Programmation de STRIPS en PROLOG

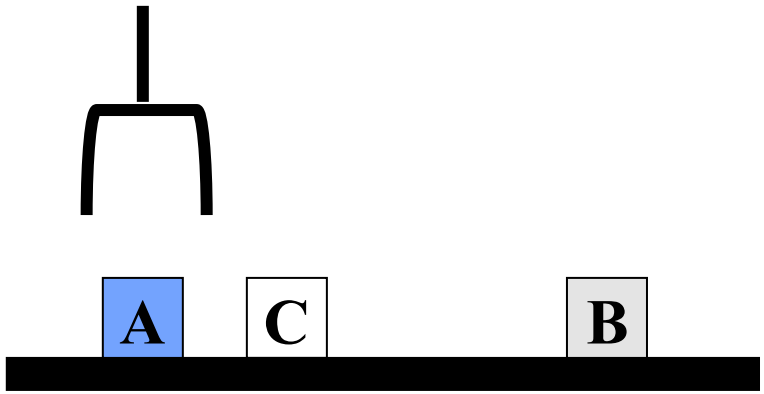
```
test(Plan):- etat_initial(I),
             etat_final(F),
             resoudre(I, [F], Plan).


resoudre(_, [], []).
resoudre(Etat, [L|Buts], Plan) :- is_list(L),
  subset(L, Etat), !, resoudre(Etat,Buts,Plan).
resoudre(Etat, [L|Buts], Plan) :- is_list(L), !,
  subtract(L, Etat, LL), append(LL, [L|Buts], NButs),
  resoudre(Etat, NButs, Plan).
resoudre(Etat, [operateur(Op)|Buts], [Op|Plan]) :- !,
  appliquer_operateur(Etat, Op, NEtat),
  resoudre(NEtat, Buts, Plan).
resoudre(Etat, [But|Buts], Plan) :- member(But, Etat), !,
  resoudre(Etat, Buts, Plan).
resoudre(Etat, [But|Buts], Plan) :-
  trouver_operateur(Etat, But, Op, P),
  intersection(P, Buts, []),
  operateur(Op, Preconds, _, _),
  resoudre(Etat, [Preconds,operateur(Op),But|Buts], Plan).
```

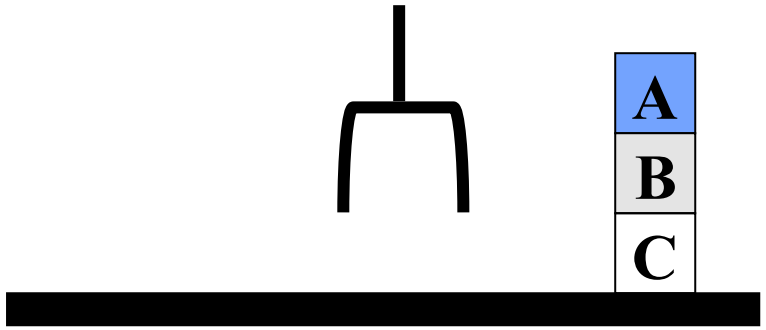
Programmation STRIPS - suite

```
trouver_operateur(Etat, But, Op, []) :-
  clause(operateur(Op, Preconds, _, AListe),Q),
  member(But, AListe), soustraction(Preconds, Etat,[]),
  call(Q), term_variables(Op,[]).
trouver_operateur(Etat, But, Op, [T|S]) :-
  clause(operateur(Op, Preconds, _, AListe),Q),
  member(But, AListe),
  soustraction(Preconds, Etat,[T|S]), call(Q),
  term_variables(Op,[]).


appliquer_operateur(Etat, Op, NEtat) :-
  operateur(Op, _, Dliste, Aliste),
  subtract(Etat, Dliste, DEtat),
  union(DEtat, Aliste, NEtat).


soustraction([], _, []).
soustraction([X|L], D, LL) :- member(X, D),
  soustraction(L, D, LL).
soustraction([X|L], D, [X|LL]) :- soustraction(L, D,
  LL).
```

Monde des blocs - opérateurs

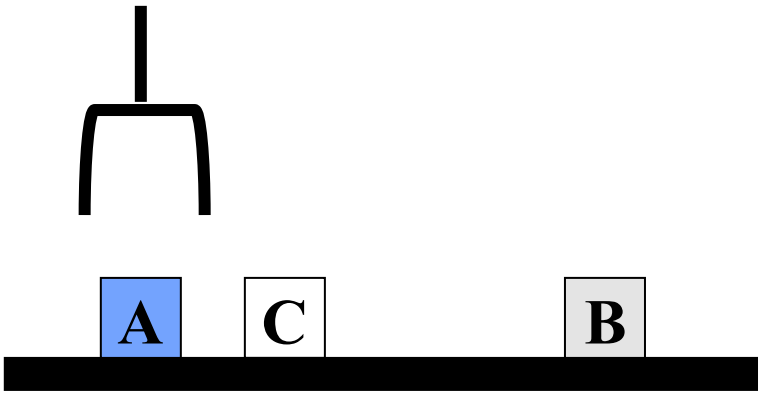
```
operateur(deposer(B),
  [poignet_prise(B)], [poignet_prise(B)],
  [poignet_vide, sur_table(B),libre(B)]).
operateur(depiler(B, C),
  [poignet_vide, bloc(B), libre(B), sur(B, C)],
  [poignet_vide, libre(B), sur(B, C)],
  [poignet_prise(B), libre(C)]):- \+ B == C.
operateur(prendre(B),
  [poignet_vide, bloc(B), libre(B), sur_table(B)],
  [poignet_vide, libre(B), sur_table(B)],
  [poignet_prise(B)]).
operateur(empiler(B,C),
  [poignet_prise(B), bloc(B), bloc(C), libre(C)],
  [poignet_prise(B), libre(C)],
  [poignet_vide, sur(B, C),libre(B)]):- \+ B == C.
```

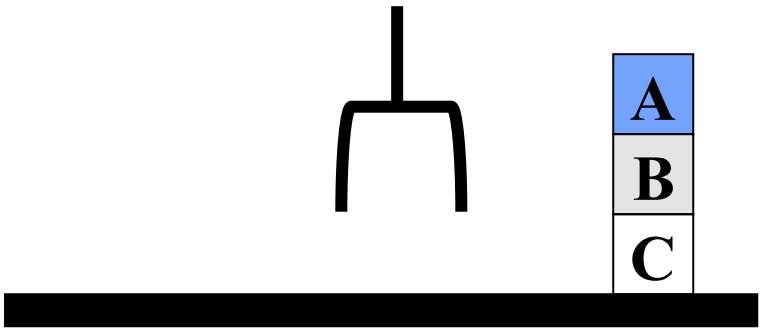




Programmation CLIPS - lancement

```
etat_initial([sur_table(a), sur_table(b),
  sur_table(c), libre(a), libre(b),
  libre(c), poignet_vide, bloc(a),
  bloc(b), bloc(c)]).

etat_final([sur_table(a), sur(b, a),
  sur(c,b), libre(c), poignet_vide]).
```

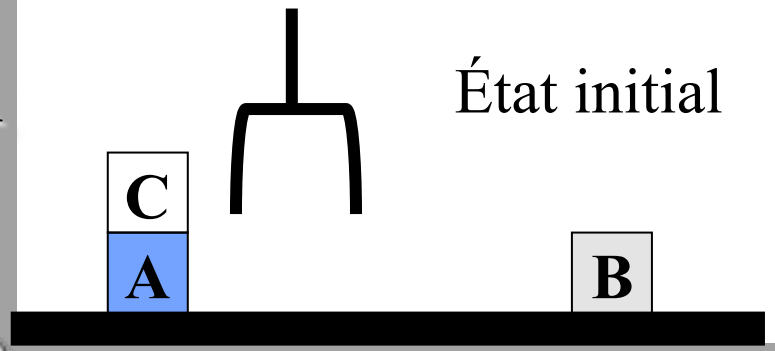




Anomalie de Sussman

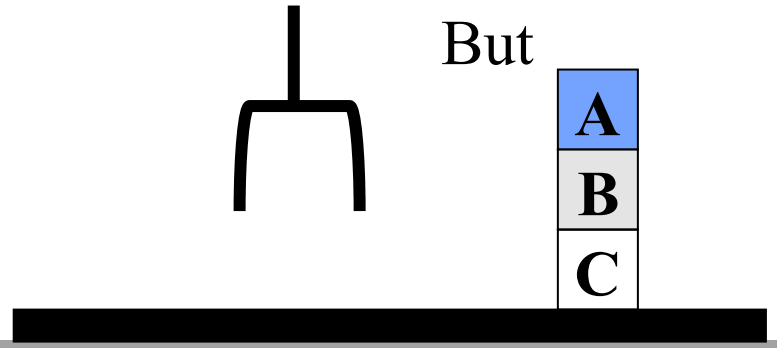
```
Achieve on(a,b) via stack(a,b) with preconds: [holding(a),clear(b)]
|Achieve holding(a) via pickup(a) with preconds: [ontable(a),clear(a),handempty]
||Achieve clear(a) via unstack(_1584,a) with preconds:
[on(_1584,a),clear(_1584),handempty]
||Applying unstack(c,a)
||Achieve handempty via putdown(_2691) with preconds: [holding(_2691)]
||Applying putdown(c)
|Applying pickup(a)
Applying stack(a,b)
Achieve on(b,c) via stack(b,c) with preconds: [holding(b),clear(c)]
|Achieve holding(b) via pickup(b) with preconds: [ontable(b),clear(b),handempty]
||Achieve clear(b) via unstack(_5625,b) with preconds:
[on(_5625,b),clear(_5625),handempty]
||Applying unstack(a,b)
||Achieve handempty via putdown(_6648) with preconds: [holding(_6648)]
||Applying putdown(a)
|Applying pickup(b)
Applying stack(b,c)
Achieve on(a,b) via stack(a,b) with preconds: [holding(a),clear(b)]
|Achieve holding(a) via pickup(a) with preconds: [ontable(a),clear(a),handempty]
|Applying pickup(a)
Applying stack(a,b)
```

From
[clear(b),clear(c),ontable(a),ontable(b),on(c,a),handempty]
To [on(a,b),on(b,c),ontable(c)]
Do:
- unstack(c,a)
- putdown(c)
- pickup(a)
- stack(a,b)
- unstack(a,b)
- putdown(a)
- pickup(b)
- stack(b,c)
- pickup(a)
- stack(a,b)



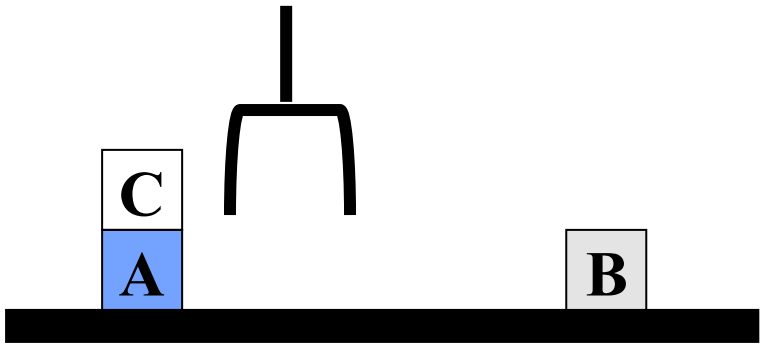
État initial

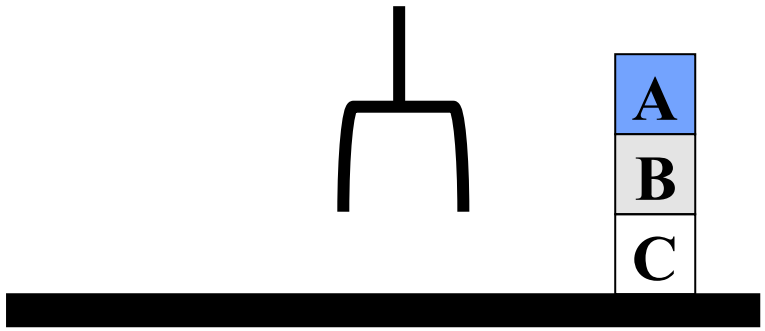
But



Anomalie de Sussman - lancement

```
etat_initial([sur_table(a), sur_table(b),
  sur(c, a), libre(b), libre(c),
  poignet_vide, bloc(a), bloc(b),
  bloc(c)]).

etat_final([sur_table(a), sur(b, a),
  sur(c,b), libre(c), poignet_vide]).
```





Un problème que STRIPS ne peut résoudre

"*Échanger le contenu de deux registres mémoires r1 et r2 dans les contenus initiaux sont respectivement a et b. On suppose un troisième registre r3 est disponible.*"

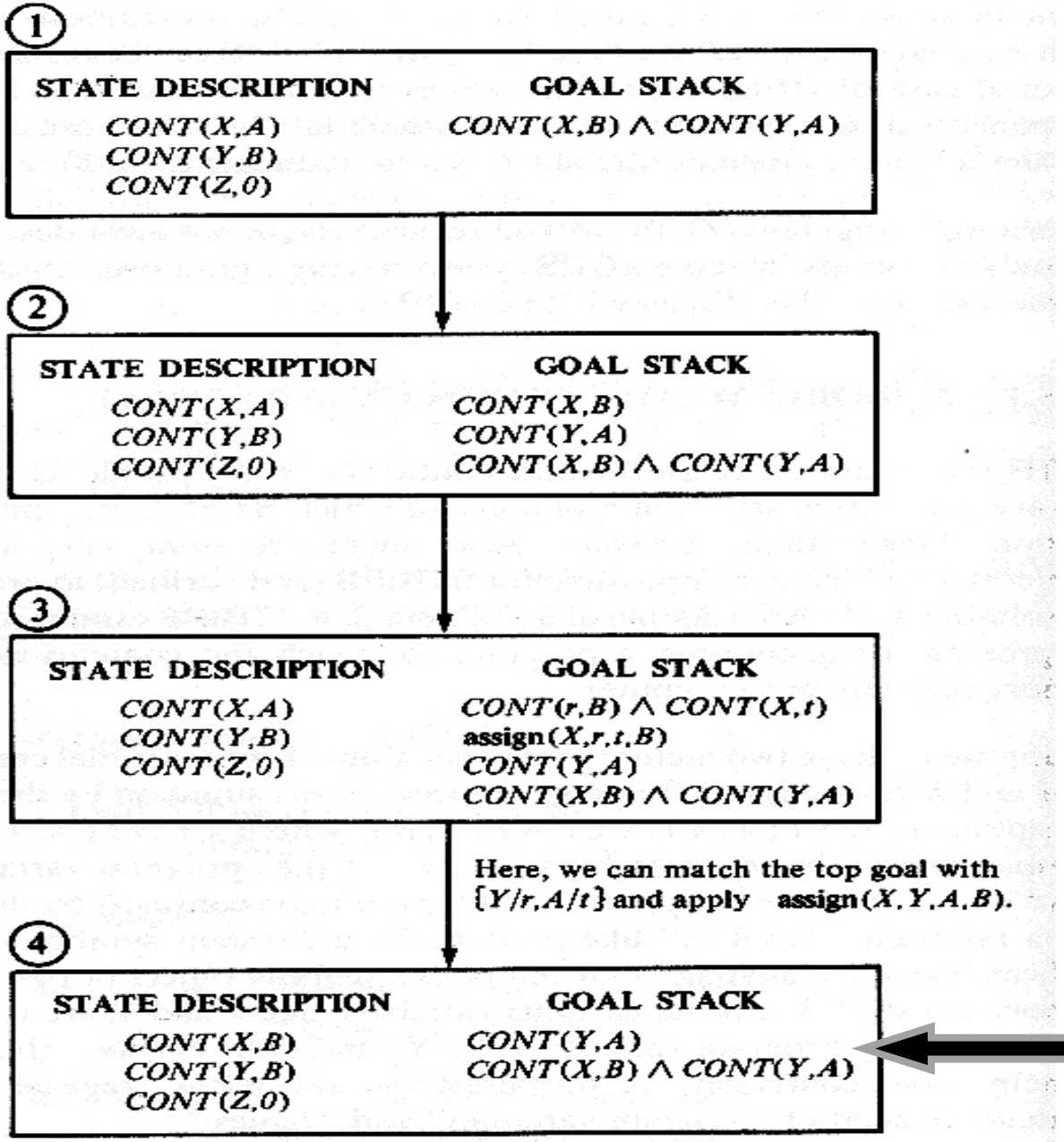
État initial: *contents(r1,a), contents(r2,b), contents(r3,0)*

But: *contents(r1,b), contents(r2,a)*

Opérateurs:

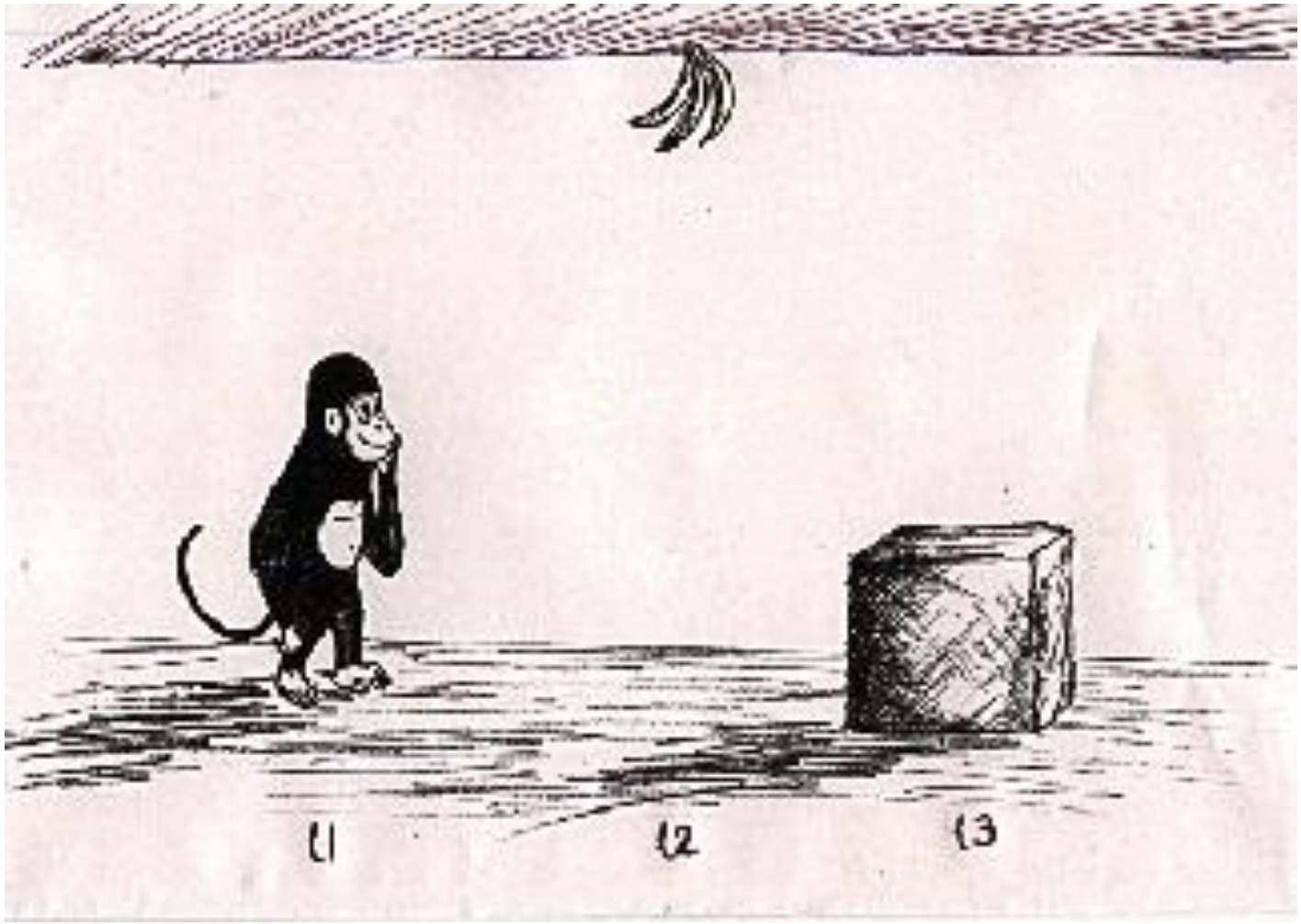
assign(X,A,Y,B)

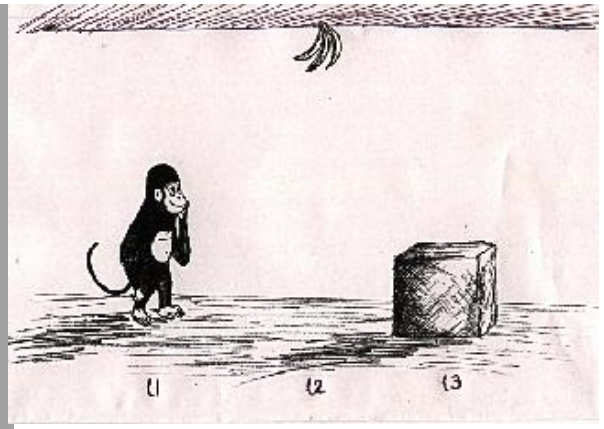
P: contents(X,A), contents(Y,B)
D: contents(Y,B)
A: contents(Y,A)


1
STATE DESCRIPTION
CONT(X,A)
CONT(Y,B)
CONT(Z,0)
GOAL STACK
CONT(X,B) ∧ CONT(Y,A)
2
STATE DESCRIPTION
CONT(X,A)
CONT(Y,B)
CONT(Z,0)
GOAL STACK
CONT(X,B)
CONT(Y,A)
CONT(X,B) ∧ CONT(Y,A)
3
STATE DESCRIPTION
CONT(X,A)
CONT(Y,B)
CONT(Z,0)
GOAL STACK
CONT(r,B) ∧ CONT(X,t)
assign(X,r,t,B)
CONT(Y,A)
CONT(X,B) ∧ CONT(Y,A)
Here, we can match the top goal with {Y/r,A/t} and apply assign(X,Y,A,B).
4
STATE DESCRIPTION
CONT(X,B)
CONT(Y,B)
CONT(Z,0)
GOAL STACK
CONT(Y,A)
CONT(X,B) ∧ CONT(Y,A)

Echec!

« Le singe et les bananes »





« Le singe et les bananes » états et opérateurs

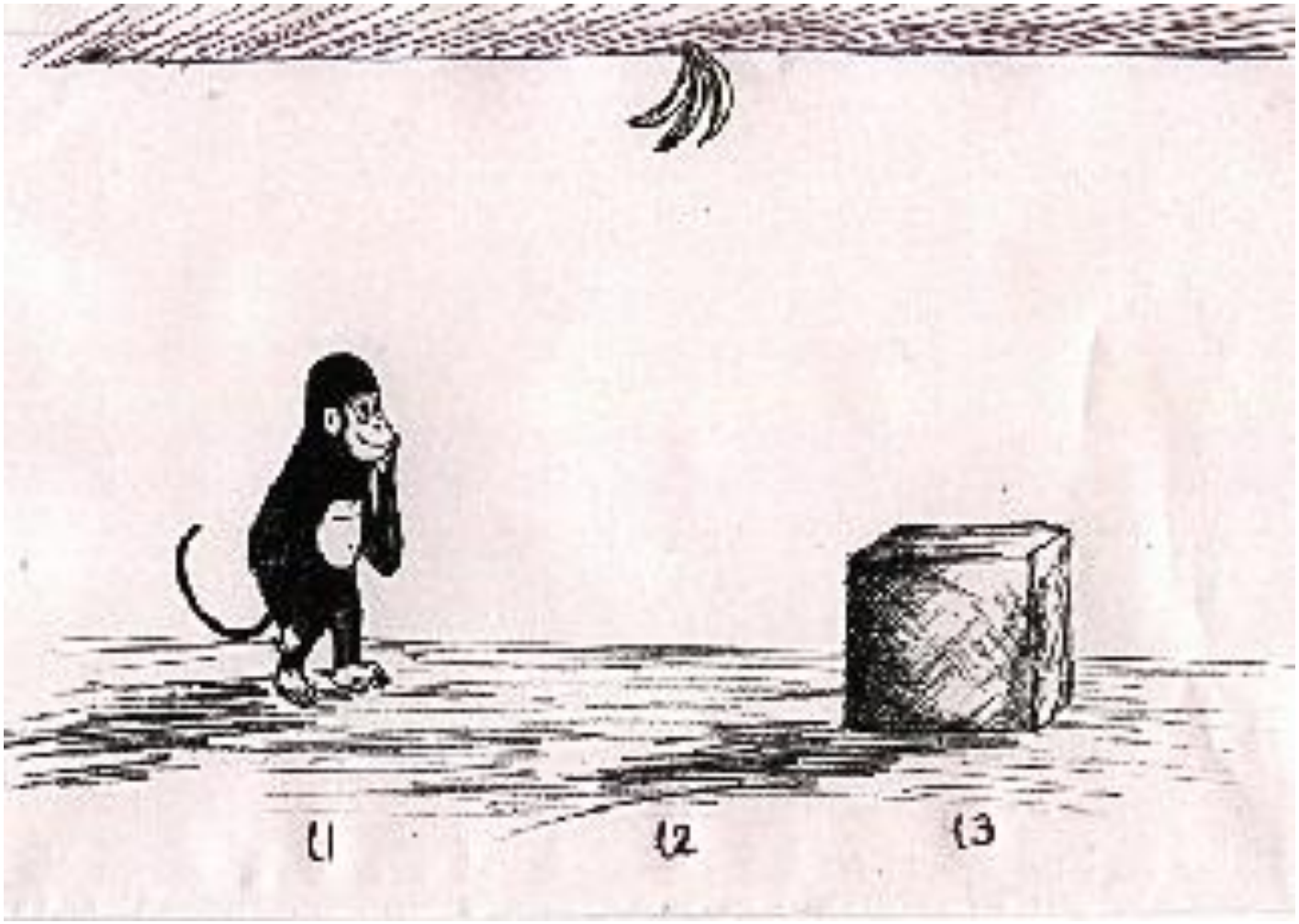
```
etat_initial([position(caisse, u1), position(bananes, u2),
position(singe,u3)]).

etat_final([possede(singe, bananes)]).

operateur(deplacer(P, Q), [position(singe, P)],
[position(singe, P)], [position(singe, Q)]) :- \+ P == Q.

operateur(pousser(U, V), [position(singe, U),
position(caisse, U)], [position(singe, U), position(caisse,
U)], [position(singe, V), position(caisse, V)]) :- \+ U == V.

operateur(cueillir_bananes(U), [position(singe, U),
position(caisse, U), position(bananes,U)], [],
[possede(singe, bananes)]).
```

Un autre problème

- **Nous cherchons à réaliser un robot qui aurait pu remplacer Panoramix et rendre des services à Astérix et aux autres Gaulois pour les aider à cueillir le gui sacré.**
- **Rappelons que pour cueillir du gui sacré, il faut :**

1. **une serpe d'or qui se trouve dans la maison du druide**
2. **une échelle qui se trouve chez le charpentier**
3. **le gui se trouve sur le chêne sacré**
4. **il faut ramener l'échelle et la serpe là où on les a trouvées**

Version gauloise du « singe et des bananes »



Règles STRIPS

Nom : prendre(P, X)

Pré condition :
position(Artus, P)
possede(P, X)

D_liste :
possede(P, X))

A_liste :
possede(Artus, X)

Contrainte: P \= Artus

Nom : rendre(P, X)

Pré condition :
position(Artus, P)
possede(Artus, X)

D_liste :
possede(Artus, X)

A_liste :
chez(X, P)

Contrainte: P \= Artus

Règles STRIPS

Nom : déplacer(P, P')

Pré condition :
position(Artus, P)

D_liste :
position(Artus, P)

A_liste :
position(Artus, P')

Contrainte: P \= P'

Nom : cueillir_gui

Pré condition :
position(Artus, chêne)
possede(Artus, échelle)
possede(Artus, cerpe)

D_liste :

A_liste :
possede(Artus, gui)

État initial et but

État initial :

possede(druide, cerpe)
possede(charpentier,
echelle),
position(Artus, Artus)

But:

possede(druide, cerpe)
possede(charpentier
echelle),
possede(Artus, gui)

État initial et but simplifié

État initial :

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, Artus)

But:

possede(Artus, gui)

Plan:

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, Artus)

- **but**

[possede(artus, gui)]

Plan:

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, Artus)

- **but**

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan:

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, Artus)

- **but**

position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan:

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, Artus)

- **but**

[position(artus,artus)],
operateur(deplacer(artus,chene)),
position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan:

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, Artus)

- **but**

operateur(deplacer(artus,chene)),
position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

[position(artus,chene),
possede(chene,echelle)],
operateur(prendre(chene,echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

position(artus,chene),
possede(chene,echelle)

[position(artus,chene),
possede(chene,echelle)],
operateur(prendre(chene,echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene))

Echec!

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

possede(chene,echelle)

[position(artus,chene),
possede(chene,echelle)],
operateur(prendre(chene,echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Retour arrière

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

[position(artus,charpentier),
possede(charpentier,echelle)],
operateur(prendre(charpentier,
echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),
[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

position(artus,charpentier),
possede(charpentier,echelle)
[position(artus,charpentier),
possede(charpentier,echelle)],
operateur(prendre(charpentier, echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

[position(artus,chene)],
operateur(deplacer(chene,charpentier))

position(artus,charpentier),
possede(charpentier,echelle)
[position(artus,charpentier),
possede(charpentier,echelle)],
operateur(prendre(charpentier, echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, chene)

- **but**

operateur(deplacer(chene,charpentier))

position(artus,charpentier),
possede(charpentier,echelle)
[position(artus,charpentier),
possede(charpentier,echelle)],
operateur(prendre(charpentier, echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, charpentier)

- **but**

position(artus,charpentier),
possede(charpentier,echelle)
[position(artus,charpentier),
possede(charpentier,echelle)],
operateur(prendre(charpentier, echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier))

- **Etat**

possede(druide, cerpe)
possede(charpentier, echelle),
position(Artus, charpentier)

- **but**

~~position(artus,charpentier),~~
~~possede(charpentier,echelle)~~
~~[position(artus,charpentier),~~
~~possede(charpentier,echelle)]~~,
operateur(prendre(charpentier, echelle))

possede(artus,echelle),
possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

~~possede(artus,echelle),~~
possede(artus,cerpe),
[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

position(artus,charpentier),
possede(charpentier,cerpe),

[position(artus,charpentier),
possede(charpentier,cerpe)],
operateur(prendre(charpentier,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

Echec!

possede(charpentier,cerpe),

[position(artus,charpentier), possede(charpentier,cerpe)], operateur(prendre(charpentier,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

Echec!

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

[position(artus,druide),
possede(druide,cerpe)],
operateur(prendre(druide,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

position(artus,druide),
possede(druide,cerpe)
[position(artus,druide),
possede(druide,cerpe)],
operateur(prendre(druide,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

[position(artus,charpentier)], operateur(deplacer(charpentier,druide))
position(artus,druide),
possede(druide,cerpe)
[position(artus,druide),
possede(druide,cerpe)],
operateur(prendre(druide,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, charpentier)

- **but**

~~**[position(artus,charpentier)],**~~
operateur(deplacer(charpentier,druide))
position(artus,druide),
possede(druide,cerpe)
[position(artus,druide),
possede(druide,cerpe)],
operateur(prendre(druide,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)), operateur(deplacer(chene,charpentier)), operateur(prendre(charpentier, echelle)) operateur(deplacer(charpentier,druide))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

position(artus,druide),
possede(druide,cerpe)
[position(artus,druide),
possede(druide,cerpe)],
operateur(prendre(druide,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))

- **Etat**

possede(druide, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

~~position(artus,druide),~~
~~possede(druide,cerpe)~~
~~[position(artus,druide),~~
~~possede(druide,cerpe)],~~
operateur(prendre(druide,cerpe))

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

possede(artus,cerpe),

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

[position(artus,druide)],
operateur(deplacer(druide,chene))

position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, druide)

- **but**

~~**[position(artus,druide)],**~~
operateur(deplacer(druide,chene))

position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))
operateur(deplacer(druide,chene))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, chene)

- **but**

position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)

[position(artus,chene),
possede(artus,echelle),
possede(artus,cerpe)],
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))
operateur(deplacer(druide,chene))

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, chene)

- **but**

~~position(artus,chene),~~
~~possede(artus,echelle),~~
~~possede(artus,cerpe)~~

~~[position(artus,chene),~~
~~possede(artus,echelle),~~
~~possede(artus,cerpe)],~~
operateur(cueillir_gui),
possede(artus,gui),[possede(artus,gui)]

L I P 6 C N R S

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))
operateur(deplacer(druide,chene))
operateur(cueillir_gui)

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, chene)

possede(artus, gui)

- **but**

possede(artus,gui),[possede(artus,gui)]

Plan: operateur(deplacer(artus,chene)),
operateur(deplacer(chene,charpentier)),
operateur(prendre(charpentier, echelle))
operateur(deplacer(charpentier,druide))
operateur(prendre(druide,cerpe))
operateur(deplacer(druide,chene))
operateur(cueillir_gui)

- **Etat**

possede(artus, cerpe)
possede(artus, echelle),
position(Artus, chene)

possede(artus, gui)

- **but**

Remarque: plan non optimal

Plan généré

deplacer(artus,chene),
deplacer(chene,charpentier),
prendre(charpentier, echelle)
deplacer(charpentier,druide)
prendre(druide,cerpe)
deplacer(druide,chene)
cueillir_gui

Plan optimal

deplacer(artus, charpentier),
prendre(charpentier, echelle)
deplacer(charpentier,druide)
prendre(druide,cerpe)
deplacer(druide,chene)
cueillir_gui

État initial et but

État initial :

possede(druide, cerpe)
possede(charpentier,
echelle),
position(Artus, Artus)

But:

possede(druide, cerpe)
possede(charpentier
echelle),
possede(Artus, gui)

Plan obtenu

1. deplacer(artus,chene),
2. deplacer(chene,charpentier),
3. prendre(charpentier,echelle),
4. deplacer(charpentier,druide),
5. prendre(druide,cerpe),
6. deplacer(druide,chene),
7. cueillir_gui,
8. deplacer(chene,druide),
9. rendre(druide,cerpe),
10. deplacer(druide,charpentier),
11. rendre(charpentier,echelle)