L I P 6 C N R S

Planification

ASP
Calcul des situations
Planification non linéaire

IAMSI

Intelligence Artificielle et Manipulation Symbolique de l'information

Cours 9

L I P 6 C N R S

Programmation planification en ASP

- **Représentation**
  - *objets*
  - *états*
- **Buts**
- **Mouvements**
  - *effets*
  - *inertie*
  - *contraintes*
- **Génération**

Programmation planification en ASP

- **Représentation**
  - *objets*
  - *états*
- **Buts**
- **Mouvements**
  - *effets*
  - *inertie*
  - *contraintes*
- **Génération**

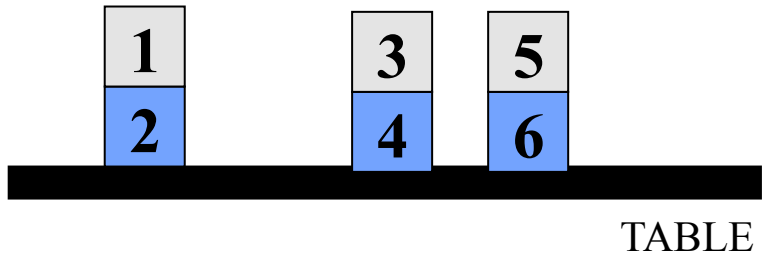
Programmation en ASP
Représentation des états

- **Les blocs et les instants sont représentés avec des nombres:**

```
block(1..6).
time(0..5).
```

- **Représenter les états avec des prédicats:**
  - **Etat initial: 0**

```
on(1, 2, 0).
on(2, table, 0).
on(3, 4, 0).
on(4, table, 0).
on(5, 6, 0).
on(6, table, 0).
```



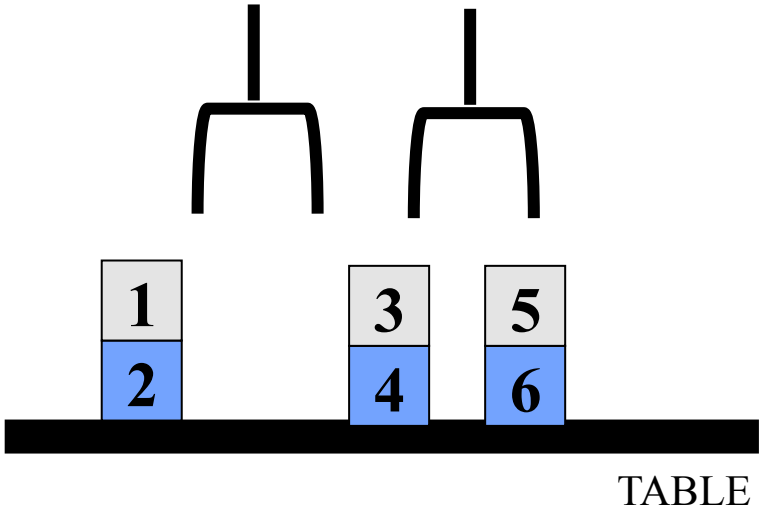
Programmation en ASP

Représentation des objets *(block, poignets)*

- **Les lieux sont représentés par des nombres. Il y en a au moins autant que de blocs... + la table**

```
block(1..6).
time(0..5).
location(B) ←
block(B).
location(table)
```

- **Nombre de poignets: variable...**

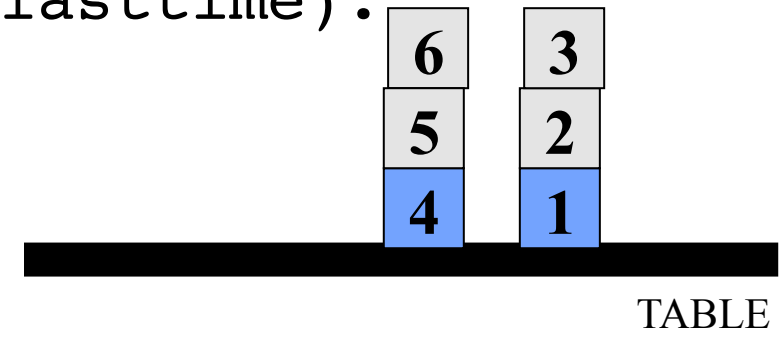


Programmation planification en ASP

- Représentation
  - *objets*
  - *états*
- **Buts**
- Mouvements
  - *effets*
  - *inertie*
  - *contraintes*
- Génération

But

```
:- not on(3, 2, lasttime).
:- not on(2, 1, lasttime).
:- not on(1, table, lasttime).
:- not on(6, 5, lasttime).
:- not on(5, 4, lasttime).
:- not on(4, table, lasttime).
```



Programmation planification en ASP

- Représentation
  - *objets*
  - *états*
- Buts
- **Mouvements**
  - *effets*
  - *inertie*
  - *contraintes*
- Génération

Mouvements

- **Effet du mouvement d'un bloc**

```
on(B, L, T+1) :- move(B, L, T), T < lasttime.
```

- **Inertie**

```
on(B, L, T+1) :-
      on(B, L, T),
      not -on(B, L, T+1),
      T < lasttime.
```

- **Position unique de chaque bloc**

```
-on(B, L1, T+1) :- on(B, L, T+1),
                   neq(L, L1).
```

Contraintes

- **Deux blocs ne peuvent pas se trouver au-dessus d'un même bloc**

```
:- 2 {on(BB, B, T) : block(BB)}.
```

- **Un bloc ne peut être mu si quelque chose se trouve au-dessus**

```
:- move(B, L, T), on(B1, B, T).
```

- **Un bloc ne peut pas être mu sur un bloc qui est en mouvement**

```
:- move(B, B1, T), move(B1, L, T).
```

Programmation planification en ASP

- **Représentation**
  - *objets*
  - *états*
- **Buts**
- **Mouvements**
  - *effets*
  - *inertie*
  - *contraintes*
- **Génération**

Génération avec n poignets

```
{move(BB, LL, T):block(BB):location(LL)}1 :-
                    T < lasttime.
```

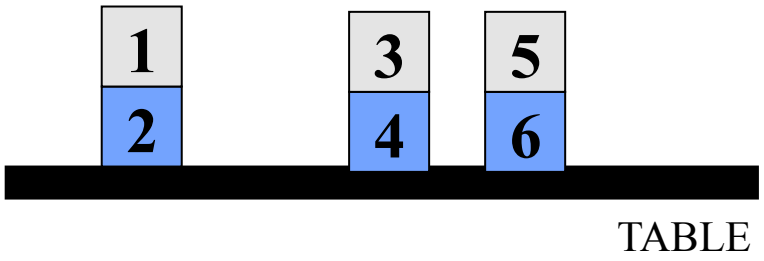
Programmation en ASP

- **Les blocs et les instants sont représentés avec des nombres:**

```
block(1..6).
time(0..5).
```

- **Représenter les états avec des prédicats:**
  - **Etat initial: 0**

```
on(1, 2, 0).
on(2, table, 0).
on(3, 4, 0).
on(4, table, 0).
on(5, 6, 0).
on(6, table, 0).
```



Programmation en ASP *(suite)*

- **Les lieux sont représentés par des nombres. Il y en a au moins autant que de blocs... + la table**

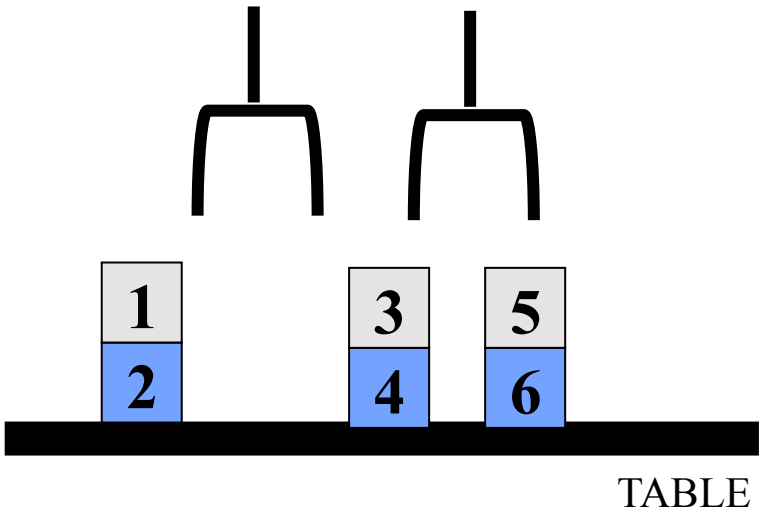
```
block(1..6).

time(0..5).

location(B) ←
block(B).

location(table)
```

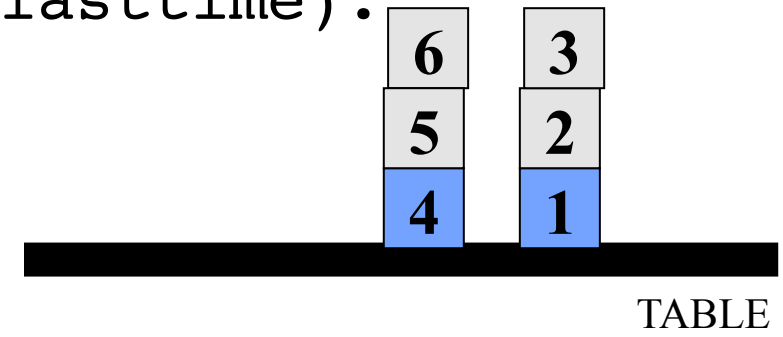
- **Nombre de poignets: variable...**



But

```
:- not on(3, 2, lasttime).
:- not on(2, 1, lasttime).
:- not on(1, table, lasttime).
:- not on(6, 5, lasttime).
:- not on(5, 4, lasttime).
:- not on(4, table, lasttime).
```

Fin cours 6



Génération avec n poignets

AnsProlog* **- 2 poignets**

```
{move(BB, LL, T):block(BB):location(LL)}2 :-
                  T < lasttime.
```

Clingo - 2 poignets

```
{move(BB, LL, T):block(BB),location(LL)}=2 :-
                  T < lasttime.
```

Mouvements – AnsProlog*

- **Effet du mouvement d'un bloc**

```
on(B, L, T+1) :- move(B, L, T), T < lasttime.
```

- **Inertie**

```
on(B, L, T+1) :-
      on(B, L, T),
      not -on(B, L, T+1),
      T < lasttime.
```

- **Position unique de chaque bloc**

```
-on(B, L1, T+1) :- on(B, L, T+1),
                   neq(L, L1).
```

Mouvements – Clingo

- **Effet du mouvement d'un bloc**

```
on(B, L, T+1) :- move(B, L, T),
block(B), time(T), location(L), T < lasttime.
```

- **Inertie**

```
on(B, L, T+1) :- block(B), time(T),
      location(L),
      on(B, L, T),
      not -on(B, L, T+1),
      T < lasttime.
```

- **Position unique de chaque bloc**

```
-on(B, L1, T+1) :- block(B), time(T),
                   location(L), location(L1),
                   on(B, L, T+1), L != L1.
```

Contraintes – AnsProlog*

- **Deux blocs ne peuvent pas se trouver au-dessus d'un même bloc**

```
:- 2 {on(BB, B, T) : block(BB)}.
```

- **Un bloc ne peut être mu si quelque chose se trouve au-dessus**

```
:- move(B, L, T), on(B1, B, T).
```

- **Un bloc ne peut pas être mu sur un bloc qui est en mouvement**

```
:- move(B, B1, T), move(B1, L, T).
```

Contraintes - Clingo

- **Deux blocs ne peuvent pas se trouver au-dessus d'un même bloc**

```
:- 2 {on(BB, B, T) : block(BB)},
     block(B), time(T).
```

- **Un bloc ne peut être mu si quelque chose se trouve au-dessus**

```
:- move(B, L, T), on(B1, B, T),
block(B), block(B1), location(L), time(T).
```

- **Un bloc ne peut pas être mu sur un bloc qui est en mouvement**

```
:- move(B, B1, T), move(B1, L, T),
block(B), block(B1), location(L), time(T).
```

Anomalies de Susmann avec ASP

- **Etat initial**

```
on(3, 1, 0).
on(1, table, 0).
on(2, table, 0).
```

- **But**

```
:- not on(2, 3, lasttime).
:- not on(1, 2, lasttime).
:- not on(3, table,
     lasttime).
```