Planification avec préférences

Ce chapitre s'intéresse au problème de la *planification* qui consiste à élaborer des plans d'action afin d'atteindre des objectifs préalablement définis. Il se focalise principalement sur la problématique de la *planification avec préférences* dont le but est de produire des plans qui maximisent la satisfaction des décideurs. En conséquence, ce chapitre propose une introduction à la planification (cf. section 1.1) ainsi qu'un état de l'art du domaine de la planification avec préférences (cf. section 1.2).

Les deux parties qui composent ce chapitre sont construites symétriquement de sorte à se répondre l'une à l'autre. Ainsi, la section 1.1.1 présente les principaux concepts de la planification tandis que la section 1.2.1 traite de la notion de préférence. De même, les sections 1.1.2 et 1.2.2 introduisent respectivement le problème de la planification classique et celui de la planification avec préférences. Les sections 1.1.3 et 1.2.3 s'intéressent quant à elles aux langages formels utilisés pour représenter et résoudre les problèmes de planification. Finalement, la section 1.2.4 présente les principaux algorithmes de planification avec préférences de l'art.

1.1 De la problématique de la planification

La planification automatisée est un domaine de l'Intelligence Artificielle qui consiste à choisir et séquencer un ensemble d'actions par anticipation de leurs résultats afin d'atteindre des objectifs fixés [66]. Si cette tâche est souvent triviale pour un certain nombre de problèmes du quotidien, elle demeure néanmoins complexe lorsque les problèmes considérés sont fortement combinatoires. Cette section introduit le modèle conceptuel de la planification (cf. section 1.1.1) ainsi que le problème dit de *planification classique* (cf. section 1.1.2). De plus, le langage formel utilisé pour spécifier et résoudre les problèmes de planification est également présenté (cf. section 1.1.3).

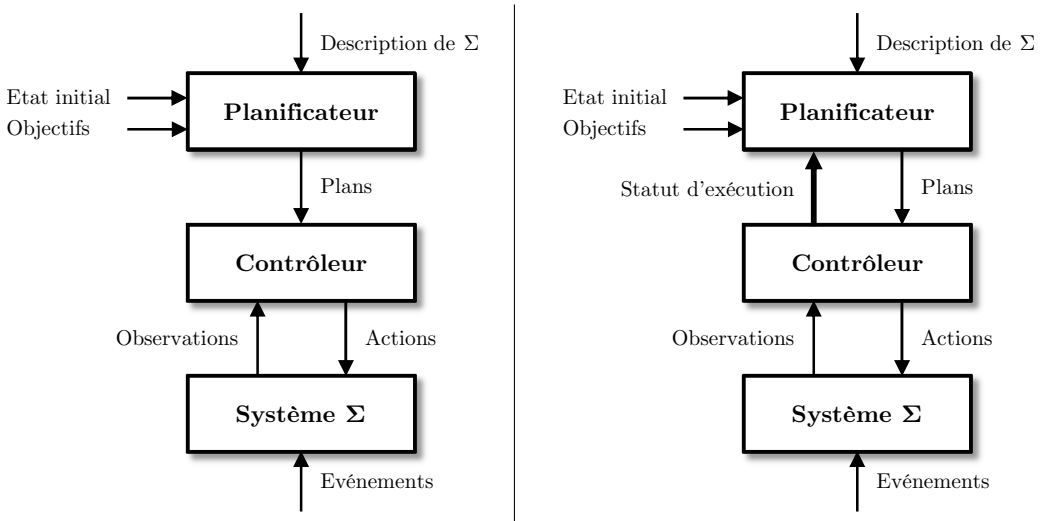
1.1.1 Modèle conceptuel de la planification

La planification a pour objectif de faire évoluer un système Σ dans un état initial connu vers un état final satisfaisant un ensemble d'objectifs fixés. Dans son acception la plus générique, l'activité de la planification englobe à la fois l'élaboration du plan d'action considéré ainsi que l'exécution de ce dernier. Le modèle conceptuel de la planification formalise l'activité de la planification et a été proposé dans l'ouvrage de référence *Automated Planning : Theory and Practice* [66]. Il repose sur les interactions de trois composants : un planificateur, un contrôleur et un système Σ (cf. figure 1.1a).

1. Le planificateur construit le plan d'action en se basant sur la description du système Σ, de son état initial et des objectifs à atteindre.
2. Le contrôleur exécute le plan qui lui est transmis produisant ainsi des actions sur le système. Il s'appuie sur les informations (éventuellement incomplètes) qu'il possède sur l'état actuel du système.
3. Le système Σ évolue en réponse aux actions qui sont exécutées par le contrôleur ou aux événements extérieurs qui surviennent.

Le contrôleur est généralement supposé assez robuste pour gérer les différences qui peuvent exister entre le monde réel et son modèle Σ. Si cette hypothèse n'est pas acceptable, le contrôleur peut retourner un statut d'exécution au planificateur permettant à ce dernier de produire un nouveau plan lorsque la situation observée

et la situation attendue divergent. Le terme de planification dynamique est alors employé puisque la création du plan et son exécution deviennent intimement liées (cf. boucle de rétroaction sur la figure 1.1b).



(a) Version statique (b) Version dynamique

FIGURE 1.1 – Modèle conceptuel de la planification [66]

D'un point de vue formel, le modèle des systèmes à événements discrets [40] est utilisé pour spécifier et capturer l'évolution dynamique du système Σ.

Définition 1.1 - Système à événements discrets [40]
Un système à événements discrets est un quadruplet $\Sigma = (S, A, E, \gamma)$ avec :

- S un ensemble fini ou récursivement énumérable d'états ;
- A un ensemble fini ou récursivement énumérable d'actions ;
- E un ensemble fini ou récursivement énumérable d'événements ;
- $\gamma : S \times A \times E \to \mathfrak{P}(S)$ une fonction de transition d'états.

Modéliser Σ comme un système à événements discrets revient à caractériser ce dernier par l'état $s \in S$ dans lequel il se trouve. Une évolution du système Σ correspond donc à un changement d'état $s \to s'$. Dans le modèle conceptuel de la planification, la transition d'un état s vers un état s' est représentée par l'application d'un couple action/événement (a, e) dans l'état s. L'action neutre α et l'événement

neutre ϵ sont introduits afin de décrire des transitions provoquées uniquement par des actions ou des événements. Dans de tels cas, les notations $\gamma(s, a)$ et $\gamma(s, e)$ sont utilisées en lieu et place de $\gamma(s, a, \epsilon)$ et $\gamma(s, \alpha, e)$. Il convient de remarquer que bien que les actions et les événements contribuent tous deux à l'évolution du système, leur sémantique diffère. Les actions sont contrôlées par les personnes en charge de l'exécution du plan alors que les événements sont des transitions qui échappent à leur contrôle. Ces derniers peuvent être causés par la dynamique interne du système ou encore par l'évolution de l'environnement du système.

La fonction γ précise l'ensemble des états s' vers lesquels le système Σ est susceptible d'évoluer à partir de l'état s en réponse à une action a, un événement e ou un couple action/événement (a, e). Ainsi, la taille de l'ensemble retourné par $\gamma(s, a)$ notée $|\gamma(s, a)|$ fournit plusieurs informations quant à l'exécution de a dans s. Si $|\gamma(s, a)| > 0$, l'action a est dite applicable dans s puisque le système Σ peut évoluer vers au moins un état s' lors de l'exécution de a dans s. De plus, si $|\gamma(s, a)| \leq 1$ alors l'application de a dans s est dite déterministe. En effet, si l'action a est exécutable et qu'elle est exécutée dans s, Σ ne peut alors évoluer que vers un unique état s'.

Le modèle conceptuel de la planification [66] propose également huit hypothèses pour caractériser les différentes classes de problèmes de planification.

Hypothèse H1 (Σ fini). L'ensemble S des états de Σ est fini.

Hypothèse H2 (Σ complètement observable). L'état du système Σ est entièrement observable. Par conséquent, les observations du système sur lesquelles s'appuie le contrôleur sont toutes parfaites.

Hypothèse H3 (Σ déterministe). Le système Σ est déterministe si pour tout état s, pour toute action a et pour tout événement e, $|\gamma(s, a, e)| \leq 1$. Ainsi, en réponse à un couple action/événement (a, e) applicable dans s, le système Σ ne peut évoluer que vers un unique état s'.

Hypothèse H4 (Σ statique). L'ensemble E des événements de Σ est vide.

Hypothèse H5 (Plans séquentiels). Un problème de planification admet des plans solutions représentés par une séquence d'actions finie ordonnée linéairement, il n'y a donc aucune parallélisme dans les plans solutions.

Hypothèse H6 (Objectifs restreints). Les objectifs à atteindre ne portent que sur l'état final du système. Ainsi, une solution est une séquence de transitions d'états qui aboutit à un état final $s \in S_G$ avec S_G l'ensemble des états dans lesquels les objectifs sont vérifiés.

Hypothèse H7 (Temps implicite). Les actions et événements n'ont aucune durée intrinsèque, les transitions sont donc considérées instantanées.

Hypothèse H8 (Planification statique). Aucun mécanisme de planification dynamique n'est mis en œuvre.

1.1.2 Problème de la planification classique

Le terme *planification classique* fait référence à la classe des problèmes obtenue lorsque les huit hypothèses du modèle conceptuel de la planification sont considérées simultanément. L'étude de ce problème est fondamentale puisque la majorité des problèmes de planification sont définis par extension de ce dernier à l'image de la problématique de la planification avec préférences.

Problème de la planification classique [66] *Étant donné un système à événements discrets $\Sigma = (S, A, \gamma)$, un problème de planification classique est défini par le triplet $PC = (\Sigma, s_0, S_G)$ où s_0 est un état initial et $S_G \subset S$ est l'ensemble des états qui vérifient les objectifs G.*

Une solution de PC est une séquence d'actions $\langle a_1, \cdots, a_n \rangle$ qui correspond à une séquence d'états $\langle s_0, \cdots, s_n \rangle$ telle que :

$$s_1 = \gamma(s_0, a_1), \cdots, s_n = \gamma(s_{n-1}, a_n) \quad et \quad s_n \in S_G.$$

Exemple illustratif

Pour illustrer la problématique de la planification classique, une version simplifiée du problème *Rovers* [42] est utilisée. Dans ce problème, des robots réalisent une exploration planétaire. Ils doivent prendre des photographies et récolter des échan-

tillons de sols ou de roches de plusieurs lieux différents. La récolte des échantillons de sols et de roches est modélisée par deux actions distinctes dans ce problème pour préciser que les robots utilisent des outils différents dans les deux cas.

L'exemple considéré ici s'intéresse au cas d'un unique robot $N1$ qui évolue à travers six lieux Li différents. L'ensemble { robot, planète } constitue le système Σ et évolue en fonction des actions A du robot : déplacement d'un lieu à un autre, prise de photographies et récolte d'échantillons. Tous les lieux ne sont pas accessibles les uns des autres en raison de la présence d'obstacles infranchissables par le robot. En outre, pour pouvoir prendre la photographie d'un lieu $L1$, le robot doit soit se situer dans $L1$ soit dans un lieu $L2$ à partir duquel $L1$ est visible. Dans l'état initial s_0 (représenté sur la figure 1.2), le robot est en $L5$, des échantillons de sols sont présents dans tous les lieux mais seuls les lieux $L2$, $L3$ et $L6$ possèdent des échantillons de roches. Enfin, les objectifs G sont notés [S4], [R3] et [P1] et désignent respectivement l'acquisition d'un échantillon de sol de $L4$, l'acquisition d'un échantillon de roche de $L3$ et la prise d'une photographie de $L1$.

Cet exemple sera enrichi (utilisation de plusieurs robots...) tout au long de ce chapitre pour présenter les mécanismes de la planification ainsi que pour illustrer le problème de la planification avec préférences (cf. sections 1.2.2, 1.1.3 et 1.2.3).

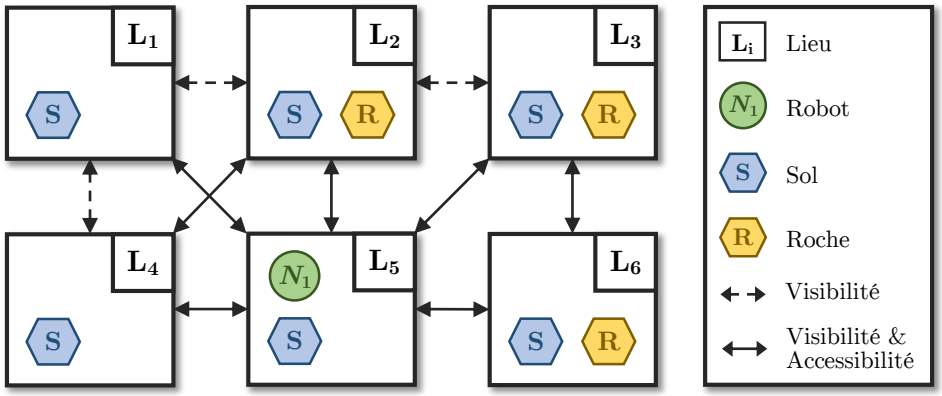


FIGURE 1.2 – Situation initiale de l'exemple *Rovers*

Le plan 1 est l'une des solutions du problème considéré. En effet, la séquence d'actions proposée est valide au regard de la fonction de transition γ et les objectifs [S4], [R3] et [P1] sont tous atteints dans l'état final.

Plan 1
1. *Naviguer avec N1 de L5 vers L3*
2. *Prélever échantillon roche avec N1 en L3*
3. *Naviguer avec N1 de L3 vers L5*
4. *Naviguer avec N1 de L5 vers L4*
5. *Prélever échantillon sol avec N1 en L4*
6. *Photographier L1 avec N1 en L4*

FIGURE 1.3 – Plan 1 : Solution de l'exemple *Rovers*

Du fait de l'objectif d'illustration de cette section, l'exemple retenu a été volontairement choisi de sorte à être très facile à appréhender et à résoudre. Les problèmes de planification sont généralement plus complexes et fortement combinatoires à l'image de celui mentionné dans l'annexe B.5. Pour prendre la mesure de cette complexité, un exemple inspiré des problématiques rencontrées dans les plateformes de transports multimodales est considéré. Dans ce problème, on cherche à déplacer un certain nombre de containers d'un point de stockage à un autre. Les containers peuvent être empilés les uns sur les autres formant ainsi des piles. Chaque point de stockage peut abriter un certain nombre de piles. Pour résoudre le problème, plusieurs robots capables de soulever et transporter des containers sont mis à disposition. Lorsque ce problème est instancié avec cinq points de stockage, trois piles par point de stockage, trois robots et cent containers, le système peut se trouver dans approximativement 10^{277} états différents [66]. A titre de comparaison, l'âge de l'univers est estimé à 10^{17} secondes [19] et l'univers observable serait constitué de 10^{80} atomes [46].

1.1.3 Langage formel pour la planification

Afin de standardiser la représentation des problèmes de planification, le langage PDDL (Planning Domain Definition Language) a été proposé en 1998 par la communauté de la planification [107]. Le PDDL est couramment utilisé et a connu quatre évolutions majeures depuis sa création [49, 61, 64, 91]. Cette section a pour but de présenter en détails les mécanismes de la planification. A cette fin, la syntaxe du PDDL et la sémantique qui y est associée sont introduits. Les éléments de

sémantique présentés traitent de la planification classique et de l'exploitation de variables numériques. En outre, ils sont illustrés à l'aide de l'exemple introduit dans la section 1.1.2.

Syntaxe pour la planification

Le PDDL a été élaboré en s'inspirant de plusieurs travaux antérieurs dont notamment les langages STRIPS [99] et ADL [114], le formalisme UCPOP [11] ainsi que le *situation calculus* [106, 120]. Par ailleurs, le PDDL présente de nombreuses similarités avec la logique du premier ordre. Les notions de *prédicats*, d'*arité*, de *formules*, d'*atomes*, de *littéraux* et de *formules closes* qui sont utilisées dans la suite de cette section sont définies dans l'annexe A qui présente la syntaxe de la logique du premier ordre.

L'ensemble des états caractérisant un problème de planification ne peut généralement pas être énuméré explicitement au vu de sa grande taille. Pour contourner ce problème et représenter de façon compacte les problèmes de planification, les états sont donc caractérisés à l'aide de formules de la logique du premier ordre dont les variables peuvent éventuellement être typées. Ainsi, pour formaliser un problème de planification en PDDL, les principaux éléments à renseigner sont des prédicats et des objets, des actions, une situation initiale et des objectifs et contraintes. La notation BNF [4] est utilisée pour décrire avec précision et sans ambigüité la syntaxe du PDDL [63, 91]. Quelques extraits du langage PDDL sont exposés ci-dessous. Des exemples complets sont disponibles dans l'annexe B.

```
(:objects
  N1 - robot
  L1 - lieu   L2 - lieu   L3 - lieu
  L4 - lieu   L5 - lieu   L6 - lieu
)
```

FIGURE 1.4 – Objets de l'exemple *Rovers*

```
(:predicates
  (presence_echantillon_sol ?x - lieu)
  (presence_echantillon_roche ?x - lieu)
  (visible ?x - lieu ?y - lieu)
  (accessible ?x - lieu ?y - lieu)
  (position ?x - robot ?y - lieu)
  (possede_echantillon_sol ?x - robot ?y - lieu)
  (possede_echantillon_roche ?x - robot ?y - lieu)
  (possede_photographie ?x - robot ?y - lieu)
)
```

FIGURE 1.5 – Prédicats de l'exemple *Rovers*

```
(:action Naviguer
  :parameters (?x - robot ?y - lieu ?z - lieu)
  :precondition (and
    (position ?x ?y) (accessible ?y ?z))
  :effect (and
    (not (position ?x ?y)) (position ?x ?z))
)
```

FIGURE 1.6 – Action Naviguer de l'exemple *Rovers*

```
(:init
  (presence_echantillon_sol L1)
  (presence_echantillon_sol L2)
  (presence_echantillon_roche L2)
  ...
  (visible L1 L2) (visible L2 L1)
  (accessible L1 L5) (accessible L5 L1)
  ...
  (position N1 L5)
)
```

FIGURE 1.7 – Situation initiale de l'exemple *Rovers*

```
(:goal (and
  (possede_echantillon_sol N1 L4)
  (possede_echantillon_roche N1 L3)
  (possede_photographie N1 L1)
))
```

FIGURE 1.8 – Objectifs pour l'exemple *Rovers*

Sémantique pour la planification classique

La syntaxe du PDDL définit l'ensemble des symboles utilisables pour formaliser un problème de planification. Il convient à présent d'introduire la sémantique qui y est associée afin de préciser le sens de ces symboles. Cette dernière permet d'expliquer comment les problèmes de planification sont résolus d'un point de vue conceptuel. En conséquence, elle constitue une étape fondamentale de la résolution des problèmes de planification.

Les définitions présentées sont issues de la version 2.1 du PDDL [61]. Elles ont été simplifiées afin de se limiter aux seuls éléments utilisés dans le cadre de cette étude. En particulier, les aspects temporels formalisés dans [61] n'ont pas été retenus. La notion d'instance d'un problème de planification est présentée par la définition 1.2. Les définitions 1.3 et 1.4 précisent la structure des états S du système Σ considéré. Les définitions 1.5 à 1.9 s'intéressent quant à elles aux actions du problème. Finalement, la notion de plan valide est introduite à l'aide des définitions 1.10 à 1.13.

Définition 1.2 - Instance d'un problème de planification [61]
Une instance d'un problème de planification classique I_C est une paire $(Dom, Prob)$ définie telle que :

- $Dom = (R, A)$ est le *domaine de planification*. Il est constitué d'un ensemble de prédicats R et d'un ensemble d'actions A.
- $Prob = (O, s_0, G)$ est le *problème de planification*. Il est constitué d'un ensemble d'objets O et des formules s_0 et G qui décrivent respectivement l'état initial et les objectifs à atteindre dans l'état final.

L'objet *Dom* constitue une caractérisation générique de l'instance I_C (prédicats considérés et actions réalisables) tandis que *Prob* précise les caractéristiques spécifiques de l'instance I_C (état initial et objectifs du problème). Par conséquent, un domaine de planification peut être utilisé par plusieurs problèmes de planification différents définissant alors autant d'instances de planification.

L'ensemble Atm_{I_C} des atomes de I_C est formé par l'application des prédicats du domaine R aux objets du problème O (en respectant l'arité des prédicats).

Les notions de *structure d'état* et de *valeur de vérité d'une formule dans un état* sont à présent introduites. Considérées conjointement, ces deux définitions permettent de décrire l'ensemble des états dans lequel un système à événements discrets Σ peut se trouver sans pour autant devoir les énumérer explicitement.

Définition 1.3 - Structure d'un état [61]
Étant donné une instance I_C, un état s est défini par la donnée d'un ensemble $Atm \in \mathfrak{P}(Atm_{I_C})$. Atm est appelé l'état logique de s et constitue un sous-ensemble des atomes de I_C.

Définition 1.4 - Valeur de vérité d'une formule [61]
Soit un état $s = (Atm)$, la formule atomique ϕ est vraie dans s si et seulement si $\phi \in Atm$. La notation $s \models \phi$ est utilisée pour dénoter que ϕ est vraie dans s.

La valeur de vérité d'une formule ϕ quelconque (qui peut contenir des connecteurs et quantificateurs logiques comme précisé dans l'annexe A) est déterminée à partir des valeurs de vérité des formules atomiques qui la composent en utilisant l'interprétation usuelle de la logique du premier ordre [37].

Dans l'exemple de la section 1.1.2, le lieu $L4$ est accessible et visible depuis le lieu $L5$ (cf. figure 1.2). Soit $s_0 = (Atm_0)$ l'état initial du problème et ϕ_1, ϕ_2 les formules atomiques qui correspondent aux prédicats PDDL `(accessible L4 L5)` et `(visible L4 L5)`. Les formules ϕ_1 et ϕ_2 sont vraies dans s_0 ($\phi_1, \phi_2 \in Atm_0$) ce qui permet notamment d'écrire que la formule $\phi_3 = \phi_1 \wedge \phi_2$ l'est également ($s_0 \models \phi_3$).

De la même façon que les prédicats sont instanciés avec les objets du problème, les actions d'une instance sont successivement transformées en *actions aplanies* puis en *actions closes* (cf. définition 1.5 à 1.7). Une fois ces transformations effectuées, il devient possible d'introduire les notions d'*applicabilité d'une action* dans un état et de résultat de l'*exécution d'une action* dans un état (voir définitions 1.8 et 1.9).

Définition 1.5 - Structure d'une action [61]
Une action $a \in A$ d'une instance I_C est un triplet ($\hat{a}$, Pre_a, Eff_a) avec :

- $\hat{a}$: le nom de l'action
- Pre_a : une formule qui décrit les préconditions de a
- Eff_a : une formule qui décrit les effets de a

Définition 1.6 - Actions aplanies [61]
Étant donné une instance I_C, l'ensemble des *actions aplanies* FA est défini comme l'ensemble contenant initialement A et construit de la façon suivante :

> Tant que FA contient une action a ayant une formule avec un quantificateur, remplacer a par une version dans laquelle la formule quantifiée (Q (v_1, $\cdots$, v_k) P) est remplacée par la conjonction (si Q est le quantificateur universel) ou la disjonction (si Q est le quantificateur existentiel) des atomes obtenus par toutes les substitutions possibles des objets de I_C dans les variables v_1, $\cdots$, v_k du prédicat P.

Afin d'illustrer le mécanisme de génération des actions aplanies, l'action a_1 `Panorama_planète ?x - robot ?y - lieu` est considérée. Pour que a_1 soit exécutable, le robot doit se trouver dans un lieu à partir duquel tous les lieux de la planète sont visibles. Ainsi, Pre_{a_1} est représenté par la formule :

```
(and (position ?x ?y) (forall (?z - lieu) (visible ?y ?z)))
```

Après aplanissement de a_1, la forme de Pre_{a_1} devient :

```
(and (position ?x ?y) (visible ?y L1) (visible ?y L2)
(visible ?y L3) (visible ?y L4) (visible ?y L5) (visible ?y L6))
```

Définition 1.7 - Actions closes [61]
Soit une instance I_C et une action $a_F \in FA$ formée par aplanissement de $a \in A$. L'ensemble des *actions closes* de a noté GA_a est formé en réalisant l'ensemble des substitutions possibles de variables de a_F par des objets de I_C. De plus, l'ensemble $GA = \bigcup_{a \in A} GA_a$ est l'ensemble des actions closes de I_C.

Par construction (et par définition de la syntaxe du PDDL [63]), si a_G est une action close, alors Eff_{a_G} est une conjonction de littéraux. Une action close peut donc être vue comme un quadruplet ($\hat{a_G}$, Pre_{a_G}, Add_{a_G}, Del_{a_G}) avec :

- $\hat{a_G}$: le nom de l'action et des objets substitués aux variables de a
- $GPre_{a_G}$: les préconditions de a_G représentées par une formule close
- Add_{a_G} : les effets positifs de a_G constitués de l'ensemble des atomes qui sont considérés comme des littéraux positifs dans Eff_{a_G}
- Del_{a_G} : les effets négatifs de a_G constitués de l'ensemble des atomes qui sont considérés comme des littéraux négatifs dans Eff_{a_G}

L'action a_2 `Naviguer ?x - robot ?y - lieu ?z - lieu` présentée sur la figure 1.6 est utilisée pour illustrer la construction des actions closes. Dans le cadre de l'exemple de la section 1.1.2, l'ensemble GA_{a_2} contient 36 actions (1 robot $\times$ 6 lieux $\times$ 6 lieux) dont notamment l'action `Naviguer N1 L3 L4`. L'atome `(position N1 L4)` est l'unique effet positif de cette action de même que l'atome `(position N1 L3)` est l'unique effet négatif de cette dernière.

Définition 1.8 - Applicabilité d'une action [61]
Étant donné une action $a \in GA$, a est applicable dans s si $s \models Pre_a$.

Définition 1.9 - Execution d'une action [61]
Soit un état s et une action $a \in GA$. Si a est applicable dans s, la fonction $\gamma : S \times GA \rightarrow S$ définit l'état résultant de l'exécution de a dans s par :

$$\gamma(s, a) = (Atm \setminus Del_a) \cup Add_a$$

L'action a_3 `Naviguer N1 L5 L3` a pour préconditions $GPre_{a_3} = \phi_4 \wedge \phi_5$ avec `(position N1 L5)` et `(accessible L5 L3)` les prédicats correspondants respectivement à ϕ_4 et ϕ_5. Puisque $s_0 \models \phi_4$ et $s_0 \models \phi_5$, il en résulte que a_3 est exécutable

dans l'état initial s_0. De plus, Del_{a_3} et Add_{a_3} sont respectivement caractérisés par les atomes `(position N1 L5)` et `(position N1 L3)`. Ainsi, l'exécution de a_3 dans s_0 conduit à un état s_1 dans lequel le robot $N1$ n'est plus dans le lieu $L5$ mais dans le lieu $L3$.

A présent que les mécanismes de transition d'états ont été précisés, il est possible de définir la notion de *plan exécutable* et de *plan valide*.

Définition 1.10 - Structure d'un plan [61]
Étant donné une instance I_C, un plan de I_C est une séquence $\langle \hat{a}_1, \cdots, \hat{a}_n \rangle$ telle que pour tout $i \in \{1, \cdots, n\}$, $\hat{a}_i$ est le nom d'une action close.

Définition 1.11 - Trajectoire d'un plan [64]
Étant donné un plan $x = \langle \hat{a}_1, \cdots, \hat{a}_n \rangle$, la séquence d'états $\langle s_0, \cdots, s_n \rangle$ définie telle que $s_1 = \gamma(s_0, a_1)$, $\cdots$, $s_n = \gamma(s_{n-1}, a_n)$ est appelée *trajectoire de* x.

Définition 1.12 - Exécutabilité d'un plan [61]
Soit une instance I_C et un plan $x = \langle \hat{a}_1, \cdots, \hat{a}_n \rangle$ de trajectoire $\langle s_0, \cdots, s_n \rangle$. Si pour tout $i \in \{1, \cdots, n\}$, a_i est applicable dans s_{i-1} alors x est exécutable.

Définition 1.13 - Validité d'un plan [61]
Soit une instance I_C et un plan $x = \langle \hat{a}_1, \cdots, \hat{a}_n \rangle$ de trajectoire $\langle s_0, \cdots, s_n \rangle$. Le plan x est valide s'il est exécutable et si $s_n \models G$.

Le lecteur peut vérifier que le plan 1 présenté dans la section 1.1.2 est valide. En effet, il vérifie les objectifs présentés par la figure 1.8 tout en étant exécutable par rapport à la situation initiale s_0 (voir figures 1.2 et 1.7).

Sémantique pour les expressions numériques

Cette section enrichit la sémantique de la planification classique avec la notion d'expressions numériques. Ces dernières permettent de représenter des problèmes de planification plus complexes et sont notamment utilisées dans le chapitre 3. Les définitions 1.14 à 1.21 complètent la sémantique introduite par les définitions 1.2 à 1.13 en lui ajoutant les éléments liés aux expressions numériques. Pour illustrer

ces nouvelles notions, l'exemple précédent est étendu en considérant plusieurs robots et en tenant compte de la quantité d'énergie qu'ils consomment lors de leurs déplacements. Afin de simplifier le problème, il sera considéré que chaque déplacement consomme 10 unités d'énergie. De plus, un robot est autorisé à se déplacer uniquement s'il a consommé moins de 75 unités d'énergie depuis le début de la mission.

Définition 1.14 - Instance d'un problème de planification [61]
Une instance d'un problème de planification avec expressions numériques I_N est redéfinie par extension de la définition 1.2 en ajoutant un ensemble F de *fonctions numériques* à $Dom = (F, R, A)$.

L'ensemble PNE_{I_N} des *expressions numériques primitives* de I_N est formé par l'application des fonctions F du domaine aux objets O du problème (en respectant l'arité des fonctions). La dimension de I_N notée *dim* est le nombre d'expressions numériques qui peuvent être construites dans I_N et vaut $|PNE_{I_N}|$.

Il convient de ne pas confondre les fonctions de F qui retournent un réel et les fonctions de la logique du premier ordre qui retournent un objet de I_N. De telles fonctions existent en PDDL [91] mais ne sont pas décrites ici puisqu'elles ne sont pas considérées dans le cadre de cette étude.

Pour prendre en compte cette modification, un vecteur de valeurs numériques doit maintenant être associé à chaque état. De plus, les formules peuvent à présent contenir des opérateurs de comparaison ($=$, $\geq$, $\leq$, $>$, $<$, ...).

Définition 1.15 - Structure d'un état [61]
Par extension de la définition 1.3, un état s est une paire (Atm, v) avec $v \in \mathbb{R}_{\perp}^{dim}$ où $\mathbb{R}_{\perp} = \mathbb{R} \cup \{\perp\}$ et $\perp$ désigne une valeur non définie.

Soit deux variables e_1 et e_2 correspondant à l'énergie consommé par les robots $N1$ et $N2$ lors de leurs déplacements. Chaque état s contient à présent un vecteur $v = (e_1, e_2)$ qui précise les valeurs de e_1 et e_2 dans s.

Définition 1.16 - Normalisation d'une formule [61]
Étant donné une instance I_N de dimension dim, une formule ϕ et un vecteur $V \in \mathbb{R}^{dim}$, soit $Ind : PNE_{I_N} \rightarrow \{1, \cdots, dim\}$ une fonction qui associe à toute expression numérique de I_N l'indice d'un élément de V.

La *forme normalisée* de ϕ par rapport à V notée $\mathcal{N}(\phi)[V]$ est obtenue en substituant chaque expression numérique primitive f de ϕ par le réel $V_{Ind(f)}$.

Définition 1.17 - Valeur de vérité d'une formule [61]
Soit un état $s = (Atm, v)$, la valeur de vérité d'une formule ϕ est définie par extension de la définition 1.4 à partir de $\mathcal{N}(\phi)[V := v]$ la forme normalisée de ϕ dans s en utilisant l'interprétation usuelle des opérateurs de comparaison.

Soit `(<= (energy ?x - robot) 75)` la condition supplémentaire de déplacement des robots. Dans le cas de l'action a_3 `Naviguer N1 L5 L3`, cette formule ϕ_6 s'écrit `(<= (energy N1) 75)` où `(energy N1)` représente l'expression numérique primitive qui associe au robot $N1$ sa consommation d'énergie. Ainsi, la forme normalisée de ϕ_6 par rapport à un vecteur $v = (e_1, e_2)$ est `(<= e1 75)` puisque $Ind($`energy N1`$) = 1$. La précondition ϕ_6 est vraie dans un état s si $s \models \mathcal{N}(\phi_6)[v]$ ce qui s'interprète naturellement par $e_1 \leq 75$.

Les actions peuvent modifier les valeurs numériques des états par l'intermédiaire d'*effets numériques* qui sont constitués de l'opérateur d'affectation `assign`, d'une expression numérique primitive notée *lvalue* et d'une *expression numérique* (expression arithmétique quelconque dont les termes sont des réels ou des expressions numériques primitives) notée *rvalue* [61].

Définition 1.18 - Formule d'affectation [61]
La *formule d'affectation* associée à un effet numérique est formée en remplaçant l'opérateur d'affectation `assign` par l'opération arithmétique $=$ et en annotant l'expression numérique primitive *lvalue* d'un symbole « prime ».

La *forme normalisée* d'une formule d'affectation ϕ par rapport à (V', V) notée $\mathcal{N}(\phi)[V', V]$ est obtenue en substituant chaque expression numérique primitive f de ϕ par le réel $V'_{Ind(f)}$ si f est annotée d'un symbole « prime » et par le réel $V_{Ind(f)}$ dans le cas contraire.

A titre d'exemple, soit l'effet numérique `(assign (energy ?x - robot) (+ (energy ?x - robot) 10))` qui représente la consommation d'énergie des robots lors de leurs déplacements. Dans le cas de l'action a_3, l'expression numérique primitive *lvalue* est `(energy N1)` et l'expression numérique *rvalue* est `(+ (energy N1) 10)`. La formule d'affectation ϕ_7 associée à cet effet numérique est `(= (energy N1)' (+ (energy N1) 10))`. Par conséquent, la forme normalisée de ϕ_7 par rapport à (v', v) avec $v' = (e'_1, e'_2)$ et $v = (e_1, e_2)$ s'exprime `(= e1' (+ e1 10))` ce qui s'interprète par l'affectation $e'_1 = e_1 + 10$.

Définition 1.19 - Actions closes [61]
Une action close est redéfinie par extension de la définition 1.7 à l'aide d'un quintuplet ($\hat{a_G}$, Pre_{a_G}, Add_{a_G}, Del_{a_G}, NEf_{a_G}) avec :

- NEf_{a_G} : les effets numériques de a_G constitués de l'ensemble des formules d'affectation associées aux effets numériques de Eff_{a_G}

Définition 1.20 - Applicabilité d'une action [61]
La notion d'applicabilité d'une action $a \in GA$ dans un état s est redéfinie par extension de la définition 1.8 en ajoutant la condition suivante : aucune expression numérique primitive n'est annotée d'un symbole « prime » plus d'une fois lors de la construction des effets numériques de NEf_a.

Définition 1.21 - Execution d'une action [61]
La fonction $\gamma : S \times GA \rightarrow S$ qui précise l'état $s' = (Atm', v')$ résultant de l'exécution d'une action $a \in GA$ dans un état $s = (Atm, v)$ est redéfinie par extension de la définition 1.9 par l'ajout de la fonction NF obtenue par composition des fonctions de l'ensemble $\{ NF_\phi : \mathbb{R}_\perp^{dim} \rightarrow \mathbb{R}_\perp^{dim} \mid \phi \in NEf_a \}$ avec :

- $NF_\phi(v) = v'$ où $v'_{Ind(f)}$ est défini tel que $\mathcal{N}(\phi)[V' := v', V := v]$ soit satisfaite si f est une expression numérique primitive annotée d'un symbole « prime » ; et $v'_{Ind(f)} = v_{Ind(f)}$ dans le cas contraire.

La condition supplémentaire d'applicabilité permet à toutes les affectations de valeurs numériques d'une action d'être consistantes entre elles. Il en résulte que la loi de composition des fonctions est commutative pour l'ensemble des fonctions NF_ϕ ce qui permet de construire NF sans se soucier de l'ordre dans lequel les fonctions NF_ϕ sont appliquées.

La définition 1.21 peut sembler complexe de prime abord mais son interprétation est relativement naturelle. En effet, la fonction NF est définie de sorte que les affectations de variables réalisées coïncident avec l'effet numérique considéré. Ainsi, dans un problème avec deux robots $N1$ et $N2$ dont les consommations d'énergie sont nulles dans l'état initial s_0, cette définition garantit que l'exécution de l'action a_3 (à savoir un déplacement de $N1$) conduit effectivement à un état s_1 dans lequel $e_1 = 10$ et $e_2 = 0$.

A l'exception des considérations sur les expressions numériques, cette première section s'est focalisée sur le problème de la planification classique. Ce dernier est parfois trop limité pour modéliser correctement certains problèmes réels. Dans ce cas, il est nécessaire de considérer des classes de problèmes plus complexes qui ne reposent que sur un sous-ensemble des hypothèses du modèle conceptuel de la planification (cf. section 1.1.1). Ces nouvelles classes de problèmes forment autant de sous-domaines de la planification à l'image de la *planification temporelle* qui ne considère pas H7, de la *planification dans l'incertain* qui ne considère pas H2, H3 et H5 ou encore de la *planification avec préférences* qui ne considère ni H1 ni H6.

Domaine de la gestion de crise

L'élaboration d'une réponse à une situation critique par les membres d'une cellule de crise constitue un problème de planification. En effet, à partir d'une situation initiale connue (la situation de crise), les décideurs se fixent des objectifs puis essayent de déterminer un plan d'action permettant de les atteindre. Le plan solution séquence les différentes actions qui doivent être mises en œuvres par les partenaires mobilisés afin de résoudre la crise. Il précise à ces derniers ce qu'ils doivent faire mais les laisse entièrement autonomes quant à la réalisation des actions qui leurs sont assignées (principe de subsidiarité).

Les définitions proposées s'appuient sur les notions d'objet, d'état, d'action et d'expressions numériques. En gestion de crise, un état représente simplement l'état du monde à un instant donné et constitue donc une description de l'ensemble des éléments qu'il est pertinent de considérer au vu du problème à

résoudre. Par exemple, pour représenter qu'un bâtiment b_0 est en feu au début de la crise, il est possible de créer un objet bâtiment b_0 et un prédicat $(enFeu\ ?b)$ (pouvant caractériser n'importe quel bâtiment). L'atome $(enFeu\ b_0)$ qui est obtenu par combinaison de ces deux éléments appartient alors à l'état initial s_0. L'action « Eteindre incendie » des pompiers pourrait être utilisée pour passer de l'état s_0 à un état s_1 qui ne contiendrait plus l'atome $(enFeu\ b_0)$. Un autre exemple d'action que les pompiers peuvent être amenés à exécuter consiste à inonder une infrastructure en eau propre dans le but de la protéger (métro, bâtiment...) en cas d'inondation. Par ailleurs, les expressions numériques peuvent être employées pour dénombrer les ressources gérées par la cellule de crise à l'image de denrées alimentaires par exemple.

Pour finir, il convient de discuter de la pertinence des hypothèses du modèle conceptuel de la planification pour le domaine de la gestion de crise. L'approche de planification avec préférences retenue dans cette étude (cf. section 1.2) ne considère pas les hypothèses H1 et H6. Par ailleurs, les hypothèses H2, H3 et H4 ne sont pas toujours compatibles avec la gestion de crise puisque le système Σ n'est pas nécessairement observable, déterministe ou statique. Elles peuvent néanmoins être considérées si l'hypothèse H8 est relaxée. En effet, si de nouveaux éléments concernant l'état de Σ sont portés à la connaissance des décideurs lors de l'exécution du plan solution (violation de H2) ou qu'une action ne produit pas les effets attendus (violation de H3) ou encore qu'un événement inattendu survient (violation de H4), il est possible, dans le cas où l'hypothèse H8 est relaxée, de ré-exécuter le processus de planification pour adresser ces problèmes (approche de type planification dynamique, voir section 4.1.2 pour une discussion à ce propos). De plus, les hypothèses H5 et H7 peuvent également être considérées trop restrictives pour le domaine de la gestion de crise. Il est néanmoins possible de les relaxer en s'appuyant sur des mécanismes d'ordonnancement et des mécanismes de planification temporelle (voir discussion section 4.2.3). Ainsi, des plans d'action pour la gestion de crise peuvent être construits en ne considérant que les hypothèses H2, H3 et H4 dans le cas où une démarche de planification dynamique est utilisée. Dans le cadre de cette étude, l'hypothèse H7 est également retenue puisqu'aucun mécanisme de planification temporelle n'est considéré.

1.2 De la problématique de la planification avec préférences

La thématique des préférences a été très étudiée par la communauté de l'Intelligence Artificielle [32, 44, 67, 117, 121]. Il apparait clairement que les préférences d'un décideur peuvent être modélisées avec beaucoup de diversité. En conséquence, un grand nombre de formalismes au pouvoir expressif varié ont été proposés dans la littérature. Le domaine de la planification automatisée ne déroge pas à cette règle et plusieurs approches ont été suggérées pour résoudre les problèmes de planification avec préférences.

La planification avec préférences se distingue de la planification classique par l'utilisation d'objectifs dits étendus. Ces derniers surclassent les objectifs restreints considérés dans le cadre de l'hypothèse H6 et sont présentés dans la section 1.2.1. Le problème de la planification avec préférences est ensuite introduit dans la section 1.2.2. La section 1.2.3 précise quant à elle les divers formalismes utilisés pour représenter ce type de problèmes. Elle se focalise principalement sur la sémantique du PDDL3 qui est le formalisme le plus employé mais présente également quelques langages alternatifs. Pour finir, les méthodes et algorithmes de l'art employés pour résoudre les problèmes de planification avec préférences sont exposés dans la section 1.2.4.

1.2.1 Objectifs étendus et préférences

Les objectifs étendus peuvent être de natures différentes. On distingue les *objectifs* (ou contraintes fortes) qui doivent nécessairement être vérifiés par les solutions du problème des *préférences* (ou contraintes faibles) qui peuvent ne pas être vérifiées par un plan solution.

Il est possible de subdiviser à nouveau les objectifs en deux catégories. Les *objectifs finaux* correspondent exactement aux objectifs restreints de l'hypothèse H6. Des *objectifs de trajectoires* peuvent également être considérés afin de contraindre les séquences d'action solutions d'un problème. Ils permettent par exemple d'interdire les transitions vers certains états spécifiques ou encore de forcer le système à passer par un état avant un autre...

A l'image des objectifs, les préférences peuvent représenter des conditions sur les états finaux (*préférences finales*) ou sur les états intermédiaires (*préférences de trajectoires*) des plans solutions. Néanmoins, elles peuvent également porter sur n'importe quelle variable numérique (*préférences numériques*) définie dans le problème. Ceci leur confère un pouvoir expressif certain puisqu'elles peuvent aussi bien décrire des coûts et des risques à minimiser (coût de production, délai de livraison...) que des quantités ou des gains à maximiser (par exemple un niveau de qualité). Dans le cas où des variables numériques sont utilisées pour décrire les états du système, il devient alors impossible de considérer l'hypothèse H1 du système fini.

Dans le cadre de la planification avec objectifs étendus et préférences, les objectifs constituent des *conditions de satisfaisabilité* que les plans solutions doivent nécessairement vérifier. Les préférences constituent quant à elles des *conditions d'optimalité* que l'utilisateur aimerait voir satisfaites si possible. En conséquence, ces dernières permettent de définir la notion de *qualité* (ou d'*utilité*) d'une solution qui peut être utilisée pour identifier les meilleurs plans parmi l'ensemble des plans valides. En effet, il est possible d'introduire une relation d'ordre $\succsim$ sur l'ensemble des plans solutions afin de comparer ces derniers au sens des préférences.

La sémantique associée à $\succsim$ est la suivante : si $x_1 \succsim x_2$, alors x_1 est au moins aussi préféré que x_2. Plusieurs méthodes peuvent être employées pour définir la relation $\succsim$ et calculer l'utilité des plans. Une méthode simpliste pourrait par exemple considérer qu'un plan x_1 est préféré à un plan x_2 s'il satisfait davantage de préférences que x_2. Ainsi, si s_n^1 et s_n^2 sont les états finaux de x_1 et x_2, alors $x_1 \succsim x_2$ si et seulement si le nombre de préférences satisfaites dans s_n^1 est plus grand que le nombre de préférences satisfaites dans s_n^2. L'une des méthodes les plus couramment utilisée (voir section 1.2.3 pour plus de détails) consiste à définir une fonction d'utilité U qui associe à chaque plan une valeur numérique. Ainsi, lorsque cette dernière doit être maximisée, x_1 est préféré à x_2 si et seulement si $U(x_1) \geq U(x_2)$. La fonction U est généralement choisie de sorte à pondérer les différentes préférences du problème en fonction de leurs importances respectives.

1.2.2 Problème de la planification avec préférences

Ces précisions au sujet des objectifs étendus permettent d'introduire formellement le problème de la planification avec préférences.

Problème de la planification avec préférences [7] *Étant donné un système à événements discrets $\Sigma = (S, A, \gamma)$, soit PT un problème de planification défini par le triplet $(\Sigma, s_0, S_{G'})$ où s_0 est un état initial et $S_{G'}$ l'ensemble des états qui vérifient les objectifs étendus G'.*

Soit X l'ensemble des solutions de PT c'est à dire l'ensemble des séquences d'actions $\langle a_1, \cdots, a_n \rangle$ qui correspondent à une séquence d'états $\langle s_0, \cdots, s_n \rangle$ telle que $s_1 = \gamma(s_0, a_1)$, $\cdots$, $s_n = \gamma(s_{n-1}, a_n)$ avec $s_n \in S_{G'}$.

Soit $\succsim$ une relation d'ordre sur X, un problème de planification avec préférences est défini par le couple $PP = (PT, \succsim)$.

L'ensemble des solutions de PP est l'ensemble X des solutions de PT. De plus, un plan $x \in X$ est une solution optimale si $\forall x' \in X,\ x \succsim x'$.

L'ensemble $\Lambda_P = \{x \in X \mid \forall x' \in X,\ x \succsim x'\}$ tel que défini précédemment est l'ensemble des solutions optimales de PT au sens de $\succsim$. La planification avec préférences constitue donc un problème d'optimisation (recherche d'une des meilleures solutions) et s'oppose en cela à la planification classique qui constitue un problème de satisfaisabilité (recherche d'une solution quelconque).

Exemple illustratif

L'exemple introduit à la section 1.1.2 est à présent enrichi afin de constituer un problème de planification avec préférences. Aux objectifs finaux [S4], [R3] et [P1] vient s'ajouter l'objectif de trajectoire [SometimeBefore R3 S4] qui s'interprète comme la contrainte que l'acquisition d'un échantillon de sol du lieu $L4$ soit réalisée avant l'acquisition d'un échantillon de roche du lieu $L3$. Le plan 1 présenté dans la section 1.1.2 ne respecte pas cette contrainte et ne constitue par conséquent pas une solution du problème nouvellement défini.

Cinq préférences notées [R2], [S6], [Sometime N1 L1], [AtMostOnce N1 L6] et [ConsommationEnergie N1] sont également ajoutées au problème initial. Les préférences [R2] et [S6] sont des préférences finales et s'interprètent respectivement comme le souhait de voir réaliser l'acquisition d'un échantillon de roche du lieu $L2$ et l'acquisition d'un échantillon de sol du lieu $L6$. Les préférences [Sometime N1 L1] et [AtMostOnce N1 L6] sont quant à elles des préférences de trajectoires qui s'interprètent respectivement comme le souhait que le robot $N1$ soit dans le lieu $L1$ dans au moins l'un des états induits par le plan et ne soit pas dans le lieu $L6$ dans plus d'un des états induit par le plan. Si la notion d'expression numérique présentée dans la section 1.1.3 est utilisée, il serait également possible de construire une préférence numérique à partir de la variable [ConsommationEnergie N1]. Cette dernière représenterait alors la satisfaction du décideur quant à la quantité d'énergie consommée par le robot $N1$ lors de l'exécution du plan.

Le plan 2 est une solution du problème puisque sa séquence d'actions est valide et qu'il vérifie les objectifs [S4], [R3], [P1] et [SometimeBefore R3 S4].

Plan 2

1. *Naviguer avec N1 de L5 vers L4*
2. *Prélever échantillon sol avec N1 en L4*
3. *Photographier L1 avec N1 en L4*
4. *Naviguer avec N1 de L4 vers L2*
5. *Prélever échantillon roche avec N1 en L2*
6. *Naviguer avec N1 de L2 vers L5*
7. *Naviguer avec N1 de L5 vers L6*
8. *Prélever échantillon sol avec N1 en L6*
9. *Naviguer avec N1 de L6 vers L3*
10. *Prélever échantillon roche avec N1 en L3*

Le plan 2 satisfait les préférences [R2], [S6] et [AtMostOnce N1 L6] mais ne satisfait pas [Sometime N1 L1] puisque le robot ne visite pas le lieu $L1$. Pour pouvoir comparer le plan 2 à d'autres plans solutions, il serait nécessaire de préciser l'expression de la relation d'ordre $\succsim$ à partir des préférences considérées.

Par ailleurs, il convient de remarquer que la construction de $\succsim$ soulève plusieurs questions. Comment représenter simultanément et de façon cohérente des préférences numériques comme la quantité d'énergie consommée et des préférences non numériques ? Comment gérer les importances relatives des différentes préférences (par exemple, si la satisfaction de [R2] seule est préférée à la satisfaction de [S6] seule) ? Comment prendre en compte les éventuelles synergies entre les préférences considérées ? Les principaux formalismes de l'art utilisés pour définir une relation de préférences $\succsim$ dans le cadre de la planification sont présentés dans la section 1.2.3.

1.2.3 Langage formel pour la planification avec préférences

Plusieurs formalismes pour raisonner sur les problèmes de planification avec préférences ont été proposés dans la littérature (voir [7] pour un état de l'art complet). Cette section présente en détails (syntaxe et sémantique) le PDDL3 [64] qui a enrichi les versions précédentes du langage par la prise en compte des préférences. Bien que le PDDL3 soit le langage le plus utilisé, les principales approches alternatives sont également présentées. Pour conclure, les différentes méthodes utilisées pour raisonner sur les préférences sont comparées les unes aux autres.

Syntaxe du langage PDDL3

Lors de la définition du PDDL3, la notation BNF du langage a été étendue pour permettre la représentation des préférences [63]. Ces modifications sont illustrées à l'aide de quelques extraits basés sur l'exemple introduit dans la section 1.2.2. La représentation complète de ce problème est disponible dans l'annexe B.3.

```
(:goal (and
  (possede_echantillon_sol N1 L4)
  (possede_echantillon_roche N1 L3)
  (possede_photographie N1 L1)

  (preference f1 (possede_echantillon_sol N1 L6))
  (preference f2 (possede_echantillon_roche N1 L2))
))
```

FIGURE 1.9 – Objectifs finaux et préférences finales en PDDL

```
(:constraints (and
  (sometime-before (possede_echantillon_roche N1 L3)
    (possede_echantillon_sol N1 L4))

  (preference s1 (sometime (position N1 L1)))
  (preference a1 (at-most-once (position N1 L6)))
))
```

FIGURE 1.10 – Objectifs et préférences de trajectoire en PDDL

Sémantique des préférences en PDDL3

Cette section enrichit la sémantique de la planification classique avec la notion de préférences. Les définitions 1.22 à 1.26 sont introduites dans ce but et sont construites par extension des définitions présentées dans la section 1.1.3.

Définition 1.22 - Instance d'un problème de planification [7]
Une instance d'un problème de planification avec préférences I_P est redéfinie par extension de la définition 1.14 en ajoutant les formules G' et SG ainsi qu'une relation d'ordre $\succsim$ à $Prob = (O, s_0, G', SG, \succsim)$ avec :

- G' une formule qui représente un ensemble d'objectifs étendus
- SG une formule qui représente un ensemble de préférences
- $\succsim$ une relation d'ordre construite à partir de SG

Dans le langage PDDL3, l'interprétation sémantique des préférences et des objectifs étendus s'appuie sur la notion de trajectoire qui s'interprète naturellement comme une succession de transitions d'états (cf. définition 1.11). Par ailleurs, elle repose sur des mécanismes analogues à ceux des logiques modales [24, 62] et s'inspire notamment de travaux de planification utilisant la logique temporelle [3].

Définition 1.23 - Valeur de vérité d'un objectif de trajectoire [64]
Étant donné un plan x et deux formules ϕ et ψ, la valeur de vérité des objectifs de trajectoire s'interprète par rapport à la trajectoire de x :

$$\langle s_0, \cdots, s_n \rangle \models \phi \quad \Leftrightarrow \quad s_n \models \phi$$

$$\langle s_0, \cdots, s_n \rangle \models (\texttt{always}\ \phi) \quad \Leftrightarrow \quad \forall i : 0 \leq i \leq n,\ s_i \models \phi$$

$$\langle s_0, \cdots, s_n \rangle \models (\texttt{sometime}\ \phi) \quad \Leftrightarrow \quad \exists i : 0 \leq i \leq n,\ s_i \models \phi$$

$$\langle s_0, \cdots, s_n \rangle \models (\texttt{at-most-once}\ \phi) \quad \Leftrightarrow \quad \forall i : 0 \leq i \leq n, \text{ si } s_i \models \phi, \text{ alors } \exists j : j \geq i, \forall k : i \leq k \leq j, s_k \models \phi \text{ et } \forall k : k > j,\ s_k \models \neg\phi$$

$$\langle s_0, \cdots, s_n \rangle \models (\texttt{sometime-before}\ \phi\ \psi) \quad \Leftrightarrow \quad \forall i : 0 \leq i \leq n, \text{ si } s_i \models \phi \text{ alors } \exists j : 0 \leq j < i,\ s_j \models \psi$$

Il existe également des opérateurs modaux temporels à l'image de `within` qui permet par exemple de préciser qu'une formule doit être vérifiée avant une date donnée. La sémantique des objectifs de trajectoire temporels est précisée dans [64] mais n'est pas reprise ici puisque ces derniers ne sont pas utilisés dans cette étude.

Les préférences étant des contraintes faibles (cf. section 1.2.1), elles sont toujours considérées comme vraies afin de ne pas impacter la validité des plans. En revanche, elles sont déterminantes pour définir l'utilité d'un plan.

Définition 1.24 - Valeur de vérité d'une préférence [64]
Etant donné un plan $\langle \hat{a}_1, \cdots, \hat{a}_n \rangle$ et une préférence $p \in SG$ définie à partir de la formule d'un objectif de trajectoire Φ, la valeur de vérité de p est toujours vraie, ce qui est dénoté par :

$$\langle s_0, \cdots, s_n \rangle \models (\texttt{preference}\ \Phi)$$

De plus, p est *satisfaite* si $\langle s_0, \cdots, s_n \rangle \models \Phi$ et *violée* dans le cas contraire.

Pour illustrer ces définitions, l'exemple introduit dans la section 1.2.2 est considéré. Soit $\langle s_0, \cdots, s_{10} \rangle$ la trajectoire correspondant au plan 2. De plus, soit ϕ_8 et ϕ_9 les deux prédicats décrits respectivement par `(position N1 L1)` et `(position N1 L6)`. Puisqu'il n'existe pas de $i \in [0, 10]$ tel que $s_i \models \phi_8$, la préférence `(sometime (position N1 L1))` est violée dans le plan 2. En revanche, $\forall i \in \{7; 8\}, s_i \models \phi_9$ et $\forall i \in [0; 6] \bigcup \{9; 10\}, s_i \models \neg\phi_9$ ce qui permet de déduire que la préférence `(at-most-once (position N1 L6))` est satisfaite par le plan 2.

Définition 1.25 - Validité d'un plan [64]
La notion de validité d'un plan $\langle \hat{a}_1, \cdots, \hat{a}_n \rangle$ est redéfinie par extension de la définition 1.13 en considérant que la formule des objectifs étendus G' doit être vraie par rapport à la trajectoire du plan à savoir $\langle s_0, \cdots, s_n \rangle \models G'$.

Définition 1.26 - Utilité d'un plan [7]
Soit X l'ensemble des plans valides de I_P et $U : X \to \mathbb{R}$ une fonction qui associe à tout plan $x \in X$ une valeur d'*utilité* sur la base des préférences de I_P. Si la fonction U doit être minimisée, alors un plan x_1 est préféré à x_2 noté $x_1 \succsim x_2$ si et seulement si $U(x_1) \leq U(x_2)$.

Domaine de la gestion de crise
En gestion de crise, les préférences PDDL3 *représentent des objectifs qu'il est souhaitable d'atteindre dans un plan solution sans que cela ne soit pour autant obligatoire. Par exemple, les décideurs peuvent préférer fournir un chauffage électrique à des victimes plutôt que de simples couvertures ou encore assurer une continuité de service complète si cela est possible (télécommunications, électricité...). Ces deux préférences peuvent être respectivement modélisées par une préférence finale et une préférence de trajectoire de type* `always`.

Autres formalismes de représentation des préférences

Le cas du PDDL3 ayant été traité, cette section s'intéresse à quelques formalismes alternatifs qui peuvent être employés dans le cadre de la planification avec préférences. Ces derniers ont tous pour objectif de définir une relation de préférence

permettant de distinguer les meilleurs plans parmi l'ensemble des plans solutions néanmoins leurs approches respectives diffèrent fortement. Ainsi, les *approches quantitatives* telles que la planification à satisfaction partielle (PSP) [130, 144], la planification basée sur la théorie de la décision [30] ou encore les approches basées sur le PDDL3 proposent d'associer une valeur numérique à chaque plan de sorte à pouvoir les comparer entre eux. Il devient alors facile d'estimer la qualité d'un plan puisqu'il suffit de calculer la valeur numérique qui lui est associée (cf. définition 1.26 par exemple). En revanche, dans le cas des *approches qualitatives*, la relation de préférence $x_1 \succsim x_2$ est définie uniquement à partir des propriétés des plans x_1 et x_2 sans qu'aucune valeur numérique ne leur soit associée. Les langages $\mathcal{PP}$ [131] et $\mathcal{LPP}$ [5, 23], les réseaux de préférences conditionnelles (CP-Nets) [29, 31] et les langages utilisant des bases de connaissances [56] sont des formalismes qualitatifs. Cette section ne s'intéresse qu'aux approches PSP, $\mathcal{LPP}$ et CP-Nets puisque ces dernières sont les alternatives au PDDL3 les plus couramment utilisées. Une description exhaustive de tous les langages utilisés dans le cadre de la planification avec préférences est disponible dans [7].

La planification à satisfaction partielle (PSP) [130, 144] étend le problème de la planification classique par l'introduction de deux fonctions U et C. Soit I_C une instance d'un problème de planification, la fonction $U : Atm_{I_C} \rightarrow \mathbb{R}^+$ associe à chaque atome de I_C une utilité positive. De même, la fonction $C : A \rightarrow \mathbb{R}^+$ définit un coût positif pour chaque action de I_C. Le bénéfice net d'un plan solution x est défini comme la différence entre la somme des utilités associées aux objectifs atteints dans x et la somme des coûts des actions de x. La relation de préférences sur l'ensemble des plans solutions d'un problème PSP est définie à partir du bénéfice net. Ainsi, un plan x_1 est préféré à un plan x_2 lorsque son bénéfice net est supérieur ou égal à celui de x_2.

Le langage $\mathcal{LPP}$ [5, 23] s'appuie sur le formalisme du *situation calculus*. Il permet d'exprimer des préférences finales ou de trajectoire avec une sémantique similaire à celle du PDDL3. Ces préférences élémentaires peuvent ensuite être composées au sein de formules de préférences atomiques. Ainsi, à partir de trois formules élémentaires ϕ_1, ϕ_2 et ϕ_3 et des niveaux de satisfaction $\{ excellent, satisfaisant, neutre \}$, il est possible de créer la formule de préférence atomique $\psi := \phi_1[excellent] \gg \phi_2[satisfaisant] \gg \phi_3[neutre]$. Le niveau de satisfaction de ψ dépend des valeurs

de vérités de ϕ_1, ϕ_2 et ϕ_3. Dans cet exemple, si ϕ_1 est satisfaite alors le niveau de satisfaction de ψ est *excellent*. En revanche, si ϕ_1 et ϕ_2 sont violées mais que ϕ_3 est satisfaite, le niveau de satisfaction associé à ψ est *neutre*. Finalement, plusieurs méthodes peuvent être employées pour agréger les formules de préférences atomiques. Certaines d'entre elles associent une valeur d'utilité aux plans ce qui confère alors une nature hybride (qualitative et quantitative) au langage.

Les réseaux de préférences conditionnels (CP-Nets) [29, 31] sont des représentations graphiques utilisées pour formaliser de façon compacte et intuitive les préférences des utilisateurs. Ils diffèrent beaucoup de l'approche PDDL3 en se focalisant sur des *préférences conditionnelles* dites *Ceteris Paribus*. De telles préférences permettent par exemple d'exprimer que *toute chose égale par ailleurs*, une formule ϕ_1 est préférée à ϕ_2 si ϕ_3 est vraie mais que ϕ_2 est préférée à ϕ_1 si ϕ_4 est vraie. En planification, elles peuvent être utilisées pour représenter des préférences entre des états objectifs. Ainsi, un plan x_1 est préféré à un plan x_2 si son état final s_n^1 est au moins aussi préféré que l'état final s_n^2 au sens du réseau de préférences conditionnelles considéré. Il convient de préciser que la relation d'ordre ainsi définie est partielle. En conséquence, il peut exister des plans qui ne peuvent être comparés avec cette relation de préférences.

Comparaison des formalismes de représentation des préférences

Cette section a pour but de comparer les formalismes de représentation de préférences précédemment listés. L'approche PDDL3/MAUT décrite en détails dans le chapitre 3 est également considérée afin de permettre au lecteur de la positionner dès à présent par rapport à l'état de l'art.

Les critères de comparaison retenus sont (i) la nature et le type de préférences qui peuvent être représentées, (ii) la méthode utilisée pour agréger ces dernières et (iii) la complexité de la construction du modèle de préférences correspondant. Une telle comparaison est nécessairement imparfaite puisque certains aspects des formalismes considérés sont par nature incomparables. Néanmoins, le prisme d'étude retenu permet de dresser un panorama général de l'emploi des préférences en planification.

En ce qui concerne la nature des préférences, les *représentations floues* qui permettent de préciser le degré de satisfaction d'une préférence sont à opposer aux *représentations binaires* qui ne peuvent exprimer que le fait qu'une préférence soit satisfaite ou non. Tous les formalismes mentionnés jusqu'à présent reposent sur la représentation binaire. En outre, plusieurs types de préférences peuvent être distinguées. Les *préférences finales*, *préférences de trajectoire* et *préférences numériques* représentent respectivement des choix du décideur qui portent sur l'état final à atteindre, les états intermédiaires du plan ou l'évolution d'une variable numérique du problème. A celles-ci viennent s'ajouter les *préférences conditionnelles* présentées dans la section précédente. De plus, le formalisme considéré peut permettre la *composition* de plusieurs préférences élémentaires afin de former des préférences plus riches à l'image des formules de préférences atomiques du langage $\mathcal{LPP}$.

Une fois que les préférences du décideur ont été spécifiées, il est nécessaire de les agréger afin de pouvoir comparer des plans entre eux. Agréger les préférences selon un *ordre lexicographique* est l'une des stratégies les plus intuitives pour les décideurs. Pour cela, les préférences sont classées par ordre de priorité $p_1, p_2, \cdots, p_n$. Le plan x_1 est alors préféré à x_2 si la valeur associée à la préférence p_1 dans x_1 est supérieure à celle de x_2. En cas d'égalité, la comparaison se poursuit avec p_2 et ce jusqu'à ce qu'un plan soit préféré à l'autre ou jusqu'à ce que toutes les préférences aient été considérées auquel cas les deux plans sont indifférenciés du point de vue du décideur. Bien que cette méthode soit très facile à mettre en œuvre, elle ne permet pas d'effectuer de compromis entre les différentes préférences du décideur. L'utilisation de méthodes basées sur une *pondération* entre les préférences du décideur permet d'adresser cette limite. La somme pondérée est généralement utilisée à cet effet mais des techniques plus génériques peuvent également être mises en œuvre ; ces dernières pouvant notamment rendre compte des éventuelles *interactions* qui existent entre les préférences du décideur.

Finalement, il est également crucial de considérer la complexité de la réalisation du modèle de préférences. Les préférences et les mécanismes d'agrégation utilisés peuvent être *simples* ou *complexes* à définir. Lorsque la construction est *manuelle*, l'utilisateur doit déterminer seul les différents paramètres du modèle de préférences utilisé. Au contraire, lorsque celle-ci est *outillée*, il peut s'appuyer sur un ensemble de méthodes scientifiques ou logicielles afin de réaliser son modèle de préférences.

	Préférence	**Agrégation**	**Modèle**
PDDL3	Binaire Finale, Trajectoire, *Numérique* Aucune composition	Pondération Pas interaction	Préférence : Simple *Agrégation : Complexe* Construction Manuelle
PDDL3 MAUT	**Floue** Finale, Trajectoire, **Numérique** **Composition**	Pondération **Interactions**	Préférence : Complexe Agrégation : Complexe **Construction Outillée**
PSP	Binaire Finale Aucune composition	Pondération Pas interaction	Préférence : Simple *Agrégation : Complexe* Construction Manuelle
$\mathcal{LPP}$	Binaire Finale, Trajectoire Composition	Ordre lexico. Pondération Pas interaction	*Préférence : Complexe* *Agrégation : Complexe* Construction Manuelle
CP-Nets	Binaire Finale, Cond. Aucune composition	Pondération Pas interaction	Préférence : Simple *Agrégation : Complexe* Construction Manuelle

TABLEAU 1.1 – Formalismes pour la planification avec préférences

Dans le tableau 1.1, les éléments de la colonne « Préférence » *en italique* indiquent que la fonctionnalité peut être considérée comme partiellement supportée. Par exemple en PDDL3, la plupart des modèles réalisés contiennent seulement une préférence numérique qui représente un coût global associé au plan. Bien que le langage permette d'utiliser un grand nombre de préférences numériques, la construction du modèle de préférences devient rapidement complexe dans de tels cas. Les éléments *en italique* de la colonne « Modèle » précisent les étapes de la construction du modèle qui peuvent être difficiles. Cette complexité est généralement liée au fait qu'il est demandé aux utilisateurs de fournir directement des paramètres numériques du modèle. Par exemple, lors de l'étape d'agrégation en PDDL3, les utilisateurs doivent renseigner les poids relatifs de leurs préférences afin de construire une somme pondérée. Il peut être très complexe pour les utilisateurs de fournir

ces informations de sorte que le modèle correspondant ait un sens opérationnel notamment quand le nombre de préférences devient important ou que ces dernières ne sont pas toutes du même type.

La ligne « PDDL3/MAUT » du tableau 1.1 correspond à une extension du langage PDDL à partir de mécanismes d'aide à la décision multicritère. Réalisée dans le cadre de cette étude et décrite en détails dans le chapitre 3, cette approche repose sur une représentation floue des préférences. Elle permet notamment de traiter le cas de plusieurs préférences numériques, de composer facilement des préférences élémentaires et de représenter des interactions entre préférences. Ce gain en pouvoir expressif augmente la complexité du modèle mais la construction de ce dernier peut être outillée de sorte à être simplifiée, voire devenir quasi-intuitive pour les utilisateurs.

1.2.4 Algorithmes de planification avec préférences

Cette section s'intéresse aux planificateurs et algorithmes utilisés pour résoudre les problèmes de planification avec préférences. La comparaison proposée s'appuie sur l'étude réalisée dans [7]. Les algorithmes retenus sont caractérisés par le formalisme de représentation des préférences et la stratégie de résolution qu'ils emploient. En outre, une attention particulière est portée aux algorithmes basés sur le langage PDDL3 puisque ces derniers peuvent être facilement comparés les uns aux autres à l'aide des problèmes de références utilisés lors des compétitions internationales de planification.

Comparaison des algorithmes de planification avec préférences

Plusieurs définitions sont introduites afin de caractériser les algorithmes de planification en fonction de leur stratégie de résolution des problèmes.

Définition 1.27 - Algorithme optimal [7]
Un algorithme A est optimal si pour tout problème $PP = (PT, \succsim)$, il retourne un plan optimal au sens de $\succsim$.

Beaucoup d'algorithmes ne sont pas conçus pour rechercher des plans de taille infinie et ne peuvent par conséquent pas être considérés comme optimaux au sens de la définition 1.27. Cependant, ils peuvent être capables de conduire à des solutions optimales pour des plans de taille donnée ce qui motive la définition 1.28. La taille d'un plan $\langle a_1, \cdots, a_n \rangle$ s'interprète comme le nombre d'actions présentes dans le plan à savoir n.

Définition 1.28 - Algorithme k-optimal [7]
Étant donné un entier positif k, un algorithme A est dit k-optimal si pour tout problème $PP = (PT, \succsim)$, il retourne un plan optimal au sens de $\succsim$ dont la taille est au plus k ; si un tel plan existe.

La recherche de solutions optimales pouvant être très complexe, une approche classique consiste à retourner une succession de plans suboptimaux de qualité croissante ; et ce jusqu'à ce qu'une solution optimale soit trouvée ou que les ressources à disposition (temps ou mémoire) soient épuisées.

Définition 1.29 - Algorithme itératif [7]
Un algorithme A est itératif s'il existe une famille non vide F de problèmes PP telle que pour tout problème $(PT, \succsim) \in F$, il retourne une séquence non vide et non singleton de plans $(x_1, x_2, \cdots)$ solutions de PP telle que :

$$\forall i, j \in \mathbb{N} \text{ tel que } 1 \leq i < j,\ x_j \succsim x_i \ \text{ et } \ x_i \not\succsim x_j.$$

Équipé de ces définitions, une comparaison des algorithmes de planification avec préférences peut être réalisée (cf. tableau 1.2). Le planificateur CHOPLAN décrit en détails dans le chapitre 3 est également considéré afin de permettre au lecteur de le comparer aux algorithmes de l'art.

Planificateur	Formalisme	Opti.	*k*-Opti.	Incrém.	Réf.
CHOPLAN	PDDL3	✓	-	✓	-
GAMER	PDDL3	✓	-	-	[51]
HPLAN-P	PDDL3	✓	✓	✓	[6]
HSP	PDDL3	✓	-	-	[78]
LPRPG-P	PDDL3	✓	-	✓	[39]
MIPS-BDD	PDDL3	✓	-	-	[48]
MIPS-XXL	PDDL3	✓	-	✓	[50]
SGPlan5	PDDL3	-	-	-	[83]
$Yochan^{PS}$	PDDL3	-	-	✓	[20]
BBOP-LP	PSP	✓	-	✓	[21]
$Sapa^{PS}$	PSP	-	-	✓	[144]
HPLAN-QP	$\mathcal{LPP}$	✓	✓	✓	[5]
PPLAN	$\mathcal{LPP}$	-	✓	-	[23]
PREFPLAN	CP-Nets	-	✓	-	[31]

TABLEAU 1.2 – Algorithmes pour la planification avec préférences [7]

Comparaison des planificateurs basés sur le langage PDDL3

Des compétitions sont organisées tous les deux ou trois ans par la communauté scientifique de la planification. Celles-ci permettent de comparer différents algorithmes de planification entre eux sur la base d'un ensemble de problèmes de références formalisés en PDDL. La 5$^{\text{ème}}$ compétition internationale de planification (IPC5) [65] qui s'est déroulée en 2006 a introduit la troisième version du PDDL ainsi que plusieurs domaines et problèmes pour la planification avec préférences. Pour résoudre les problèmes proposés lors de IPC5, les planificateurs devaient être capables de supporter tout ou partie de la spécification du PDDL3. En outre, des problèmes de planification avec préférences ont également été utilisés en 2008 au cours de la 6$^{\text{ème}}$ compétition internationale de planification [43]. Néanmoins, ces derniers ne sont pas aussi riches que ceux retenus pour IPC5 puisqu'ils se

limitent à l'utilisation de préférences finales. Enfin, par manque de participants, les problématiques de la planification avec préférences n'ont pas été adressées lors des $7^{\text{ème}}$ et $8^{\text{ème}}$ éditions de la compétition internationale de planification.

Le tableau 1.3 s'intéresse aux différents planificateurs de l'art basés sur le langage PDDL3. Il précise leur support des fonctionnalités relatives aux préférences du langage PDDL ainsi que leurs éventuelles participations aux compétitions internationales de planification. Il convient de remarquer qu'aucune distinction n'a été faite entre les planificateurs qui supportent l'intégralité des préférences de trajectoire et ceux qui ne sont capables de gérer que les préférences de trajectoire non temporelles à l'image de CHOPLAN par exemple.

Planificateur	**OT**	**PF**	**PT**	**PN**	**IPC**
HPLAN-P	-	✓	✓	-	$5^{\text{ème}}$
MIPS-BDD	-	✓	✓	-	$5^{\text{ème}}$
MIPS-XXL	✓	✓	✓	✓	$5/6^{\text{ème}}$
SGPlan5	✓	✓	✓	✓	$5^{\text{ème}}$
Yochan^{PS}	-	✓	-	-	$5^{\text{ème}}$
GAMER	-	✓	-	✓	$6^{\text{ème}}$
HSP	-	✓	-	✓	$6^{\text{ème}}$
LPRPG-P	✓	✓	✓	✓	-
CHOPLAN	✓	✓	✓	✓	-

TABLEAU 1.3 – Participation des planificateurs aux compétitions internationales (IPC) et support du langage PDDL3. « OT » signifie objectifs de trajectoire, « PF » signifie préférences finales, « PT » signifie préférences de trajectoire et « PN » signifie préférences numériques.

Conclusion

Ce chapitre a débuté par la description du modèle conceptuel de la planification qui formalise l'activité de la planification automatisée. Ce dernier propose un ensemble de huit hypothèses fondamentales qui, lorsqu'elles sont considérées simultanément, définissent la classe des problèmes dits de planification classique. De plus, le langage formel PDDL qui est utilisé pour représenter les problèmes de planification a été introduit. Une grande attention a été portée à la sémantique de ce dernier puisqu'elle permet de préciser les mécanismes et concepts employés pour résoudre les problèmes de planification. Par la suite, la notion de préférences a été présentée afin de décrire la problématique de la planification avec préférences qui est au cœur de cette étude. Les différents formalismes de représentation de préférences utilisés en planification ont été exposés en détails. Ces derniers sont fondamentaux puisqu'ils influent sur le type de problèmes qui peuvent être résolus ainsi que sur les méthodes et techniques employées pour les résoudre. Pour finir, les principaux algorithmes de planification avec préférences de l'art ont été présentés.

Le chapitre 2 introduit un formalisme de modélisation de préférences utilisé en aide à la décision multicritère et encore jamais employé dans le cadre de la planification. Celui-ci sera utilisé dans le chapitre 3 pour étendre le pouvoir expressif du langage PDDL3 ainsi que pour créer un algorithme original de résolution des problèmes de planification avec préférences (CHOPLAN).