Problème du Planning

- Trouver une séquence d'actions menant à un but prédéfini
- Le problème est souvent exprimé comme une description d'un *état initial*, un ensemble d'*actions* et un *but*.
- Lorsque le système est dans un certain état, chaque action a pour effet de faire changer le système d'état, ou de rester dans le même état.
- En vérification, ce problème est plus généralement appelé *problème d'atteignabilité*.
- Après avoir vu la définition, nous allons voir comment tenter de le résoudre avec des solveurs SAT.
- Le succès de l'application des solveurs SAT à ce problème est à l'origine de l'utilisation des solveurs SAT en vérification.

Problème du Planning

- Si le système de départ est donné comme un graphe dirigé, avec un sommet initial et un sommet final, il suffit de tester qu'il existe un chemin de l'état initial à l'état but, ce qui peut être fait en temps linéaire dans la taille du graphe (nombre de sommets + nombre d'arêtes).
- Le problème du planning devient intéressant lorsque l'espace d'états du système est très grand, et n'est pas représenté explicitement en mémoire, comme par exemple toutes les configurations possibles d'un programme (valeurs des variables).
- En particulier, on s'intéresse à des systèmes dans les états sont des interprétations d'un ensemble de propositions.

Problème du Planning : Exemple

Chou-Chèvre-Loup

Un capitaine doit faire traverser une rivière à un chou, une chèvre et un loup, d'une berge A à une berge B.

- initialement, ils sont tous du même côté.
- lorsque le capitaine ne les surveille pas (i.e. n'est pas sur la même berge qu'eux), la chèvre mange le chou, et le loup la chèvre.
- le capitaine ne peut transporter qu'une seule personne à la fois (chou, chèvre ou loup).

Comment fait-il ?

Problème du Planning – États du système

- On se donne un ensemble fini de propositions $P = \{p_1, \ldots, p_n\}$.
- Chaque état du système est une interprétation $s : P \to \{0, 1\}$.
- Il existe donc 2^n états qu'on ne peut donc pas représenté explicitement en mémoire.
- Par exemple, les états d'un système peuvent représenter les valeurs des variables d'un programme (chaque variable entière de taille p au plus peut être représentée par $log_2(p)$ propositions dont les valeurs correspondent aux bits de la décomposition binaire).

Problème du Planning – États du système

- On se donne un ensemble fini de propositions $P = \{p_1, \ldots, p_n\}$.
- Chaque état du système est une interprétation $s : P \to \{0, 1\}$.
- Il existe donc 2^n états qu'on ne peut donc pas représenté explicitement en mémoire.
- Par exemple, les états d'un système peuvent représenter les valeurs des variables d'un programme (chaque variable entière de taille p au plus peut être représentée par $log_2(p)$ propositions dont les valeurs correspondent aux bits de la décomposition binaire).
- Pour notre exemple, les propositions sont $P = \{chou, chevre, loup, capitaine\}$ qui sont mises à vrai ou faux selon que le chou, la chèvre, le loup et le capitaine se trouve sur la berge A ou B (faux = berge A et vrai = berge B). Par exemple, $s(chou) = 0$, $s(chevre = 1)$, $s(loup) = 1$, $s(capitaine) = 0$ veut dire que le chou et le capitaine sont sur la berge A alors que la chèvre et le loup sont sur la berge B.

Problème du Planning – État initial

- L'état initial s_0 est une interprétation $s_0 : P \to \{0, 1\}$.
- Par exemple, les valeurs initiales des variables d'un programme.

Problème du Planning – État initial

- L'état initial s_0 est une interprétation $s_0 : P \to \{0, 1\}$.
- Par exemple, les valeurs initiales des variables d'un programme.
- Dans notre exemple, $s_0(p) = 0$ pour tout $p \in \{chou, chevre, loup, capitaine\}$ (tout le monde est sur la berge A au départ).

Problème du Planning – But

- Le but ϕ_{but} est une formule de logique propositionnelle.
- On cherche à atteindre un état s tel que $s \models \phi_{but}$.
- Dans notre exemple, $\phi_{but} = chou \wedge chevre \wedge loup$ (le chou, le loup et la chèvre sont sur la berge B).

Problème du Planning – Actions et Transitions

- On se donne un ensemble fini d'*actions* $A = \{a_1, \ldots, a_p\}$
- Chaque action a pour effet de faire changer l'état du système, mais ne peut pas être appliquée à partir de n'importe quel état.
- A chaque action a, on associe une *précondition* pre_a qui est une formule de logique propositionnelle. L'action a est applicable dans un état s si $s \models pre_a$.

Problème du Planning – Actions et Transitions

- On se donne un ensemble fini d'*actions* $A = \{a_1, \ldots, a_p\}$
- Chaque action a pour effet de faire changer l'état du système, mais ne peut pas être appliquée à partir de n'importe quel état.
- A chaque action a, on associe une *précondition* pre_a qui est une formule de logique propositionnelle. L'action a est applicable dans un état s si $s \models pre_a$.
- A chaque action a, on associe un ensemble eff_a d'*effets*, qui sont des formules propositionnelles de la forme $f \rightarrow d$ où f est une formule quelconque et d est une **conjonction** de littéraux (sans littéraux complémentaires).

Problème du Planning – Actions et Transitions

- On se donne un ensemble fini d'*actions* $A = \{a_1, \ldots, a_p\}$
- Chaque action a pour effet de faire changer l'état du système, mais ne peut pas être appliquée à partir de n'importe quel état.
- A chaque action a, on associe une *précondition* pre_a qui est une formule de logique propositionnelle. L'action a est applicable dans un état s si $s \models pre_a$.
- A chaque action a, on associe un ensemble eff_a d'*effets*, qui sont des formules propositionnelles de la forme $f \rightarrow d$ où f est une formule quelconque et d est une **conjonction** de littéraux (sans littéraux complémentaires).
- On dira que l'ensemble d'effets eff_a est *consistant* en un état s si il n'existe pas $f \rightarrow d$ et $f' \rightarrow d'$ dans eff_a tels que $s \models f \wedge f'$, et d et d' contiennent des littéraux complémentaires.

Problème du Planning – Actions et Transitions

- On se donne un ensemble fini d'*actions* $A = \{a_1, \ldots, a_p\}$
- Chaque action a pour effet de faire changer l'état du système, mais ne peut pas être appliquée à partir de n'importe quel état.
- A chaque action a, on associe une *précondition* pre_a qui est une formule de logique propositionnelle. L'action a est applicable dans un état s si $s \models pre_a$.
- A chaque action a, on associe un ensemble eff_a d'*effets*, qui sont des formules propositionnelles de la forme $f \rightarrow d$ où f est une formule quelconque et d est une **conjonction** de littéraux (sans littéraux complémentaires).
- On dira que l'ensemble d'effets eff_a est *consistant* en un état s si il n'existe pas $f \rightarrow d$ et $f' \rightarrow d'$ dans eff_a tels que $s \models f \wedge f'$, et d et d' contiennent des littéraux complémentaires.
- Si $s \models pre_a$ et eff_a est consistant en s, alors l'action a peut être appliquée à l'état s, et a pour effet de changer l'état du système vers un état s' défini comme suit :

Problème du Planning – Actions et Transitions

- Si $s \models pre_a$ et eff_a est consistant en s, alors l'action a peut être appliquée à l'état s, et a pour effet de changer l'état du système vers un état s' défini comme suit :

$$s' : P \rightarrow \{0,1\}$$
$$p \mapsto \begin{cases} 0 & \text{si il existe } f \rightarrow d \in eff_a \text{ tel que } s \models f \text{ et } \neg p \in d \\ 1 & \text{si il existe } f \rightarrow d \in eff_a \text{ tel que } s \models f \text{ et } p \in d \\ s(p) & \text{sinon.} \end{cases}$$

- en d'autre terme, si $s \models f$, alors toute variable de d est mise à vrai ou faux par s' selon qu'elle apparaît positivement ou négativement dans d
- les valeurs de vérités des propositions qui n'apparaissent dans aucun d restent inchangés.

Problème du Planning – Actions et Transitions (Exemple)

- Dans l'exemple, nous avons quatre actions possibles $A = \{trans\_chou, trans\_chevre, trans\_loup, trans\_vide\}$ selon que le capitaine décide de transporter le chou, la chèvre, le loup, ou rien.

Problème du Planning – Actions et Transitions (Exemple)

- Dans l'exemple, nous avons quatre actions possibles $A = \{trans\_chou, trans\_chevre, trans\_loup, trans\_vide\}$ selon que le capitaine décide de transporter le chou, la chèvre, le loup, ou rien.
- Chaque action ne peut s'appliquer dans n'importe quel état : pour transporter x, il faut que le capitaine soit sur la même berge que x

Problème du Planning – Actions et Transitions (Exemple)

- Dans l'exemple, nous avons quatre actions possibles $A = \{trans\_chou, trans\_chevre, trans\_loup, trans\_vide\}$ selon que le capitaine décide de transporter le chou, la chèvre, le loup, ou rien.
- Chaque action ne peut s'appliquer dans n'importe quel état : pour transporter x, il faut que le capitaine soit sur la même berge que x
- Il faut aussi qu'on ne soit pas dans une situation conflictuelle entre le chou et la chèvre, ou le loup et la chèvre (dans ce cas il ne doit avoir aucune action applicable)

Problème du Planning – Actions et Transitions (Exemple)

- Dans l'exemple, nous avons quatre actions possibles $A = \{trans\_chou, trans\_chevre, trans\_loup, trans\_vide\}$ selon que le capitaine décide de transporter le chou, la chèvre, le loup, ou rien.
- Chaque action ne peut s'appliquer dans n'importe quel état : pour transporter x, il faut que le capitaine soit sur la même berge que x
- Il faut aussi qu'on ne soit pas dans une situation conflictuelle entre le chou et la chèvre, ou le loup et la chèvre (dans ce cas il ne doit avoir aucune action applicable)
- Pour tout $x \in \{chou, chevre, loup\}$, on aura donc la précondition suivante pre_{trans_x} pour l'action $trans\_x$:

$$(capitaine \leftrightarrow x) \wedge \neg((chou \leftrightarrow chevre) \wedge (chou \leftrightarrow \neg capitaine)) \wedge \neg((loup \leftrightarrow chevre) \wedge (loup \leftrightarrow \neg capitaine))$$

Problème du Planning – Actions et Transitions (Exemple)

- Dans l'exemple, nous avons quatre actions possibles $A = \{trans\_chou, trans\_chevre, trans\_loup, trans\_vide\}$ selon que le capitaine décide de transporter le chou, la chèvre, le loup, ou rien.
- Chaque action ne peut s'appliquer dans n'importe quel état : pour transporter x, il faut que le capitaine soit sur la même berge que x
- Il faut aussi qu'on ne soit pas dans une situation conflictuelle entre le chou et la chèvre, ou le loup et la chèvre (dans ce cas il ne doit avoir aucune action applicable)
- Pour tout $x \in \{chou, chevre, loup\}$, on aura donc la précondition suivante pre_{trans_x} pour l'action $trans\_x$:

$$(capitaine \leftrightarrow x) \wedge \neg((chou \leftrightarrow chevre) \wedge (chou \leftrightarrow \neg capitaine)) \wedge \neg((loup \leftrightarrow chevre) \wedge (loup \leftrightarrow \neg capitaine))$$

- l'effet de transporter x est de déplacer x de l'autre côté, donc de changer sa valeur de vérité. On aura donc les deux effets suivants : $x \rightarrow \neg x$ et $\neg x \rightarrow x$, et pareil pour le capitaine : $capitaine \rightarrow \neg capitaine$ et $\neg capitaine \rightarrow capitaine$.

Problème du Planning – Actions et Transitions (Exemple)

- Pour l'action *trans_vide*, on a simplement la précondition

$$\neg((chou \leftrightarrow chevre) \wedge (chou \leftrightarrow \neg capitaine)) \\ \wedge \neg((loup \leftrightarrow chevre) \wedge (loup \leftrightarrow \neg capitaine))$$

- l'effet de ne transporter rien du tout change simplement la valeur de la proposition *capitaine*. On aura donc les effets $capitaine \rightarrow \neg capitaine$ et $\neg capitaine \rightarrow capitaine$.

Problème du Planning – Actions et Transitions (Exemple)

- Supposons qu'on soit dans l'état initial s_0 tel que

$$s(x) = 0 \text{ pour tout } x \in \{chevre, chou, capitaine, loup\}$$

- considérons l'action *trans_chou*. Est-ce qu'elle est applicable en s_0 ?

Problème du Planning – Actions et Transitions (Exemple)

- Supposons qu'on soit dans l'état initial s_0 tel que

$$s(x) = 0 \text{ pour tout } x \in \{chevre, chou, capitaine, loup\}$$

- considérons l'action *trans_chou*. Est-ce qu'elle est applicable en s_0 ?
- Oui, elle mène à l'état s' tel que $s'(x) = 0$ pour $x \in \{chevre, loup\}$ et $s'(y) = 1$ pour $y \in \{capitaine, chou\}$.

Problème du Planning – Actions et Transitions (Exemple)

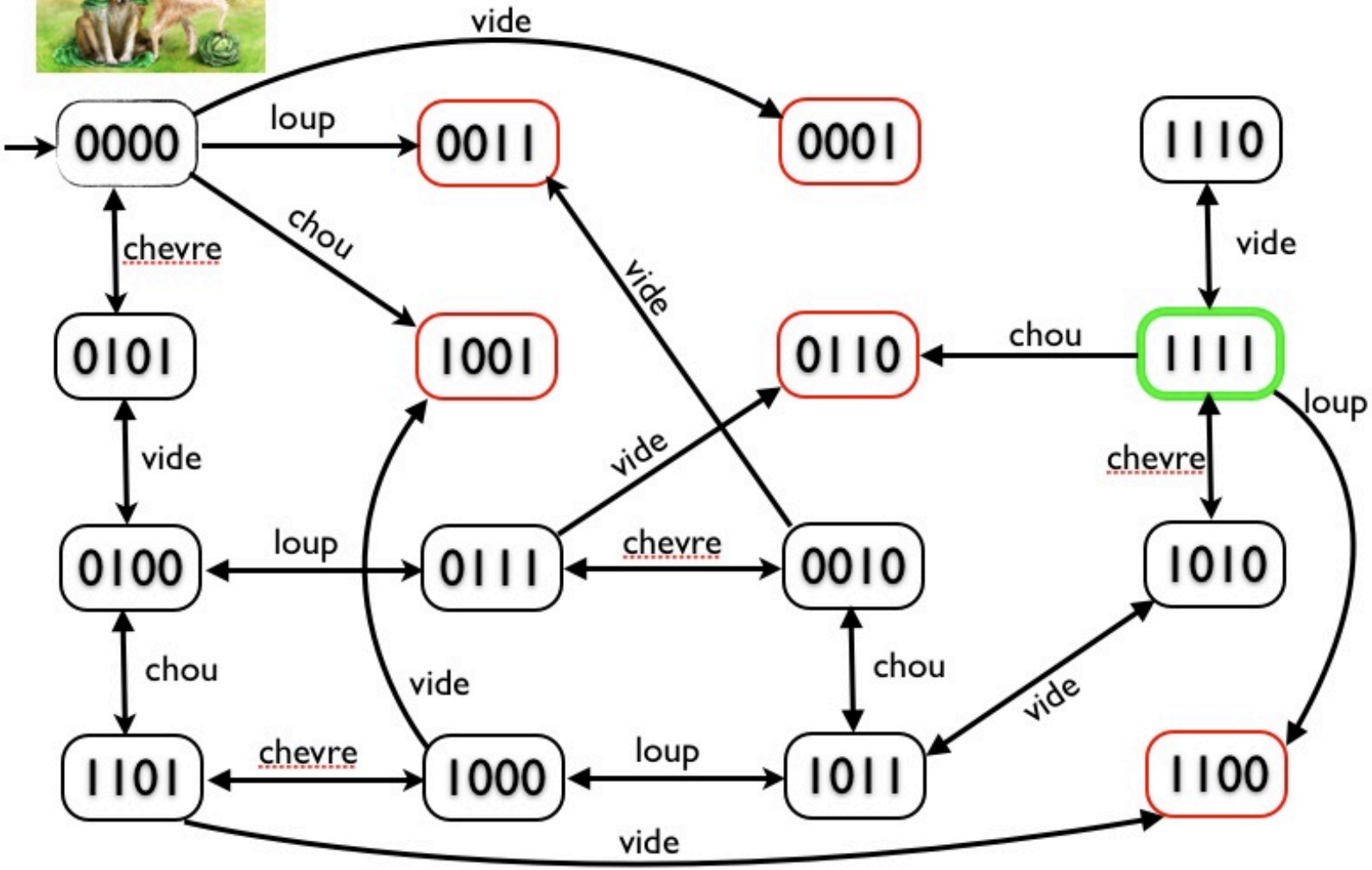
- Supposons qu'on soit dans l'état initial s_0 tel que

$$s(x) = 0 \text{ pour tout } x \in \{chevre, chou, capitaine, loup\}$$

- considérons l'action *trans_chou*. Est-ce qu'elle est applicable en s_0 ?
- Oui, elle mène à l'état s' tel que $s'(x) = 0$ pour $x \in \{chevre, loup\}$ et $s'(y) = 1$ pour $y \in \{capitaine, chou\}$.
- A partir de s', aucune action n'est applicable.

Systèmes de transition explicite

chou / chevre / loup / capitaine



Problème du Planning – Séquences d'exécution

- Etant donné un état s et une action a applicable en s, l'état successeur atteint est noté $succ_a(s)$.
- Etant donné une séquence d'action $a_1, \ldots, a_k$ successivement applicables à partir de s, on note $reach_{a_1,\ldots,a_k}(s)$ l'état atteint après avoir appliqué les k actions, i.e. :

$$reach_{a_1,\ldots,a_k}(s) \;=\; succ_{a_k}(succ_{a_{k-1}}(\ldots succ_{a_1}(s)\ldots))$$

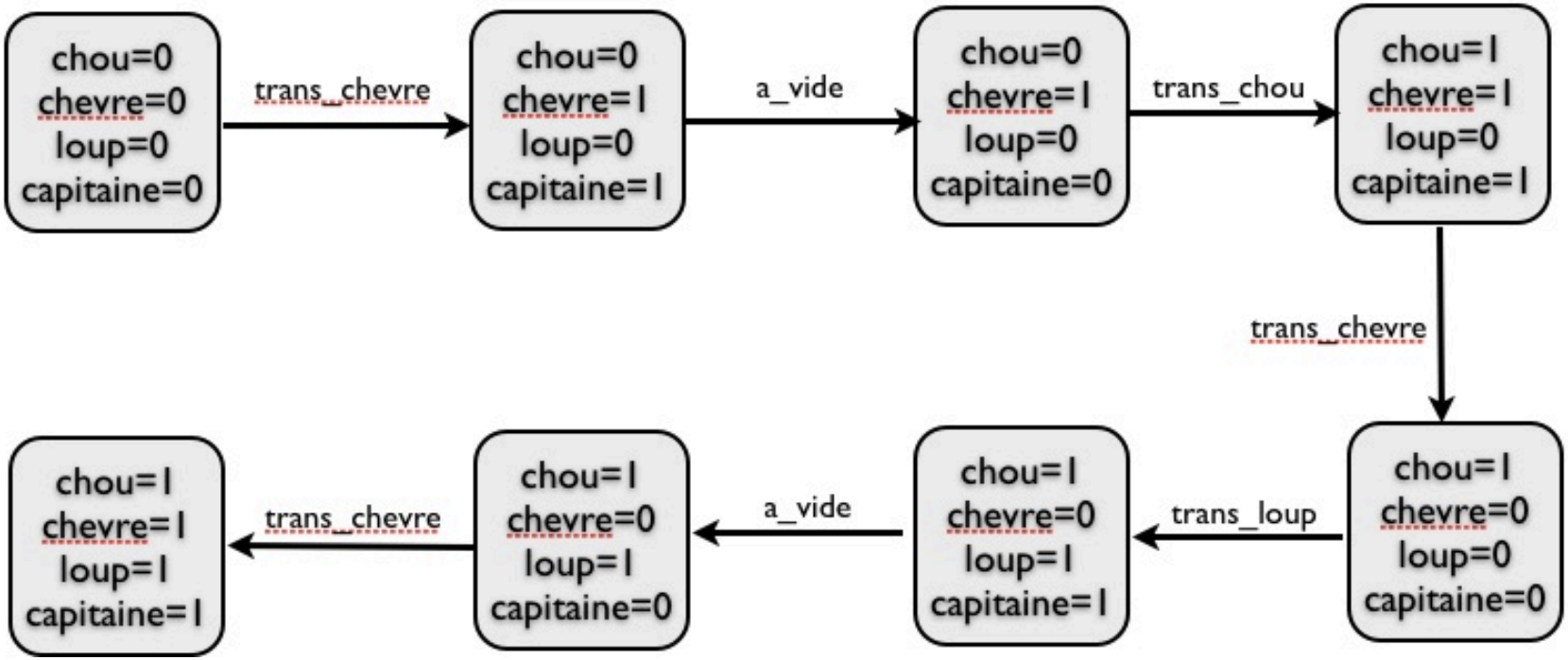
Définition (Problème de Planning)

Etant donnée une instance $(P, A, s_0, (pre_a, eff_a)_{a\in A}, \phi_{but})$ du problème de planning, l'objectif est de décider s'il existe une séquence d'actions applicables $a_1, \ldots, a_k$ à partir de l'état initial s_0 telle que $reach_{a_1,\ldots,a_k}(s_0) \models \phi_{but}$. Dans ce cas, $a_1, \ldots, a_k$ est appelé *plan*.

Autrement dit, on veut pouvoir appliquer une séquence d'actions à partir de l'état initial, qui mène dans un état qui satisfait le but ϕ_{but}.

Exemple de Plan et Séquence d'Exécution Associée

La suite d'action *trans_chevre*, *trans_vide*, *trans_chou*, *trans_chevre*, *trans_loup*, *trans_vide*, *trans_chevre* est un plan.



Modélisation du planning comme problème SAT

- On va s'intéresser à chercher des plans du taille fixée $k \in \mathbb{N}$.
- On va construire une formule ϕ_k telle que ϕ_k est satisfaisable ssi il existe un plan de longueur k.
- On appelera le SAT solveur pour des valeurs croissantes de k jusqu'à trouver un plan de taille k ou abandonner.

Modélisation du planning comme problème SAT

- On va s'intéresser à chercher des plans du taille fixée $k \in \mathbb{N}$.
- On va construire une formule ϕ_k telle que ϕ_k est satisfaisable ssi il existe un plan de longueur k.
- On appelera le SAT solveur pour des valeurs croissantes de k jusqu'à trouver un plan de taille k ou abandonner.
- L'idée principale est de dupliquer les propositions P pour tout instant $t \in \{0, \ldots, k\}$ et de coder les transitions comme des formules propositionnelles.
- Pour tout $t \in \{0, \ldots, k\}$, on note P^t l'ensemble des propositions de P indexées par t, i.e.

$$P^t = \{p^t \mid p \in P\}$$

- chaque proposition p^t représente la valeur de la proposition p à l'étape t.

Codage de l'état initial

L'état initial $s_o : P \to \{0, 1\}$ est représenté par la formule

$$\phi_{s_0} = \bigwedge_{p \in P \text{ telle que } s_0(p)=1} p^0 \ \wedge \bigwedge_{p \in P \text{ telle que } s_0(p)=0} \neg p^0$$

Codage de l'état initial

L'état initial $s_o : P \to \{0, 1\}$ est représenté par la formule

$$\phi_{s_0} = \bigwedge_{p \in P \text{ telle que } s_0(p)=1} p^0 \ \wedge \bigwedge_{p \in P \text{ telle que } s_0(p)=0} \neg p^0$$

Par exemple, pour chou-chèvre-loup, on aurait

$$\neg chou^0 \wedge \neg chevre^0 \wedge \neg loup^0 \wedge \neg capitaine^0$$

Codage du but

Rappel : le but est donné par une formule propositionnelle ϕ_{but} construite sur l'ensemble de propositions P.

- le but doit être atteint à l'étape k
- il suffit donc de remplacer toute variable p de ϕ_{but} par p^k
- on note ϕ_{but}^k la formule obtenue

Codage du but

Rappel : le but est donné par une formule propositionnelle ϕ_{but} construite sur l'ensemble de propositions P.

- le but doit être atteint à l'étape k
- il suffit donc de remplacer toute variable p de ϕ_{but} par p^k
- on note ϕ^k_{but} la formule obtenue

Dans notre exemple, on obtient :

$$\phi^k_{but} = chou^k \wedge loup^k \wedge chevre^k$$

Codage des transitions

On va coder une transition de l'étape i à l'étape $i+1$. Il faudra donc exprimer par une formule la valeur des propositions de P^{i+1} en fonction de la valeur des propositions de P^i et des actions choisies.

- choisissons d'abord une action $a \in A$. Sa précondition est donnée par la formule pre_a et ses effets sont donnés par un ensemble eff_a de formules de la forme $f \rightarrow d$ où d est une conjonction de littéraux

Codage des transitions

On va coder une transition de l'étape i à l'étape $i+1$. Il faudra donc exprimer par une formule la valeur des propositions de P^{i+1} en fonction de la valeur des propositions de P^i et des actions choisies.

- choisissons d'abord une action $a \in A$. Sa précondition est donnée par la formule pre_a et ses effets sont donnés par un ensemble eff_a de formules de la forme $f \rightarrow d$ où d est une conjonction de littéraux
- pour pouvoir appliquer a à l'étape i, il faut déjà que l'interprétation courante des variables de P^i satisfasse la formule précondition pre_a, on aura donc la contrainte notée pre_a^i, obtenue en remplaçant toute proposition p de la formule pre_a par p^i

Codage des transitions

- ensuite, il faut encoder les effets. Pour tout littéral l (donc de la forme p ou $\neg p$), on note $cond_a(l)$ l'ensemble des formules f tel que l apparaît dans d pour un certain effet $(f \rightarrow d) \in eff_a$.
- autrement dit, $cond_a(l)$ est l'ensemble des conditions f telle que si l'une d'entre elle est satisfaite, alors l'action a aura pour effet l
- Par exemple, $cond_{trans\_vide}(\neg capitaine) = \{capitaine\}$, $cond_{trans\_chou}(chevre) = \varnothing$ et $cond_{trans\_chou}(\neg chou) = chou$.

Codage des transitions

- ensuite, il faut encoder les effets. Pour tout littéral l (donc de la forme p ou $\neg p$), on note $cond_a(l)$ l'ensemble des formules f tel que l apparaît dans d pour un certain effet $(f \rightarrow d) \in eff_a$.
- autrement dit, $cond_a(l)$ est l'ensemble des conditions f telle que si l'une d'entre elle est satisfaite, alors l'action a aura pour effet l
- Par exemple, $cond_{trans\_vide}(\neg capitaine) = \{capitaine\}$, $cond_{trans\_chou}(chevre) = \varnothing$ et $cond_{trans\_chou}(\neg chou) = chou$.
- Pour l'action a, on obtient alors la formule $\phi_a^{i,i+1}$ suivante :

$$\bigwedge_{p \in P} p^{i+1} \leftrightarrow ((\bigvee_{f \in cond_a(p)} f^i) \vee (p^i \wedge \bigwedge_{f \in cond_a(\neg p)} \neg f^i))$$

- Autrement dit, pour toute proposition $p \in P$, sa valeur est vraie l'étape $i+1$ ssi il existe un effet $(f \rightarrow d) \in eff_a$ tel que p apparaît dans d et f est satisfait à l'étape i, ou p était vraie à l'étape i et il n'existe pas d'effet $(f \rightarrow d)$ tel que $\neg p$ est dans d et f^i est satisfait.

Codage des transitions

- ensuite, il faut encoder les effets. Pour tout littéral l (donc de la forme p ou $\neg p$), on note $cond_a(l)$ l'ensemble des formules f tel que l apparaît dans d pour un certain effet $(f \rightarrow d) \in eff_a$.
- autrement dit, $cond_a(l)$ est l'ensemble des conditions f telle que si l'une d'entre elle est satisfaite, alors l'action a aura pour effet l
- Par exemple, $cond_{trans\_vide}(\neg capitaine) = \{capitaine\}$, $cond_{trans\_chou}(chevre) = \varnothing$ et $cond_{trans\_chou}(\neg chou) = chou$.
- Pour l'action a, on obtient alors la formule $\phi_a^{i,i+1}$ suivante :

$$\bigwedge_{p \in P} p^{i+1} \leftrightarrow ((\bigvee_{f \in cond_a(p)} f^i) \vee (p^i \wedge \bigwedge_{f \in cond_a(\neg p)} \neg f^i))$$

- Autrement dit, pour toute proposition $p \in P$, sa valeur est vraie l'étape $i+1$ ssi il existe un effet $(f \rightarrow d) \in eff_a$ tel que p apparaît dans d et f est satisfait à l'étape i, ou p était vraie à l'étape i et il n'existe pas d'effet $(f \rightarrow d)$ tel que $\neg p$ est dans d et f^i est satisfait.
- Enfin, la transition de l'étape i à l'étape $i+1$ est possible s'il existe une action applicable, le codage final est donc donné par la formule

Codage des transitions

$$\phi^{i,i+1} = \bigvee_{a \in A} pre_a^i \wedge \phi_a^{i,i+1}$$

Pour tout $s : P \to \{0,1\}$, on note $s^i : P^i \to \{0,1\}$ l'interprétation définie par $s^i(p^i) = s(p)$ pour tout $p \in P$.
Etant données deux interprétation $s^i : P^i \to \{0,1\}$ et $r^{i+1} : P^{i+1} \to \{0,1\}$, on note $s^i \cup r^{i+1}$ l'interprétation de $P^i \cup P^{i+1}$ vers $\{0,1\}$ qui est identique à s^i pour les propositions de P^i et identique à r^{i+1} pour les propositions de P^{i+1}.

Lemme

Soit s, r deux interprétations et a une action.

1. $s^i \cup r^{i+1} \models pre_a^i \wedge \phi_a^{i,i+1}$ *si l'action a est applicable en s et $succ_a(s) = r$.*
2. $s^i \cup r^{i+1} \models \phi^{i,i+1}$ *si il existe une action a applicable en s telle que $succ_a(s) = r$.*

Exemple

Pour le problème chou-chèvre-loup, on obtient la formule suivante pour $\phi^{i,i+1}_{trans\_chou}$

$$
\begin{aligned}
& (capitaine^i \leftrightarrow chou^i) \\
\wedge\ & \neg((chou^i \leftrightarrow chevre^i) \wedge (chou^i \leftrightarrow \neg capitaine^i)) \\
\wedge\ & \neg((loup^i \leftrightarrow chevre^i) \wedge (loup^i \leftrightarrow \neg capitaine^i)) \\
\wedge\ & capitaine^{i+1} \leftrightarrow (\neg capitaine^i \vee (capitaine^i \wedge \neg capitaine^i) \\
\wedge\ & chou^{i+1} \leftrightarrow (\neg chou^i \vee (chou^i \wedge \neg chou^i) \\
\wedge\ & loup^{i+1} \leftrightarrow loup^i \\
\wedge\ & chevre^{i+1} \leftrightarrow chevre^i
\end{aligned}
$$

qui se simplifie en :

$$
\begin{aligned}
& (capitaine^i \leftrightarrow chou^i) \\
\wedge\ & \neg((chou^i \leftrightarrow chevre^i) \wedge (chou^i \leftrightarrow \neg capitaine^i)) \\
\wedge\ & \neg((loup^i \leftrightarrow chevre^i) \wedge (loup^i \leftrightarrow \neg capitaine^i)) \\
\wedge\ & capitaine^{i+1} \leftrightarrow \neg capitaine^i \\
\wedge\ & chou^{i+1} \leftrightarrow \neg chou^i \\
\wedge\ & loup^{i+1} \leftrightarrow loup^i \\
\wedge\ & chevre^{i+1} \leftrightarrow chevre^i
\end{aligned}
$$

Codage Final

Rappel : on veut construire une formule ϕ_k telle que ϕ_k est satisfaisable ssi il existe un plan de taille k.
On prend alors la formule :

$$\phi_k = \phi_{s_0} \wedge \phi^{0,1} \wedge \phi^{1,2} \wedge \cdots \wedge \phi^{k-1,k} \wedge \phi^k_{but}$$

Théorème

ϕ_k est satisfaisable ssi il existe un plan de taille k.

Attention, pour utiliser un solveur SAT, il faut encore mettre ϕ_k en FNC !

Exercice

Modifier le codage du problème de planning en problème SAT pour qu'il satisfasse ϕ_k est satisfaisable ssi il existe un plan de taille k **au plus**.

Exercice

Enigme

Quatre indiens sont poursuivis par quatre cowboys. Il doivent franchir un pont pour s'en sortir, mais il fait nuit et on besoin d'une torche pour s'éclairer. Il ne possède qu'une seule torche et ne peuvent pas passer à plus de deux sur le pont. Selon leurs capacités physiques, ils mettent respectivement $1, 2, 5$ et 10 minutes pour traverser le pont. Peuvent-ils tous traverser en 17 minutes ?

- Modéliser ce problème comme un problème de planning.
- Modifier le but précédent pour répondre à la question : peuvent-ils tous traverser en **au plus** 17 minutes ?