Un survol de la planification

Cyril Terrioux

Laboratoire des Sciences de l'Information et des Systèmes

LSIS - UMR CNRS 6168

Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

Qu’est-ce que la planification ?

Planification = action de planifier, d’organiser selon un plan déterminé

Qu’est-ce que la planification ?

Planification = action de planifier, d’organiser selon un plan déterminé

Plan = succession d’actions

Qu’est-ce que la planification ?

Planification = action de planifier, d’organiser selon un plan déterminé

Plan = succession d’actions

Plan-solution = succession d’actions menant d’un état initial à un état final

Qu’est-ce que la planification ?

Planification = action de planifier, d’organiser selon un plan déterminé

Plan = succession d’actions

Plan-solution = succession d’actions menant d’un état initial à un état final

Exécution = réalisation d’actions effectuées conformément à un plan donné.

Plusieurs tâches possibles

- Déterminer s’il existe un plan-solution.

Plusieurs tâches possibles

- Déterminer s’il existe un plan-solution.
- Trouver un plan-solution.

Plusieurs tâches possibles

- Déterminer s’il existe un plan-solution.
- Trouver un plan-solution.
- Trouver tous les plans-solutions.

Plusieurs tâches possibles

- Déterminer s’il existe un plan-solution.
- Trouver un plan-solution.
- Trouver tous les plans-solutions.
- Trouver un plan-solution vérifiant une propriété donnée.

Plusieurs tâches possibles

- Déterminer s’il existe un plan-solution.
- Trouver un plan-solution.
- Trouver tous les plans-solutions.
- Trouver un plan-solution vérifiant une propriété donnée.
- . . .

Plusieurs points de vue possibles

- la planification comme réutilisation :

  on essaye de synthétiser un plan-solution en exploitant des plans déjà générés.

Plusieurs points de vue possibles

- la planification comme réutilisation :

  on essaye de synthétiser un plan-solution en exploitant des plans déjà générés.

- la planification comme expertise :

  on essaye de synthétiser un plan-solution en exploitant des connaissances fournies par des experts.

Plusieurs points de vue possibles

- la planification comme réutilisation :

  on essaye de synthétiser un plan-solution en exploitant des plans déjà générés.

- la planification comme expertise :

  on essaye de synthétiser un plan-solution en exploitant des connaissances fournies par des experts.

- la planification comme synthèse :

  cette approche a pour but la construction de systèmes généraux afin de résoudre des problèmes diverses.

La planification comme synthèse

Elle requiert :

- une représentation du problème à résoudre en donnant :
  - l'état initial
  - l'état final
  - un ensemble d'opérateurs

La planification comme synthèse

Elle requiert :

- une représentation du problème à résoudre en donnant :
  - l'état initial
  - l'état final
  - un ensemble d'opérateurs
- un algorithme pour produire un plan-solution

La planification comme synthèse

Elle requiert :

- une représentation du problème à résoudre en donnant :
  - l'état initial
  - l'état final
  - un ensemble d'opérateurs
- un algorithme pour produire un plan-solution
- des heuristiques pour guider la recherche d'un plan-solution.

Un bref historique de la planification

- Les années 50 : les débuts de la planification

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS
- 1971 : STRIPS

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS
- 1971 : STRIPS
- Les années 90 : le renouveau et l'essor de la planification

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS
- 1971 : STRIPS
- Les années 90 : le renouveau et l'essor de la planification
- 1992 : SATPLAN

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS
- 1971 : STRIPS
- Les années 90 : le renouveau et l'essor de la planification
- 1992 : SATPLAN
- 1995 : GRAPHPLAN

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS
- 1971 : STRIPS
- Les années 90 : le renouveau et l'essor de la planification
- 1992 : SATPLAN
- 1995 : GRAPHPLAN
- 1999 : BLACKBOX (SAT)

Un bref historique de la planification

- Les années 50 : les débuts de la planification
- 1957 : GPS
- 1971 : STRIPS
- Les années 90 : le renouveau et l'essor de la planification
- 1992 : SATPLAN
- 1995 : GRAPHPLAN
- 1999 : BLACKBOX (SAT)
- 1999-2001 : utilisation des techniques CSP

Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

Hypothèses de travail

- l’univers est statique

Hypothèses de travail

- l’univers est statique
- l’agent est omniscient

Hypothèses de travail

- l’univers est statique
- l’agent est omniscient
- les actions ont des effets déterminés et connus

Hypothèses de travail

- l’univers est statique
- l’agent est omniscient
- les actions ont des effets déterminés et connus
- les actions sont atomiques

Le cadre STRIPS

Il repose sur la logique du premier ordre.

fluent = formule atomique

état = ensemble de fluent

Le cadre STRIPS

opérateur / action = (préconditions, ajouts, retraits) :

Le cadre STRIPS

opérateur / action = (préconditions, ajouts, retraits) :

- préconditions = ensemble de fluents requis pour pouvoir appliquer l'action

Le cadre STRIPS

opérateur / action = (préconditions, ajouts, retraits) :

- préconditions = ensemble de fluents requis pour pouvoir appliquer l'action
- ajouts = ensemble de fluents ajoutés à l'état courant par l'application de l'action

Le cadre STRIPS

opérateur / action = (préconditions, ajouts, retraits) :

- préconditions = ensemble de fluents requis pour pouvoir appliquer l'action
- ajouts = ensemble de fluents ajoutés à l'état courant par l'application de l'action
- retraits = ensemble de fluents retirés de l'état courant par l'application de l'action

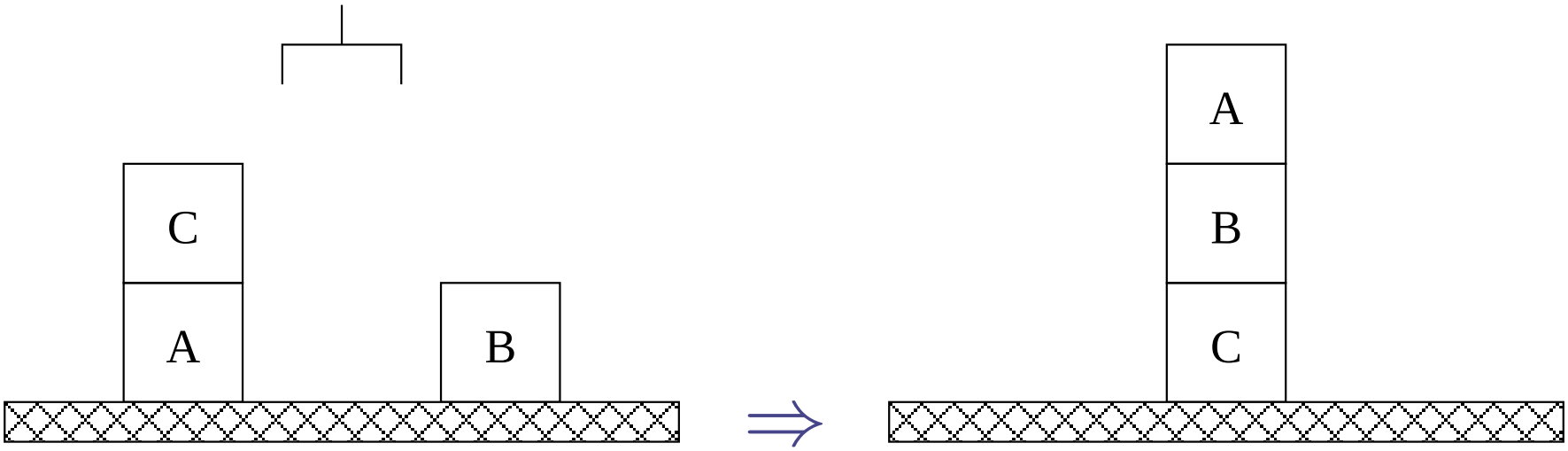
Le cadre STRIPS

opérateur / action = (préconditions, ajouts, retraits) :

- préconditions = ensemble de fluents requis pour pouvoir appliquer l'action
- ajouts = ensemble de fluents ajoutés à l'état courant par l'application de l'action
- retraits = ensemble de fluents retirés de l'état courant par l'application de l'action

Tout fluent non modifié par l'action reste présent dans l'état obtenu.

Exemple : le monde des cubes



état initial

état final

Exemple : le monde des cubes

5 prédicats :

- Découvert(X) est vrai si aucun cube n’est posé sur le cube X

Exemple : le monde des cubes

5 prédicats :

- Découvert(X) est vrai si aucun cube n'est posé sur le cube X
- Sur(X,Y) est vrai si le cube X est posé sur le cube Y

Exemple : le monde des cubes

5 prédicats :

- Découvert(X) est vrai si aucun cube n'est posé sur le cube X
- Sur(X,Y) est vrai si le cube X est posé sur le cube Y
- SurTable(X) est vrai si le cube X est posé sur la table

Exemple : le monde des cubes

5 prédicats :

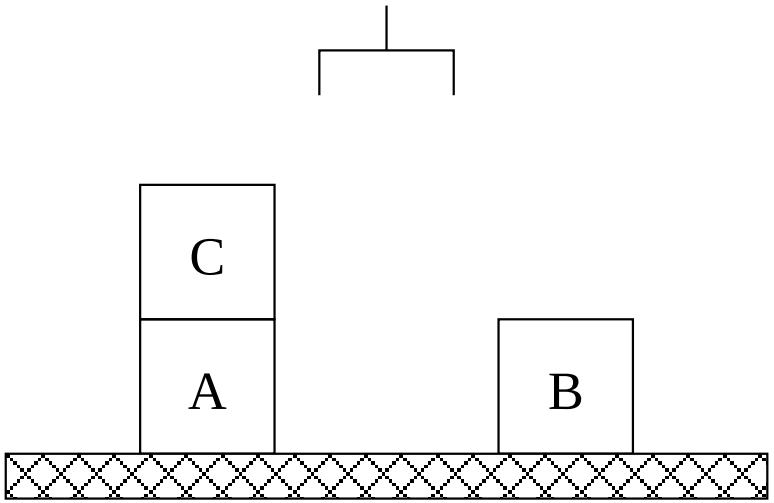
- Découvert(X) est vrai si aucun cube n'est posé sur le cube X
- Sur(X,Y) est vrai si le cube X est posé sur le cube Y
- SurTable(X) est vrai si le cube X est posé sur la table
- MainVide est vrai si la main du robot est vide

Exemple : le monde des cubes

5 prédicats :

- Découvert(X) est vrai si aucun cube n'est posé sur le cube X
- Sur(X,Y) est vrai si le cube X est posé sur le cube Y
- SurTable(X) est vrai si le cube X est posé sur la table
- MainVide est vrai si la main du robot est vide
- Tenu(X) est vrai si la main du robot contient le cube X

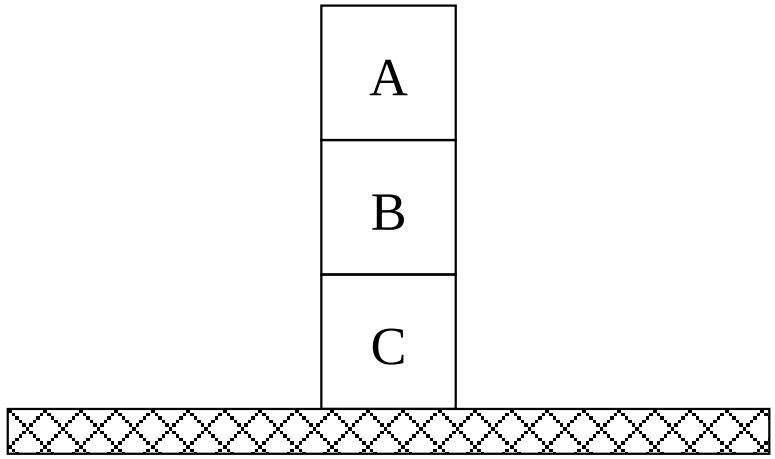
Exemple : le monde des cubes



Découvert(B)
Découvert(C)
Sur(C,A)

SurTable(A)
SurTable(B)
MainVide

Exemple : le monde des cubes



Sur (A,B) SurTable(C)

Sur (B,C)

Exemple : le monde des cubes

Les opérateurs :

- Ramasser(X) : prendre le cube X qui est sur la table

Exemple : le monde des cubes

Les opérateurs :

- Ramasser(X) : prendre le cube X qui est sur la table
- Poser(X) : poser le cube X sur la table

Exemple : le monde des cubes

Les opérateurs :

- Ramasser(X) : prendre le cube X qui est sur la table
- Poser(X) : poser le cube X sur la table
- Dépiler(X,Y) : prendre le cube X qui est sur le cube Y

Exemple : le monde des cubes

Les opérateurs :

- Ramasser(X) : prendre le cube X qui est sur la table
- Poser(X) : poser le cube X sur la table
- Dépiler(X,Y) : prendre le cube X qui est sur le cube Y
- Empiler(X,Y) : poser le cube X sur le cube Y

Exemple : le monde des cubes

L'opérateur Ramasser(X) :

- Préconditions : MainVide, Découvert(X), SurTable(X)

Exemple : le monde des cubes

L'opérateur Ramasser(X) :

- Préconditions : MainVide, Découvert(X), SurTable(X)
- Ajouts : Tenu(X)

Exemple : le monde des cubes

L'opérateur Ramasser(X) :

- Préconditions : MainVide, Découvert(X), SurTable(X)
- Ajouts : Tenu(X)
- Retraits : MainVide, Découvert(X), SurTable(X)

Exemple : le monde des cubes

L'opérateur Empiler(X,Y) :

- Préconditions : Tenu(X), Découvert(Y)

Exemple : le monde des cubes

L'opérateur Empiler(X,Y) :

- Préconditions : Tenu(X), Découvert(Y)
- Ajouts : MainVide, Sur(X,Y), Découvert(X)

Exemple : le monde des cubes

L'opérateur Empiler(X,Y) :

- Préconditions : Tenu(X), Découvert(Y)
- Ajouts : MainVide, Sur(X,Y), Découvert(X)
- Retraits : Tenu(X), Découvert(Y)

Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

Représentation

état = ensemble fini de fluents

Représentation

état = ensemble fini de fluents

Un état est dit complet s'il décrit totalement le monde

Représentation

état = ensemble fini de fluents

Un état est dit complet s'il décrit totalement le monde (c'est-à-dire quand on assigne une valeur de vérité à chaque fluent).

Représentation

état = ensemble fini de fluents

Un état est dit complet s'il décrit totalement le monde (c'est-à-dire quand on assigne une valeur de vérité à chaque fluent).

Il est dit partiel sinon.

Représentation

état = ensemble fini de fluents

Un état est dit complet s'il décrit totalement le monde (c'est-à-dire quand on assigne une valeur de vérité à chaque fluent).

Il est dit partiel sinon.

opérateur = (pre,add,del)

Représentation

état = ensemble fini de fluents

Un état est dit complet s'il décrit totalement le monde (c'est-à-dire quand on assigne une valeur de vérité à chaque fluent).

Il est dit partiel sinon.

opérateur = (pre,add,del)

action = opérateur appliqué pour une constante donnée

Problème de planification = (O,I,F)

- O est un ensemble fini d’opérateurs

Problème de planification = (O,I,F)

- O est un ensemble fini d’opérateurs
- I est un état complet représentant l’état initial

Problème de planification = (O,I,F)

- O est un ensemble fini d'opérateurs
- I est un état complet représentant l'état initial
- F est un état représentant un état final

Problème de planification = (O,I,F)

- O est un ensemble fini d'opérateurs
- I est un état complet représentant l'état initial
- F est un état représentant un état final

F est souvent un état partiel

Application d'une action à un état

Soit E un état.

Soit une action $a = (pre(a), add(a), del(a))$

Application d'une action à un état

Soit E un état.

Soit une action $a = (pre(a), add(a), del(a))$

On applique l'action a ssi $pre(a) \subseteq E$

Application d'une action à un état

Soit E un état.

Soit une action $a = (pre(a), add(a), del(a))$

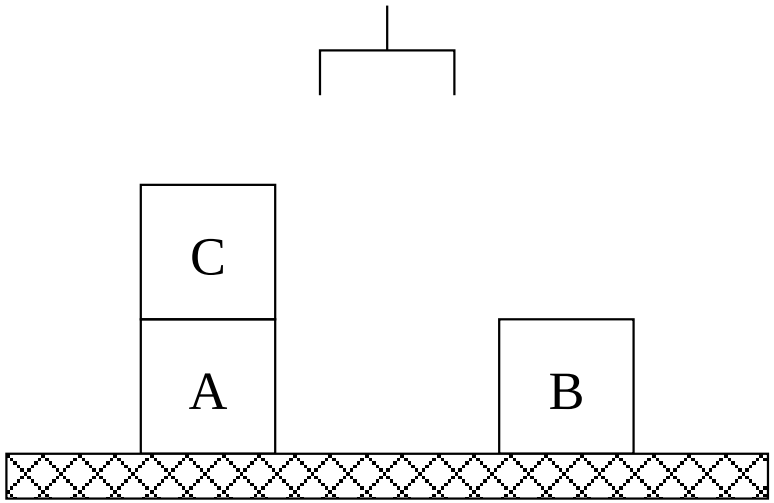
On applique l'action a ssi $pre(a) \subseteq E$

On obtient alors $E' = (E \backslash del(a)) \cup add(a)$

Exemple : le monde des cubes

Découvert(B)
Découvert(C)
Sur(C,A)

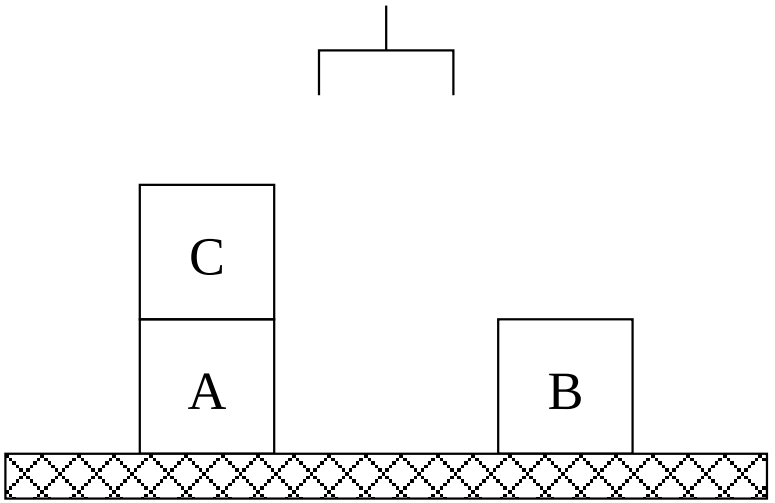
SurTable(A)
SurTable(B)
MainVide



Exemple : le monde des cubes

Découvert(B)
Découvert(C)
Sur(C,A)

SurTable(A)
SurTable(B)
MainVide



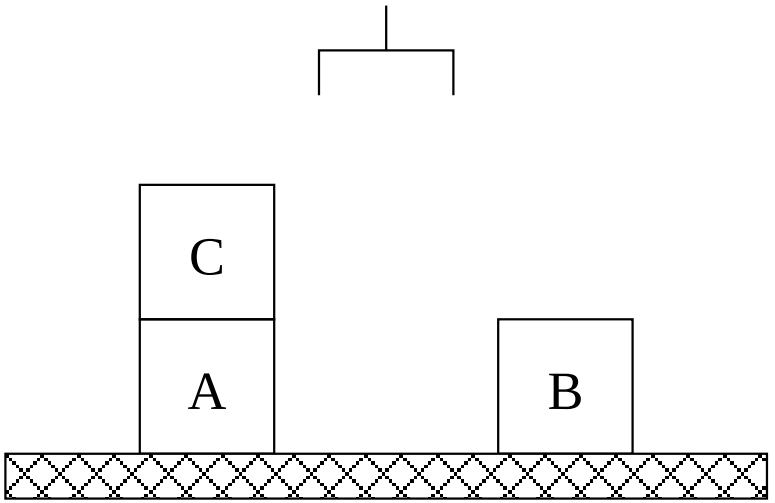
L'opérateur Ramasser(X) :

- Préconditions : MainVide, Découvert(X), SurTable(X)
- Ajouts : Tenu(X)
- Retraits : MainVide, Découvert(X), SurTable(X)

Exemple : le monde des cubes

Découvert(B)
Découvert(C)
Sur(C,A)

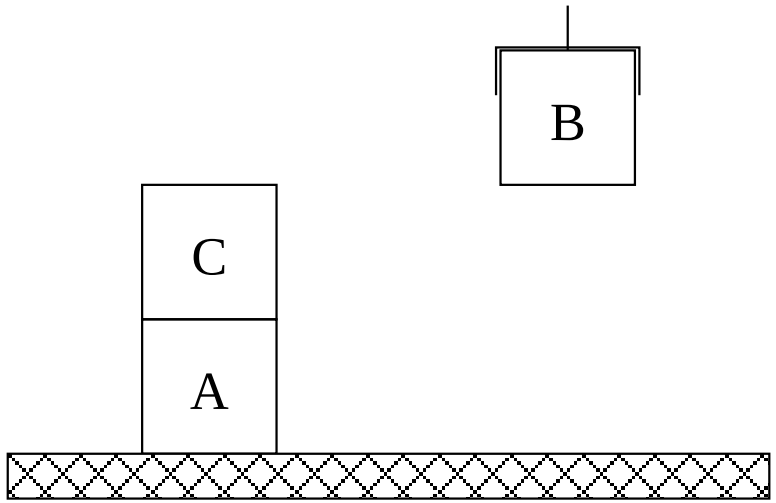
SurTable(A)
SurTable(B)
MainVide



L'action Ramasser(B) :

- Préconditions : MainVide, Découvert(B), SurTable(B)
- Ajouts : Tenu(B)
- Retraits : MainVide, Découvert(B), SurTable(B)

Exemple : le monde des cubes



Tenu(B)

Découvert(C)

SurTable(A)

Sur(C,A)

Régression

Soit E un état.

Soit une action $a = (pre(a), add(a), del(a))$

Régression

Soit E un état.

Soit une action $a = (pre(a), add(a), del(a))$

On peut régresser en utilisant l'action a ssi

- $add(a) \cap E \neq \emptyset$: a permet de produire des fluents de E

Régression

Soit E un état.

Soit une action $a = (pre(a), add(a), del(a))$

On peut régresser en utilisant l'action a ssi

- $add(a) \cap E \neq \emptyset$: a permet de produire des fluents de E
- $del(a) \cap E = \emptyset$: on ne retire aucun fluent de E

Régression

Soit E un état.

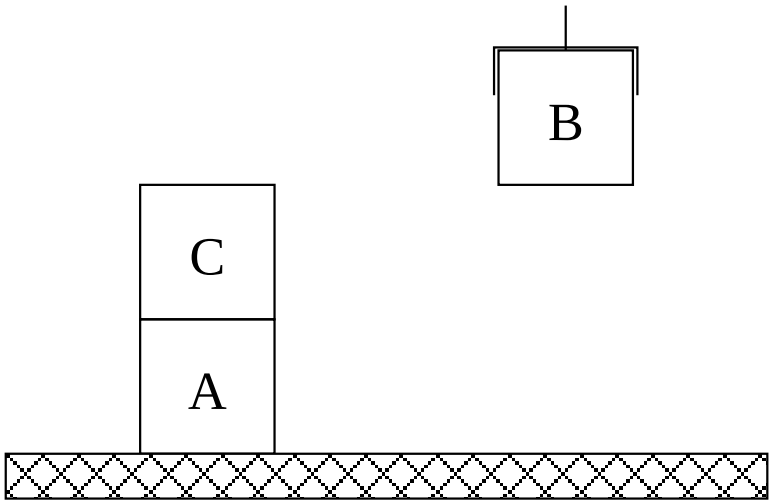
Soit une action $a = (pre(a), add(a), del(a))$

On peut régresser en utilisant l'action a ssi

- $add(a) \cap E \neq \emptyset$: a permet de produire des fluents de E
- $del(a) \cap E = \emptyset$: on ne retire aucun fluent de E

On obtient alors $E' = (E \backslash add(a)) \cup pre(a)$

Exemple : le monde des cubes



Tenu(B)
Découvert(C)
SurTable(A)
Sur(C,A)

On peut régresser en utilisant l'action Ramasser(B)

La résolution via du chaînage avant

Recherche classique dans les graphes d'états :

- nœud = un état complet

La résolution via du chaînage avant

Recherche classique dans les graphes d'états :

- nœud = un état complet
- nœud racine = l'état initial

La résolution via du chaînage avant

Recherche classique dans les graphes d'états :

- nœud = un état complet
- nœud racine = l'état initial
- nœud solution = un état final

La résolution via du chaînage avant

Recherche classique dans les graphes d'états :

- nœud = un état complet
- nœud racine = l'état initial
- nœud solution = un état final
- on passe d'un nœud à un autre par l'application d'une action

La résolution via du chaînage avant

Recherche classique dans les graphes d'états :

- nœud = un état complet
- nœud racine = l'état initial
- nœud solution = un état final
- on passe d'un nœud à un autre par l'application d'une action
- les arcs sont étiquetés par les actions utilisées.

La résolution via du chaînage avant

- recherche en profondeur d’abord,
- recherche en largeur d’abord,
- recherche le meilleur d’abord,
- A*,
- Iterative Deepening
- méthodes incomplètes,
- . . .

La résolution via du chaînage arrière

Recherche classique dans les graphes d'états :

- nœud = un état souvent partiel

La résolution via du chaînage arrière

Recherche classique dans les graphes d'états :

- nœud = un état souvent partiel
- nœud racine = un état final

La résolution via du chaînage arrière

Recherche classique dans les graphes d'états :

- nœud = un état souvent partiel
- nœud racine = un état final
- nœud solution = état partiel contenu dans l'état initial

La résolution via du chaînage arrière

Recherche classique dans les graphes d'états :

- nœud = un état souvent partiel
- nœud racine = un état final
- nœud solution = état partiel contenu dans l'état initial
- on passe d'un nœud à un autre par régression par une action

La résolution via du chaînage arrière

Recherche classique dans les graphes d'états :

- nœud = un état souvent partiel
- nœud racine = un état final
- nœud solution = état partiel contenu dans l'état initial
- on passe d'un nœud à un autre par régression par une action
- les arcs sont étiquetés par les actions utilisées.

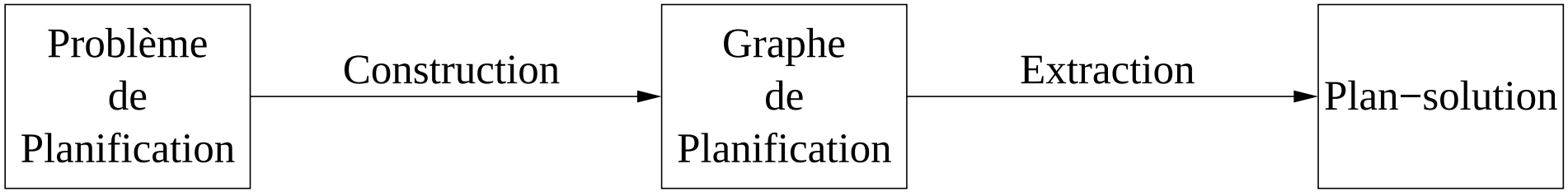
Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

Le Principe

Construction du graphe de planification

Recherche d'un plan-solution



Construction du graphe de planification

Une construction niveau par niveau ($\approx$ Iterative Deepening)

Construction du graphe de planification

Une construction niveau par niveau ($\approx$ Iterative Deepening)

On ajoute un nouveau niveau jusqu'à trouver un plan-solution

Construction du graphe de planification

Une construction niveau par niveau ($\approx$ Iterative Deepening)

On ajoute un nouveau niveau jusqu'à trouver un plan-solution

Le premier niveau contient les fluents de l'état initial

Construction du graphe de planification

Une construction niveau par niveau ($\approx$ Iterative Deepening)

On ajoute un nouveau niveau jusqu'à trouver un plan-solution

Le premier niveau contient les fluents de l'état initial

Chaque autre niveau contient deux types de nœuds :

- nœud d'action

Construction du graphe de planification

Une construction niveau par niveau ($\approx$ Iterative Deepening)

On ajoute un nouveau niveau jusqu'à trouver un plan-solution

Le premier niveau contient les fluents de l'état initial

Chaque autre niveau contient deux types de nœuds :

- nœud d'action
- nœud de fluent

Construction du graphe de planification

Trois types d’arc :

- les arcs allant d’un nœud de fluent à un nœud d’action (préconditions)

Construction du graphe de planification

Trois types d'arc :

- les arcs allant d'un nœud de fluent à un nœud d'action (préconditions)
- les arcs allant d'un nœud d'action à un nœud de fluent (retrait)

Construction du graphe de planification

Trois types d'arc :

- les arcs allant d'un nœud de fluent à un nœud d'action (préconditions)
- les arcs allant d'un nœud d'action à un nœud de fluent (retrait)
- les arcs allant d'un nœud d'action à un nœud de fluent (ajout)

Construction du graphe de planification

Trois types d'arc :

- les arcs allant d'un nœud de fluent à un nœud d'action (préconditions)
- les arcs allant d'un nœud d'action à un nœud de fluent (retrait)
- les arcs allant d'un nœud d'action à un nœud de fluent (ajout)

Calcul et mémorisation de contraintes d'exclusions mutuelles entre actions ou entre fluents

Construction du graphe de planification

Trois types d'arc :

- les arcs allant d'un nœud de fluent à un nœud d'action (préconditions)
- les arcs allant d'un nœud d'action à un nœud de fluent (retrait)
- les arcs allant d'un nœud d'action à un nœud de fluent (ajout)

Calcul et mémorisation de contraintes d'exclusions mutuelles entre actions ou entre fluents

Un calcul réalisable en temps et en espace polynomial !

Exemple

Prédicats : a,b,c,d

Etat initial : a

Etat final : d

Opérateurs :

- A = ({a},{b},∅)
- B = ({a},{c},{a})
- C = ({b,c},{d},∅)

Exemple

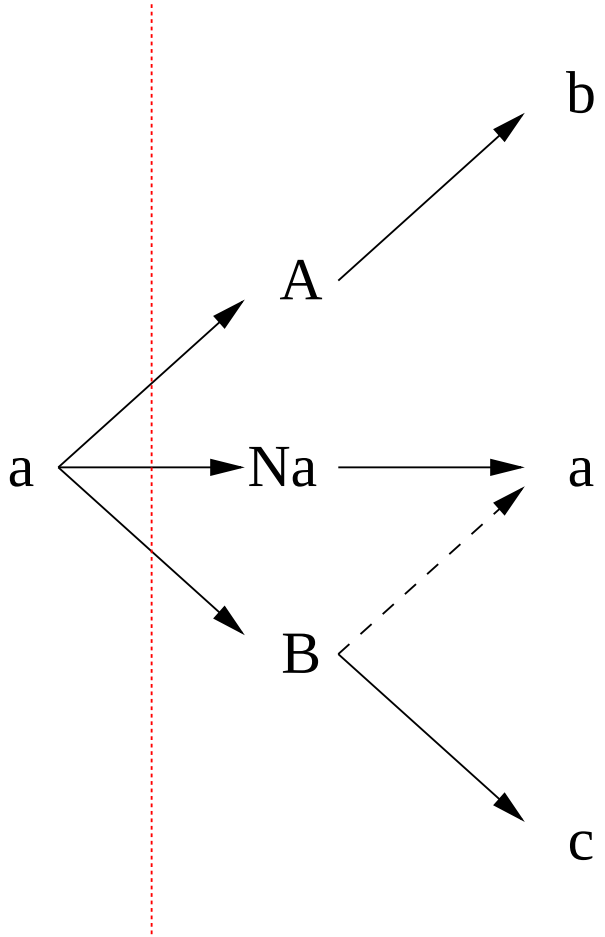
Prédicats : a,b,c,d

Etat initial : a

Etat final : d

Opérateurs :

- A = ({a},{b},∅)
- B = ({a},{c},{a})
- C = ({b,c},{d},∅)



Exemple

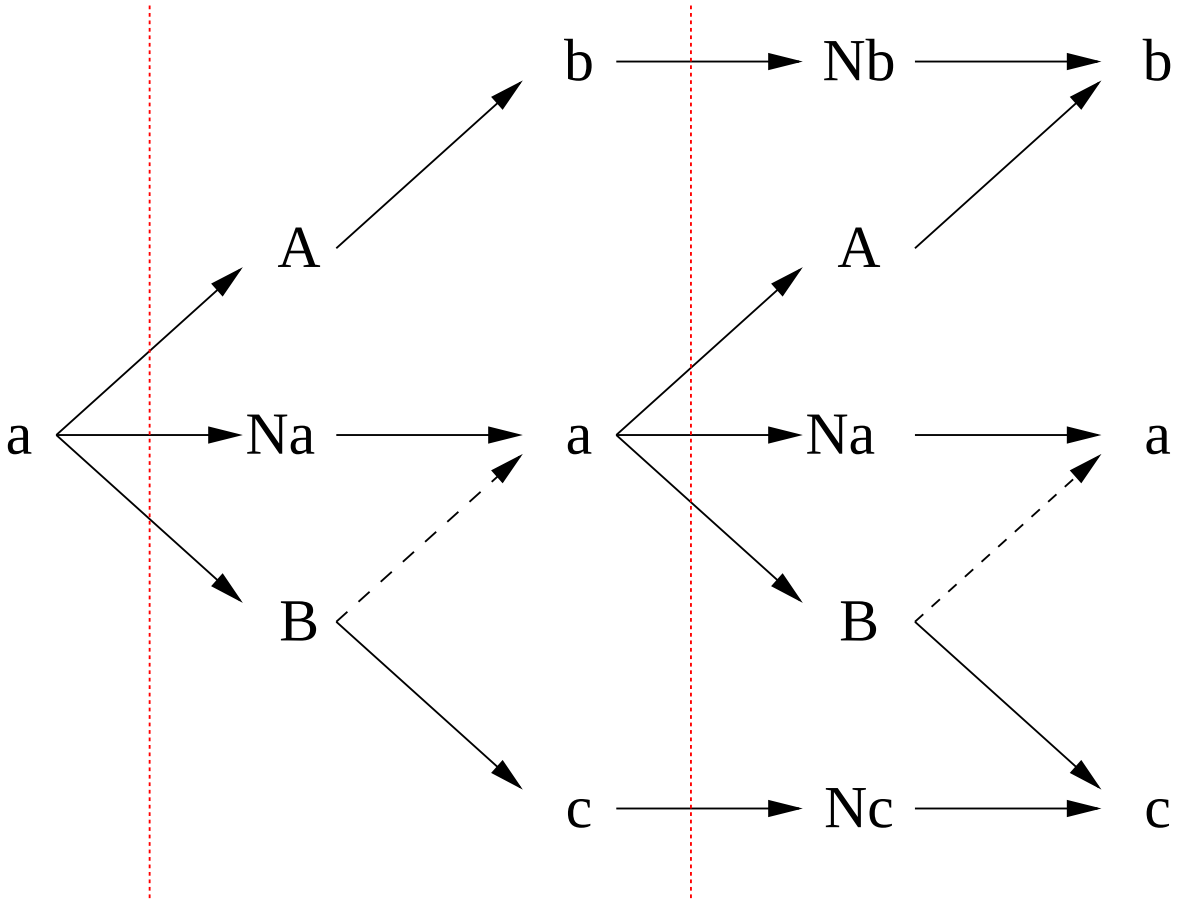
Prédicats : a,b,c,d

Etat initial : a

Etat final : d

Opérateurs :

- A = ({a},{b},∅)
- B = ({a},{c},{a})
- C = ({b,c},{d},∅)



L’extraction de solution

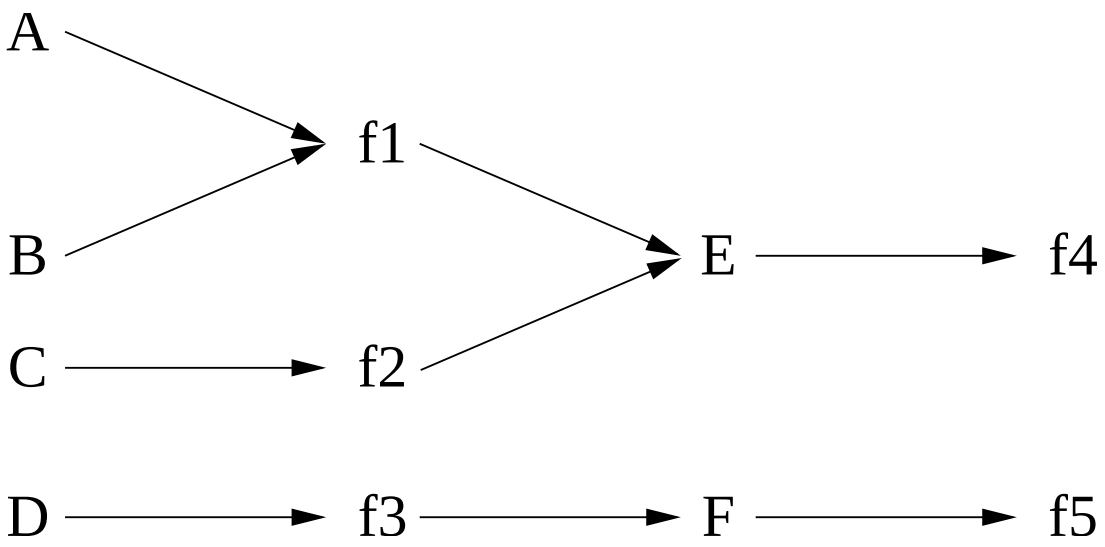
Les contraintes d’exclusions mutuelles ne sont pas suffisantes pour garantir la validité d’un plan.

Elles ne prennent pas en compte tous les cas.

L’extraction de solution

Les contraintes d’exclusions mutuelles ne sont pas suffisantes pour garantir la validité d’un plan.

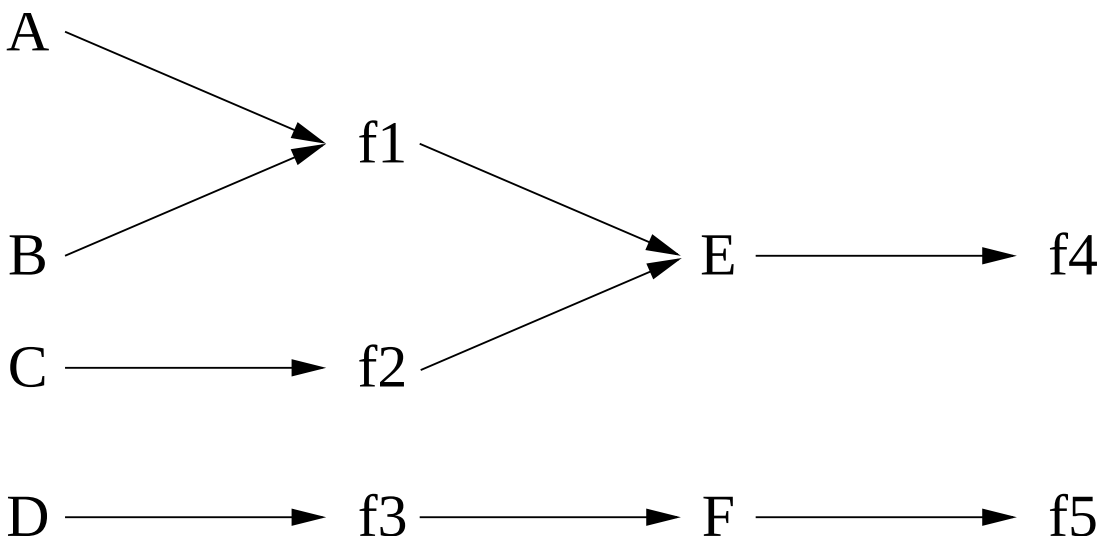
Elles ne prennent pas en compte tous les cas.



L’extraction de solution

Les contraintes d’exclusions mutuelles ne sont pas suffisantes pour garantir la validité d’un plan.

Elles ne prennent pas en compte tous les cas.

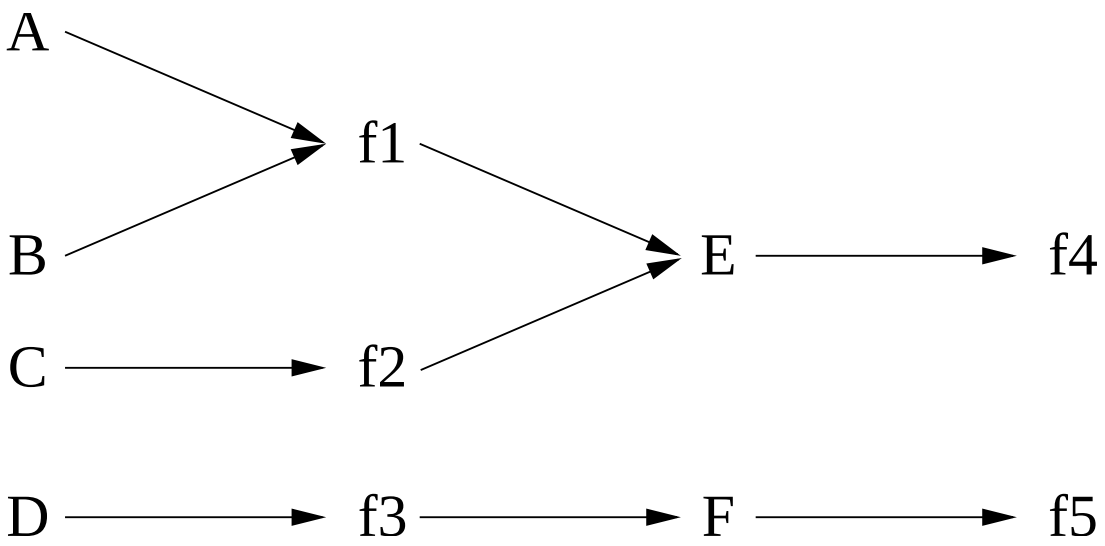


A et C ainsi que B et D sont mutuellement exclusifs

L’extraction de solution

Les contraintes d’exclusions mutuelles ne sont pas suffisantes pour garantir la validité d’un plan.

Elles ne prennent pas en compte tous les cas.



D’où la nécessité de la phase d’extraction de solution

L’extraction de solution

On recherche un plan-solution en partant de l’état final.

L’extraction de solution

On recherche un plan-solution en partant de l’état final.

Recherche dans un arbre ET/OU :

- la racine est l’état final

L’extraction de solution

On recherche un plan-solution en partant de l’état final.

Recherche dans un arbre ET/OU :

- la racine est l’état final
- les nœuds fluents sont des nœuds OU

L’extraction de solution

On recherche un plan-solution en partant de l’état final.

Recherche dans un arbre ET/OU :

- la racine est l’état final
- les nœuds fluents sont des nœuds OU
- les nœuds actions sont des nœuds ET

L’extraction de solution

On recherche un plan-solution en partant de l’état final.

Recherche dans un arbre ET/OU :

- la racine est l’état final
- les nœuds fluents sont des nœuds OU
- les nœuds actions sont des nœuds ET
- les fluents de l’état initial sont les feuilles de l’arbre

L’extraction de solution

On recherche un plan-solution en partant de l’état final.

Recherche dans un arbre ET/OU :

- la racine est l’état final
- les nœuds fluents sont des nœuds OU
- les nœuds actions sont des nœuds ET
- les fluents de l’état initial sont les feuilles de l’arbre

Si on ne trouve pas de plan-solution, on étend le graphe d’un niveau.

Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

L’idée

Exploiter la constante évolution des solveurs SAT.

L’idée

Exploiter la constante évolution des solveurs SAT.

2 principaux axes coexistent :

- coder directement le problème de planification en SAT

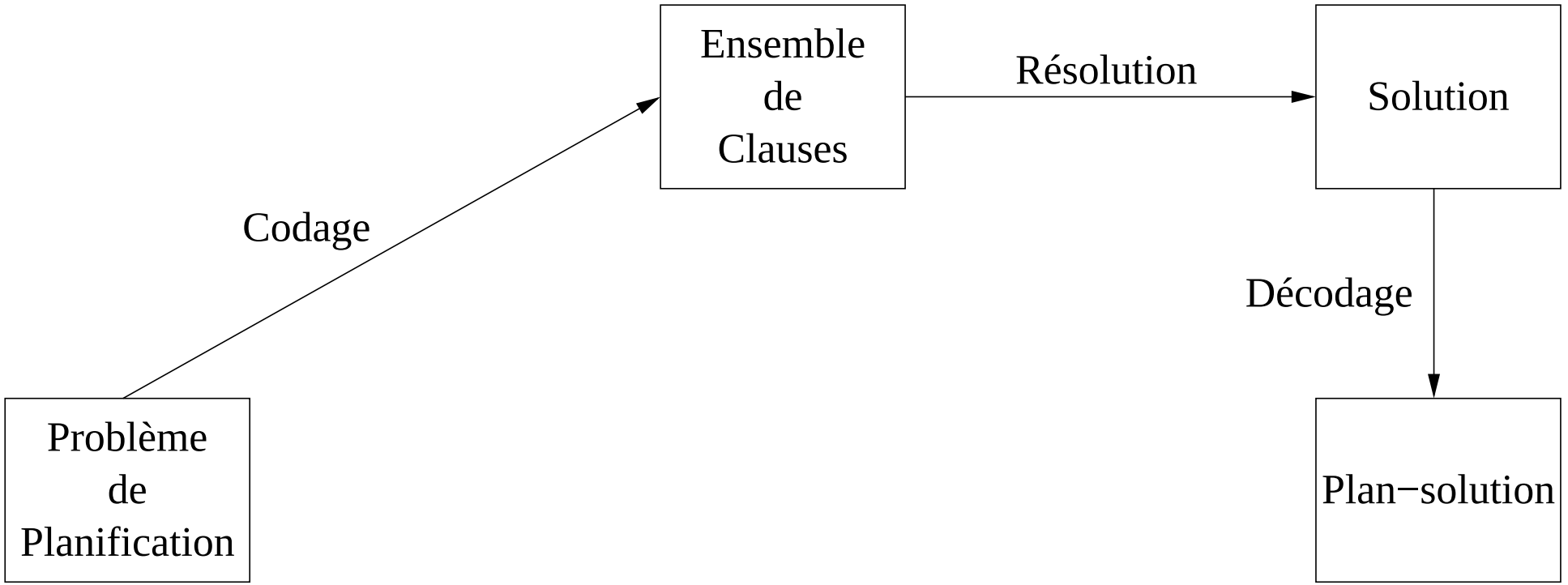
L’idée

Exploiter la constante évolution des solveurs SAT.

2 principaux axes coexistent :

- coder directement le problème de planification en SAT
- utiliser SAT pour l’extraction de solution dans une approche de type GRAPHPLAN

Le codage direct



Le codage direct

La longueur d’un plan n’est pas connue initialement.

Le codage direct

La longueur d’un plan n’est pas connue initialement.

Aussi, on procède par étape (comme Iterative Deepening).

Le codage direct

La longueur d’un plan n’est pas connue initialement.

Aussi, on procède par étape (comme Iterative Deepening).

Une même action peut être apparaître plusieurs fois.

Le codage direct

La longueur d’un plan n’est pas connue initialement.

Aussi, on procède par étape (comme Iterative Deepening).

Une même action peut être apparaître plusieurs fois.

Elle peut donc être représentée par plusieurs variables propositionnelles.

Le codage direct

La longueur d'un plan n'est pas connue initialement.

Aussi, on procède par étape (comme Iterative Deepening).

Une même action peut être apparaître plusieurs fois.

Elle peut donc être représentée par plusieurs variables propositionnelles.

Plusieurs codages possibles

Le codage direct

La longueur d'un plan n'est pas connue initialement.

Aussi, on procède par étape (comme Iterative Deepening).

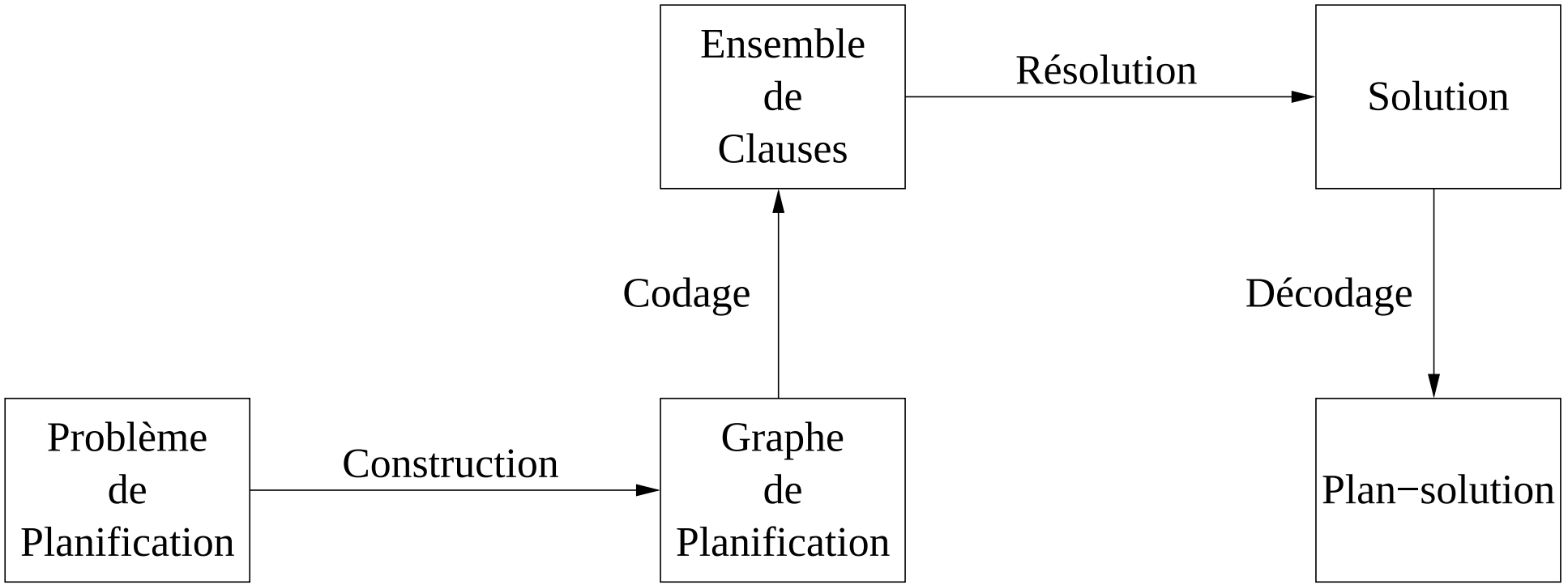
Une même action peut être apparaître plusieurs fois.

Elle peut donc être représentée par plusieurs variables propositionnelles.

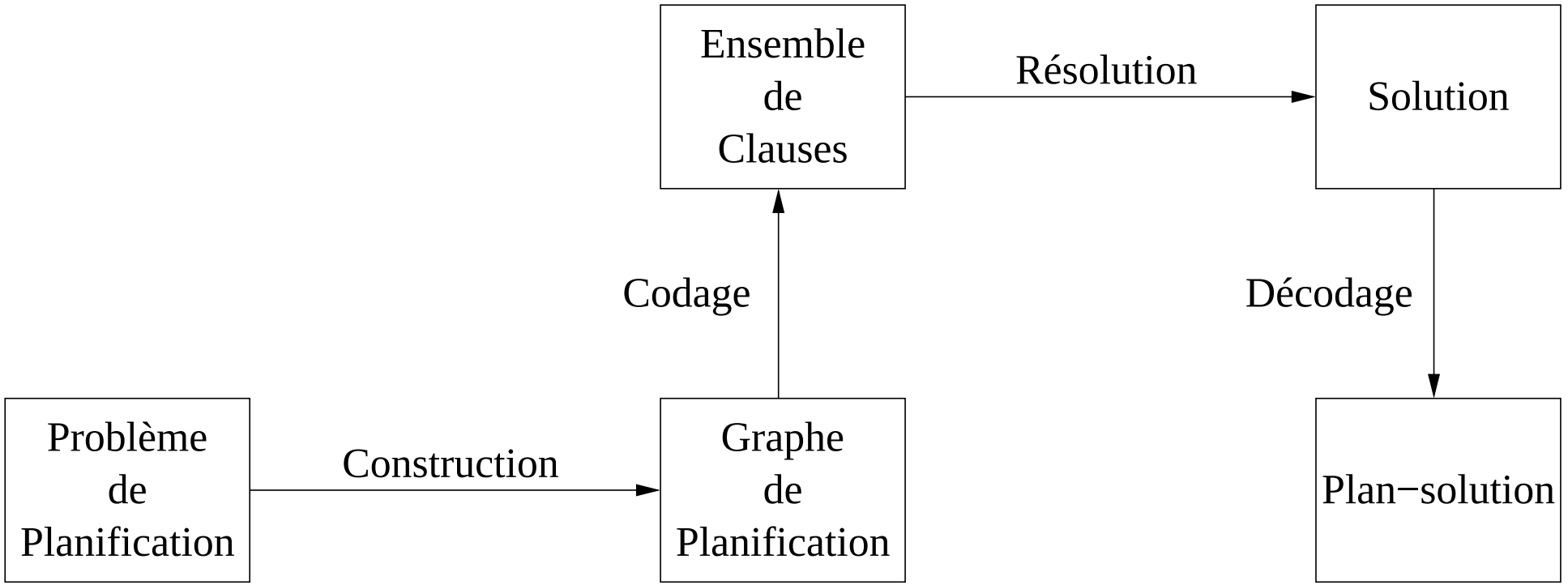
Plusieurs codages possibles

Exemple de planificateur : SATPLAN

SAT et GRAPHPLAN



SAT et GRAPHPLAN

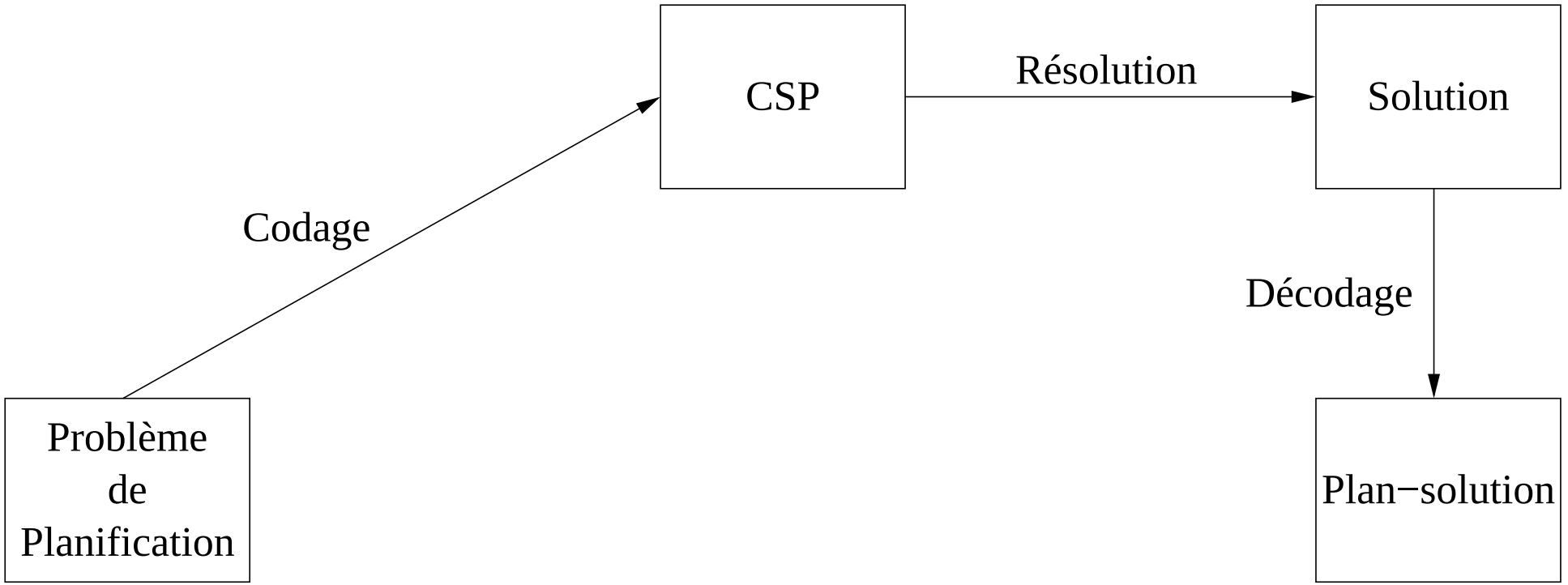


Exemple de planificateurs : BLACKBOX, CSATPLAN

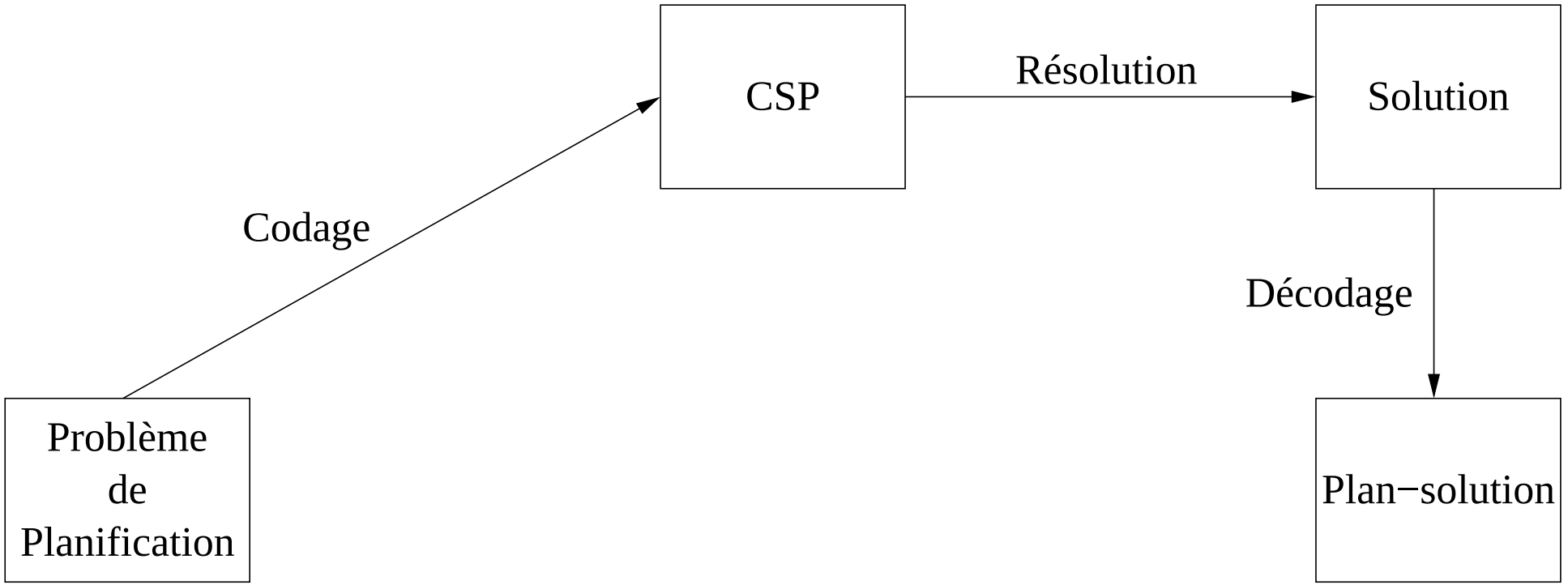
Plan

1. Introduction
2. Le cadre STRIPS
3. La planification et les graphes d'états
4. Le planificateur GRAPHPLAN
5. La planification et la logique propositionnelle
6. La planification et les CSP

Codage direct sous forme CSP

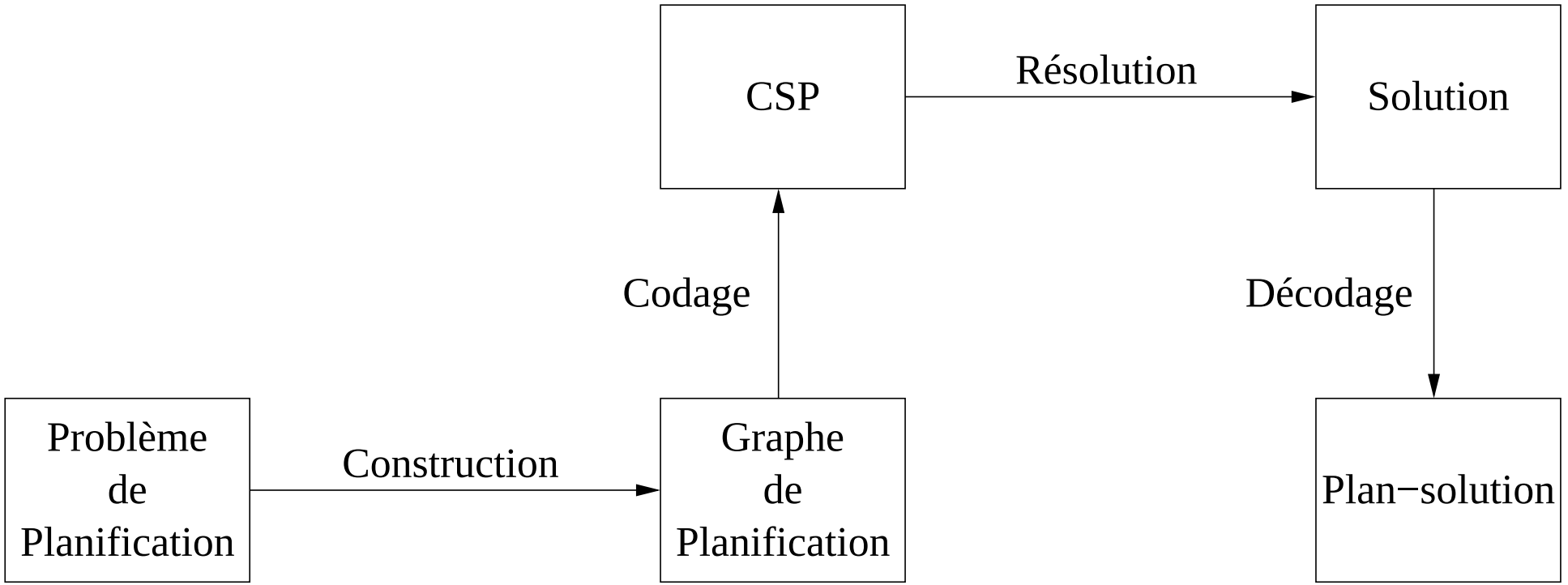


Codage direct sous forme CSP

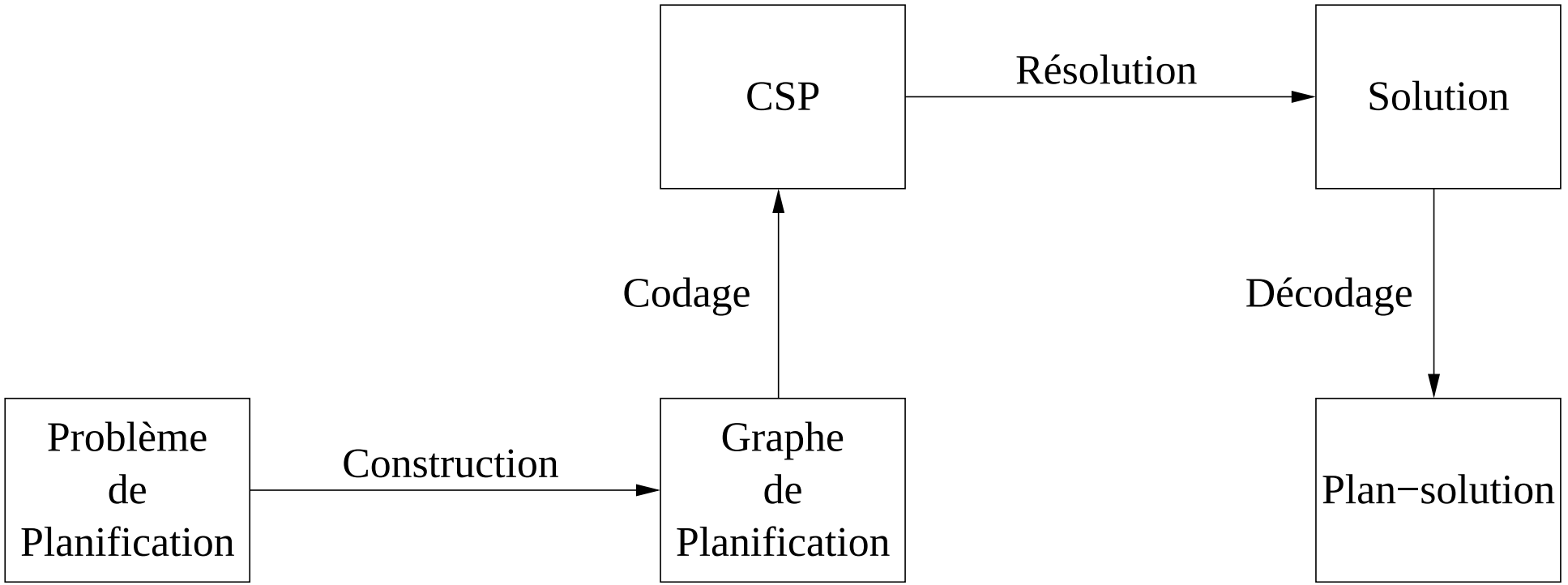


Exemple de planificateur : CPLAN

CSP et GRAPHPLAN



CSP et GRAPHPLAN



Exemple de planificateur : GP-CSP