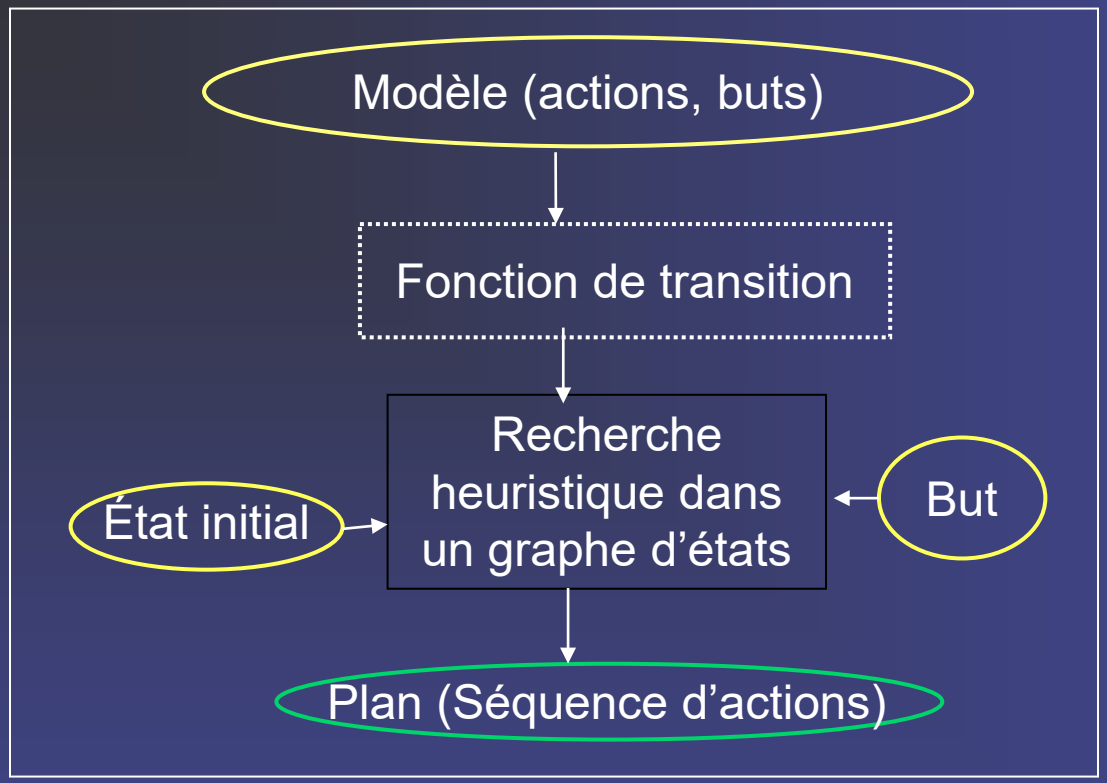
Méthodes pour choisir les actions

Il y a trois approches algorithmiques principales en IA pour développer agent capable de choisir ses actions :

1. **Programmer** un plan (contrôleur) – Donne la capacité d'avoir des comportements automatiques, mais pas forcément autonome.
2. **Apprendre un plan (contrôleur)** à partir des données.
3. **Générer** un plan (contrôleur) à partir d'un modèle d'actions primitives

Dans la pratique courante, ces approches sont complémentaires.

Architecture générale d'un planificateur opérant par recherche dans un espace d'états

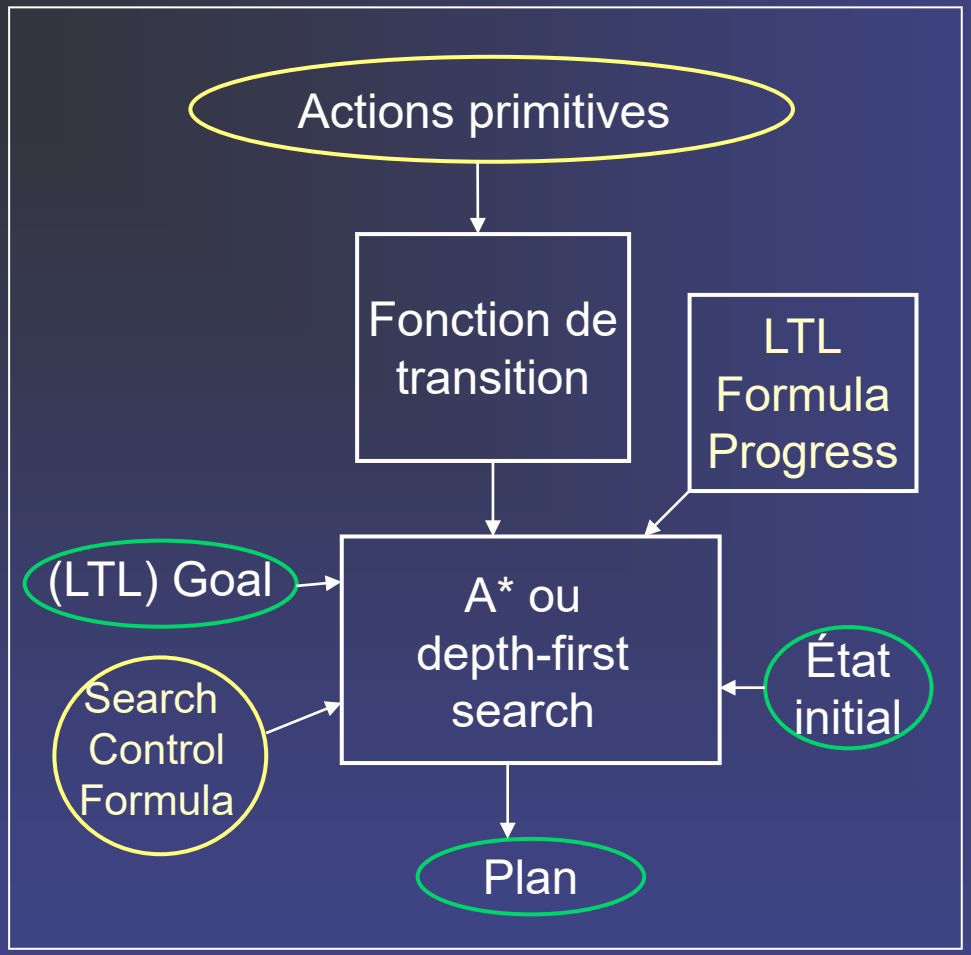


FIN DE RAPPEL

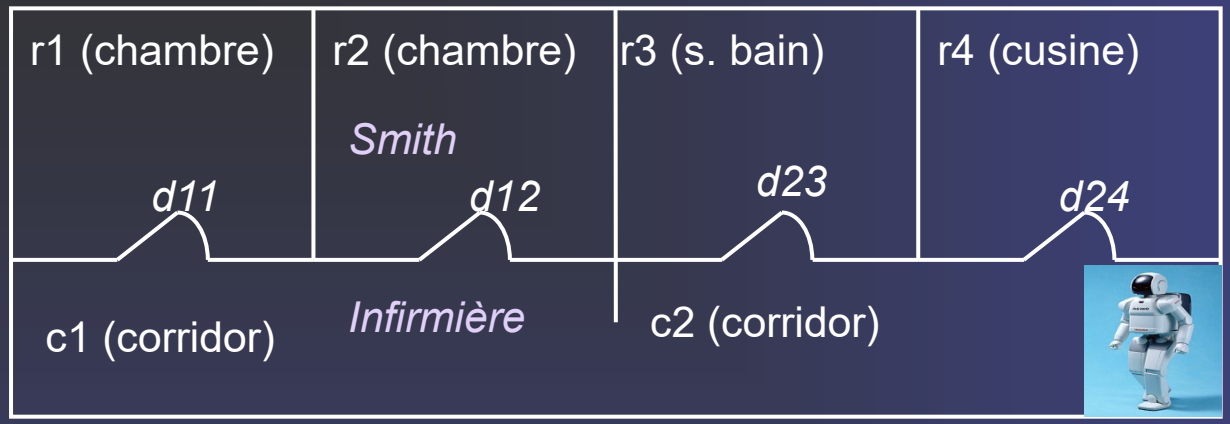
MOTIVATION

EXPLORATION DE L'ESPACE D'ÉTATS AVEC DES CONNAISSANCES STRATÉGIQUES TEMPORELLES

Architecture du planificateur



Robot domestique



- Des robots vont bientôt devenir omniprésents:
  - Transporter des objets pour eux (ex.: verre d'eau)
  - Chercher des objets (ex., lunettes)
  - Surveiller des activités (ex., les portes sont fermées)
- Le Challenge AAAI simule des tâches similaires à ces activités.

Types de tâches

- Certaines tâches sont des buts classiques (atteindre un état final) :
  - Transporter des objets à des endroits désignés.

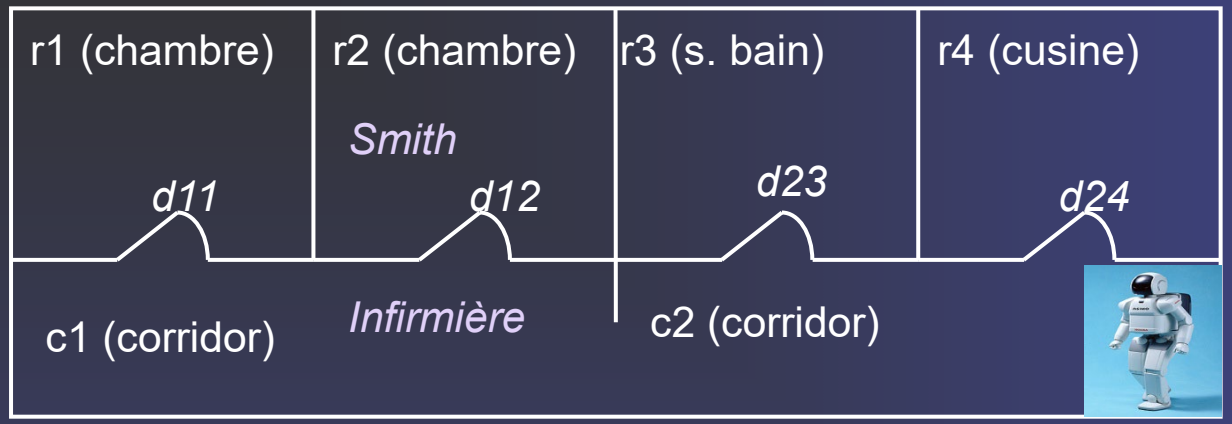
- D'autres tâches sont comportementales :
  - Conduire Madame Smith à la cuisine, lui servir un verre d'eau, et si vous rencontrez l'infermière, donnez-lui ce message.
  - Chaque fois que la cuisine est salie, la nettoyer, et une fois fini, préparer le prochain repas.

- Ce genre de buts sont dits « temporels » :
  - Maintenir certaines conditions vraies.
  - Accomplir une tâche périodiquement.
  - Accomplir une tâche dans un délai prescrit suite à une requête.
  - Accomplir des tâches dans une séquence donnée.
  - Combinaisons des buts précédents.

Connaissance experte de planification

- Un planificateur classique a comme connaissance les actions primitives ou les transitions possibles (effets, préconditions)
- Certains planificateurs utilisent en plus des connaissances expertes pour planifier.
  - Ce sont des connaissances sur les stratégies de planification.
  - Par exemple: **chaque fois** que le but est d'*avoir un objet quelconque à un endroit quelconque* et que l'état actuel durant la recherche est tel que *le robot tient l'objet mais l'objet n'est pas à sa destination* **alors ne pas** faire des actions qui auraient pour effet de *relâcher l'objet*.
    - Durant la planification, éviter d'explorer des comportements inutiles selon les connaissances données par un expert
- Ça s'appelle aussi des **connaissances de contrôle de recherche**

Règles de contrôle de recherche



- Exemple :
  - Lorsque le but exige d'amener un objet *x* à la chambre *y*, alors le planificateur s'il atteint un état tel que le robot a l'objet *x* dans la main, il (le planificateur) doit préserver cette condition jusqu'à ce qu'à un nœud tel que le robot est rendu dans *y*.
- Une telle règle permet d'éviter la génération de transitions inutiles durant la recherche d'un plan.

D'où viennent les connaissances stratégiques?

- Les connaissances de contrôle de recherche
  - Rendent le planificateur plus efficace
  - Et le font paraître plus intelligents (les personnes évitent certaines alternatives lorsqu'ils cogitent un plan, grâce à leur expérience/connaissances).

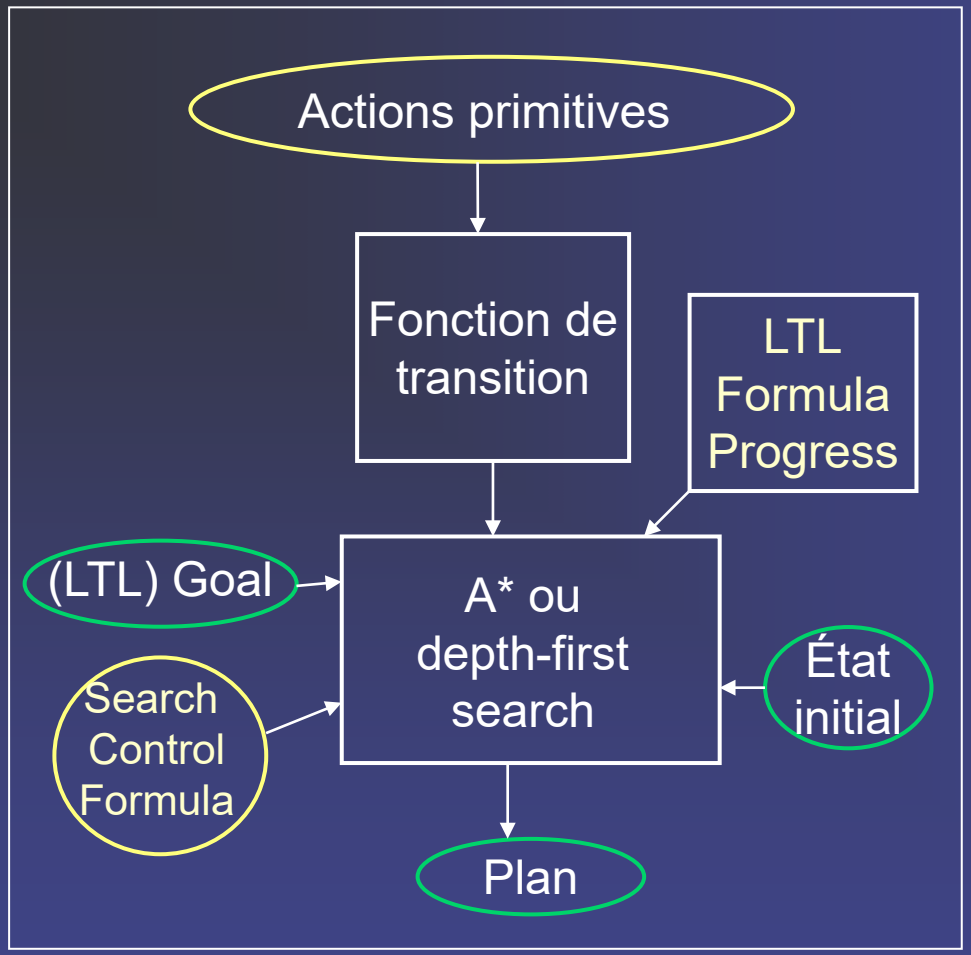
- Les connaissances stratégiques sont données par le programmeur
  … au même titre que les actions primitives.

- Certains chercheurs travaillent à développer des algorithmes capables d'apprendre de tels connaissances.
  … D'autres visent à apprendre les actions primitives.

Illustration : TLPLAN

- TLPLAN est un planificateur basé sur une recherche dans un espace d'états comme A*:
  - Utilise un langage à base de règles pour spécifier les transitions (actions primitives)
  - Utilise la Logique temporelle linéaire (LTL) pour spécifier des buts comportementales.
  - Utilise LTL pour spécifier des connaissances de contrôle de recherche.

Architecture de TLPLAN



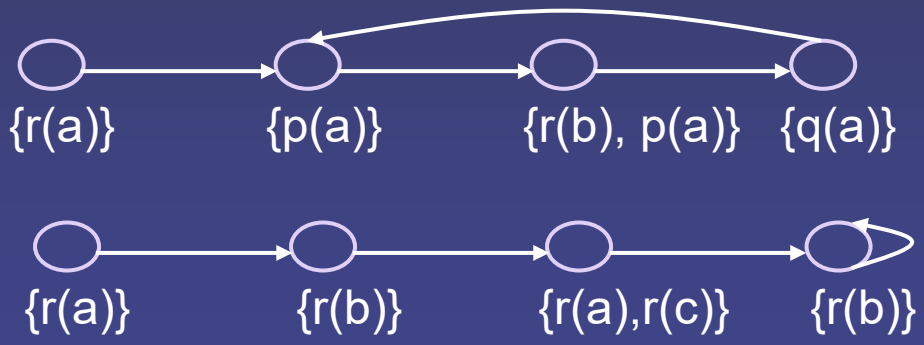
LTL

- LTL :
  - Logique du premier ordre +
  - Opérateurs temporels : O (*next)*, <> (eventually), [] (*always), U (until)*

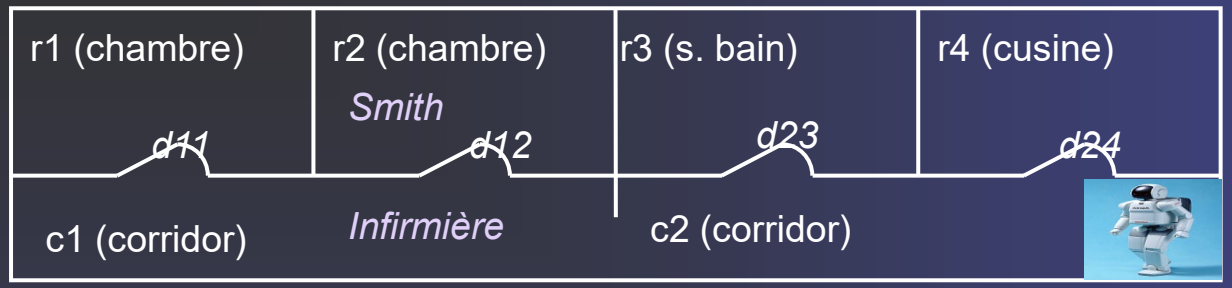
- Les opérateurs sont appliquées sur des formules, permettant leur interprétation sur des séquences d'états (générés parr A* ou depth-first search)

- Exemple: "toujours *p(x)* implique éventuellement *q(x)* et vice-versa"

$\forall$ x dom(x) [] ((p (x) -> <> q (x)) $\wedge$ [] (q (x) -> <> p (x)))



Exemple : robot domestique



- TLPLAN utilise LTL pour spécifier des buts du genre:
  - Chaque fois que la cuisine est salie, la nettoyer, et une fois fini, préparer le prochain repas.
- Et des connaissances stratégiques du genre:
  - Lorsque le but exige d'amener un objet *x* à la chambre *y*, le robot, s'il réussit à saisir l'objet *x* doit le garder jusqu'à ce qu'il soit rendu dans *y*.

Progression des formules LTL

- Cette technique permet de :
  - Vérifier une formule LTL sur des chemins générés par A* ou *depth-first* de manière incrémentales.
  - Dans le nœud courant, on évalue la contrainte sur l'état courant et on retarde la contraintes sur les états futurs :
    - *Cela est possible parce qu'on peut décomposer chaque formule en partie présente et partie future.*
- La partie présente évalue à FALSE dans des états terminants des préfixes ne satisfaisant pas la formule LTL.

Algorithme Formula Progression

- Entrée : - État
  - Formule LTL
- Sortie : Formule LTL retardée au successeurs de l'état
- Trois sorties possibles :
  - True : le chemin satisfait la formule
  - False : le chemin viole la formule
  - Une formule LTL avec au moins un connecteur temporel

The progression algorithm

Inputs: An $\mathcal{LT}$ formula f and a world w.

Output: A new $\mathcal{LT}$ formula f^+ representing the progression of f through the world w.

Algorithm *Progress*(f, w)

Case

(1) $f = \phi \in \mathcal{L}$ (i.e., ϕ contains no temporal modalities):
$f^+ :=$ TRUE if $w \models f$, FALSE otherwise.

(2) $f = f_1 \wedge f_2$: $f^+ := Progress(f_1, w) \wedge Progress(f_2, w)$

(3) $f = \neg f_1$: $f^+ := \neg Progress(f_1, w)$

(4) $f = \bigcirc f_1$: $f^+ := f_1$

(5) $f = f_1 \mathsf{U} f_2$: $f^+ := Progress(f_2, w) \vee (Progress(f_1, w) \wedge f)$

(6) $f = \Diamond f_1$: $f^+ := Progress(f_1, w) \vee f$

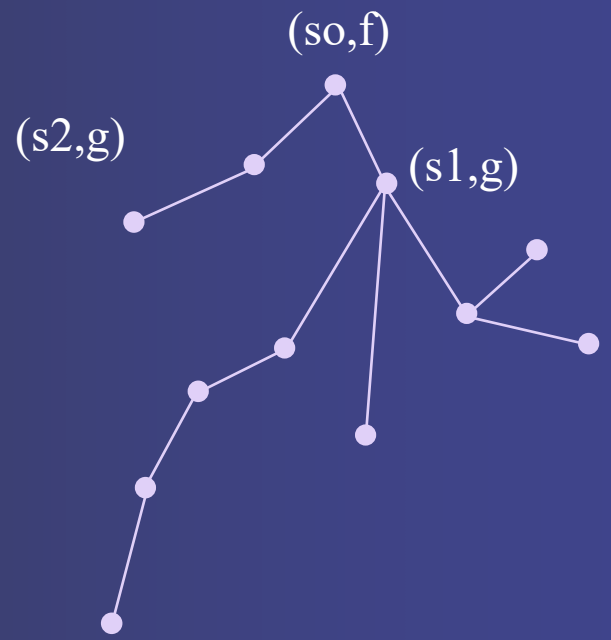
(7) $f = \Box f_1$: $f^+ := Progress(f_1, w) \wedge f$

(8) $f = \forall[x{:}\gamma(x)]f_1$: $f^+ := \bigwedge_{\{c:w\models\gamma(x/c)\}} Progress(f_1(x/c), w)$

(9) $f = \exists[x{:}\gamma(x)]f_1$: $f^+ := \bigvee_{\{c:w\models\gamma(x/c)\}} Progress(f_1(x/c), w)$

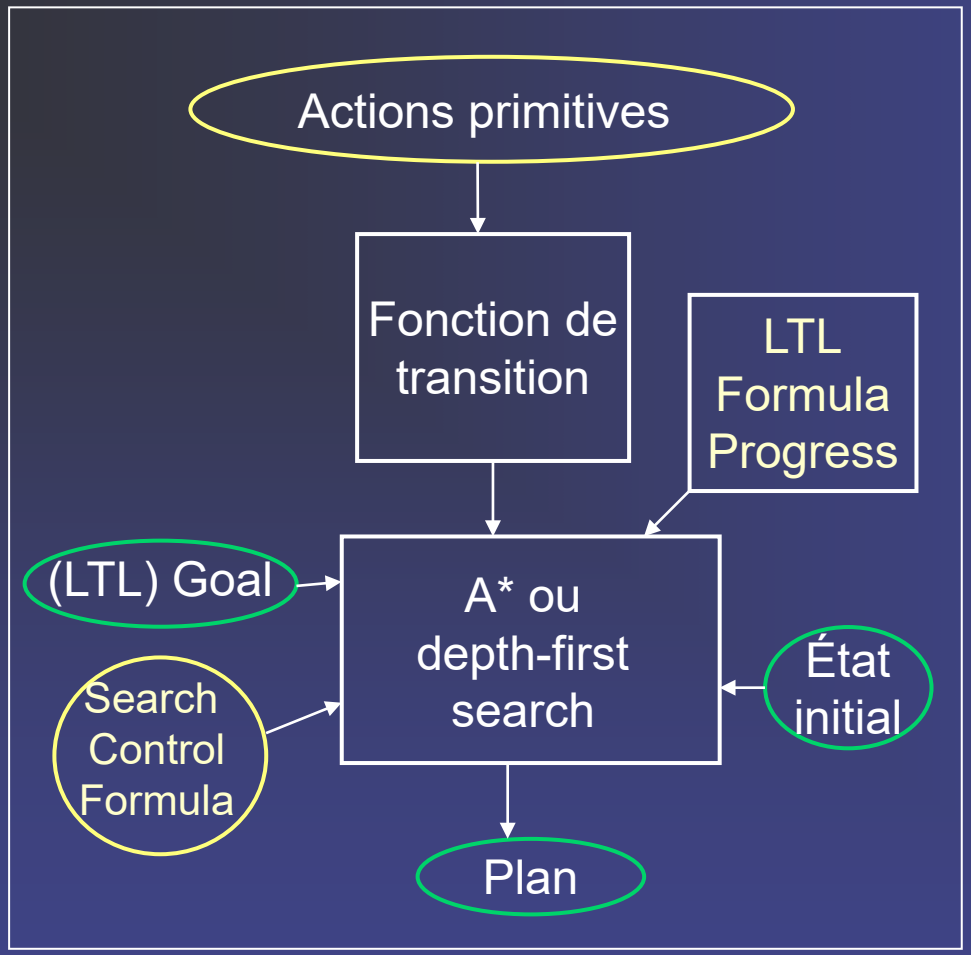
Progression LTL combinée avec A*

- L'état initial (s0) est étiqueté avec la formule LTL initiale (f).
- Chaque successeur est étiquété avec le résultat de la progression de la formule dans l'état courant.
- Les états étiquetés FALSE sont considérés comme des cul-de-sac.
  - On ne génère pas de successeurs pour eux.

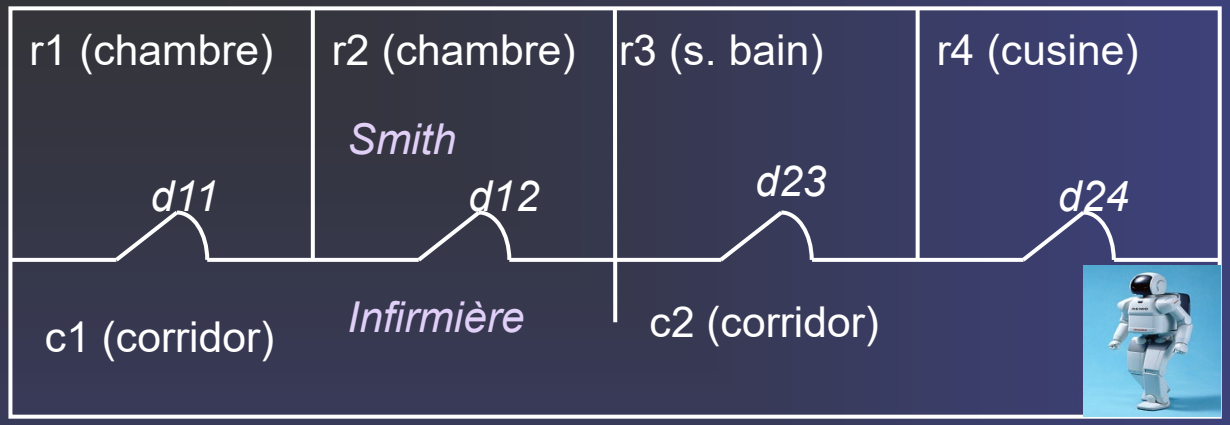


g=prog(f,s0)

Architecture de TLPLAN

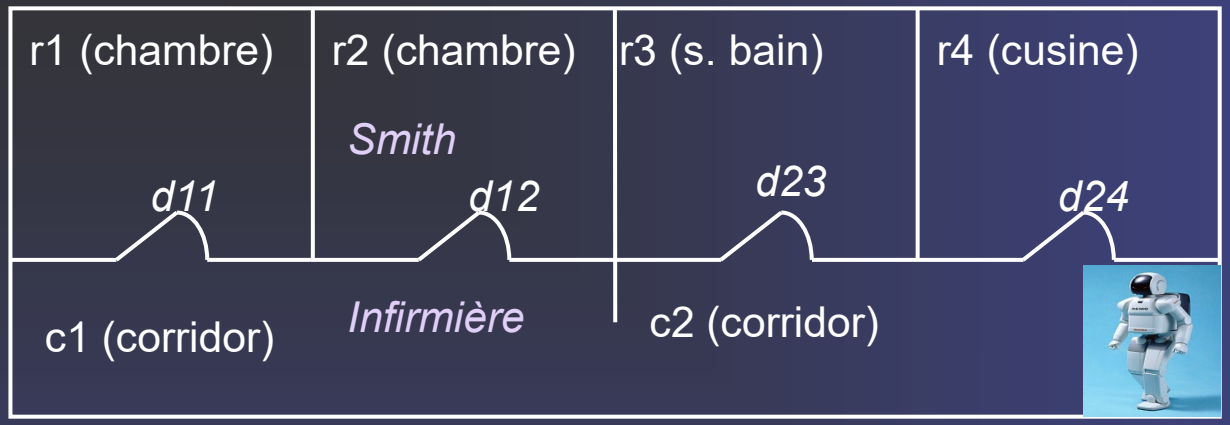


Exemple



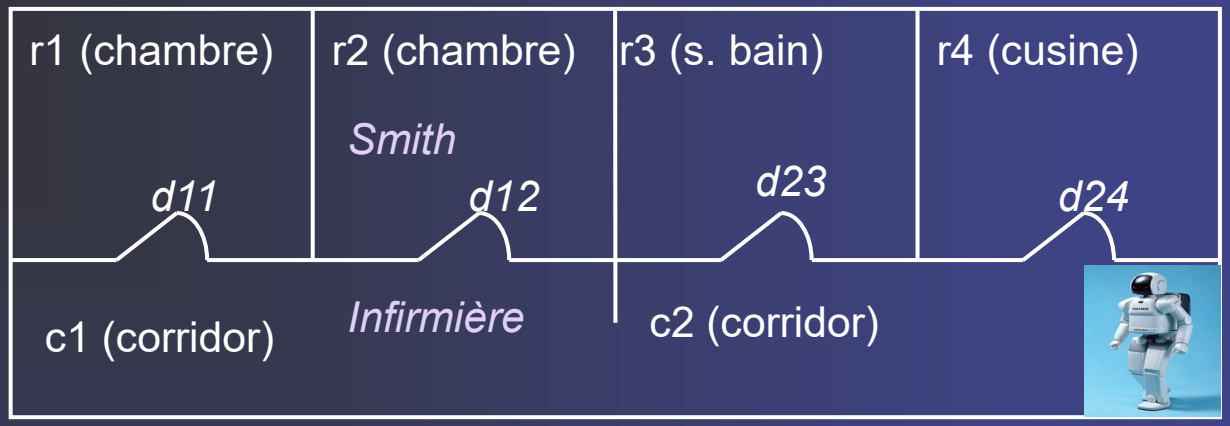
- But séquentiel :
  - <>(in(o1,r2) Λ O<>(in(o2,r4) Λ O<>(in(o4,r2))

Exemple :



- But réactif :
  - Conduire Madame Smith à la cuisine, lui servir un verre d'eau, et si vous rencontrez l'infermière, donnez-lui ce message.
  - (in(robot,c1) Λ in(nurse,c1) -> O talkto(nurse)) U
    ([] with(Smith) Λ O<> (in(Smith,bathroom) Λ O<> have(Smith, cofee)))

Exemple :



- But cyclique :
  - Le robot doit continuellement surveiller (visiter) r1 et r3
  - [] (in(robot, r1) -> <> in(robot, r3)) Λ
    [] (in(robot, r3) -> <> in(robot, r1))
  - Plan: ((close(d11), open(d11), mv(r1,c1), mv(c1,c2), mv(c2,r3),
    close(d23), open(d23), mv(r3,c2), mv(c2,c1), mv(c1,r1),0)

Au de là de TLPLAN

- TALPLANNER
  - Kvarnström and Doherty (Linköping University, Sweden)
- Préférences temporellement étendues
  - McIlRaith and Baier (University of Toronto)
- Défi: apprendre les connaissances de contrôle de recherche.

Références

- Malik Ghallab, Dana Nau & Paolo Traverso. *Automated Planning: theory and practice.* Morgan Kaufmann, 2004. http://www.laas.fr/planning/ (Chapitre 10)

- Bacchus F. and Kabanza F. Using Temporal Logic to Express Search Control Knowledge for Planning. *Artificial Intelligence* , 116(1-2):123-191, 2000

- http://www.cs.toronto.edu/tlplan/tlplan.shtml

TLPLAN

- Exemples de modèles
  - Robot world
  - Robot problems

Vers l'apprentissage automatique?

- Hann et al. (ICLR, 2021). Teaching Temporal Logic to Neural Networks. https://arxiv.org/abs/2003.04218