Game Description Language

Tristan Cazenave

Programmation des Jeux

Tristan.Cazenave@dauphine.fr

Game Description Language

- Utilisation d'une représentation logique du premier ordre pour représenter les jeux.
- Jeux à deux joueurs, multi-joueurs et à un joueur.
- 1968 : Jacques Pitrat et le General Game Playing.
- Années 70 : Prolog, Strips => langage pour faire des plans et résoudre des problèmes.

Game Description Language

- 1997 : Deep Blue bat Kasparov aux échecs mais il ne sait pas jouer à d'autres jeux.
- 2005 : Michael Genesereth propose le GDL et des compétitions de GGP.
- On donne aux programmes la description GDL d'un jeu juste avant d'y jouer.

Game Description Language

- On décrit un problème avec des états et des actions qui permettent de passer d'un état à un autre.
- On cherche des algorithmes généraux pour résoudre des problèmes ou jouer à des jeux.
- 2005-2006 : algorithmes classiques d'IA, A*, Alpha-Beta.
- 2007-… : recherche Monte Carlo

Game Description Language

- Principe de la recherche Monte-Carlo :
- On joue des parties aléatoires.
- On fait des statistiques sur les résultats de ces parties.
- On joue le coup qui a la meilleure moyenne.

Game Description Language

- Description d'un jeu (TicTacToe) :
- On définit les joueurs (black et white) :
- On utilise le prédicat role

  role(white)

  role(black)

Game Description Language

- Etat initial du problème :

  init(cell(1,1,b))

  init(cell(1,2,b))

  init(cell(1,3,b))

  init(cell(2,1,b))

  ...

  init(cell(3,3,b))

  init(control(white))

Game Description Language

- Coups légaux :
- noop : action qui ne fait rien
- En GDL comme en Prolog les variables commencent par une majuscule et les constantes par une minuscule.
- Opérateur :- est équivalent à si

Game Description Language

- Coups légaux :

```
legal(W,mark(X,Y)) :- true(cell(X,Y,b)) &
                      true(control(W))

legal(white,noop) :- true(cell(X,Y,b)) &
                     true(control(black))

legal(black,noop) :- true(cell(X,Y,b)) &
                     true(control(white))
```

Game Description Language

- On utilise le prédicat next pour passer d'un état au suivant :

```
next(cell(M,N,white)) :- does(white,mark(M,N))
                         & true(cell(M,N,b))

next(cell(M,N,black)) :- does(black,mark(M,N))
                         & true(cell(M,N,b))
```

Game Description Language

```
next(cell(M,N,W)) :- true(cell(M,N,W)) &
                     distinct(W,b)
next(cell(M,N,b)) :- does(W,mark(J,K)) &
                     true(cell(M,N,b)) &
                     distinct(M,J)
next(cell(M,N,b)) :- does(W,mark(J,K)) &
                     true(cell(M,N,b)) &
                     distinct(N,K)
```

Game Description Language

```
next(control(white)) :-
        true(control(black))

next(control(black)) :-
        true(control(white))
```

Game Description Language

- On utilise le prédicat goal pour associer une valeur à un état.
- Par convention 100 veut dire gagné :

  goal(white,100) :- line(white)

  goal(white,0) :- ~line(white) & line(black)

  goal(white, 50) :- ~line(black) & ~line (white)

  goal(black, 100) :- line(black)

  goal(black, 0) :- ~line(black) & line (white)

  goal(black, 50) :- ~line(black) & ~line (white)

Game Description Language

- On définit le prédicat line :

  line(X) :- row(M,X)

  line(X) :- column(M,X)

  line(X) :- diagonal(X)

Game Description Language

- On utilise le prédicat terminal pour décrire les états terminaux du jeu :

  terminal :- line(black)

  terminal :- line(white)

  terminal :- ~open

  open :- true(cell(X,Y,b))

Game Description Language

- Exercice :

  Ecrire les prédicats row, column et diagonal.

Game Description Language

- Prédicats utilisés en GDL :

  legal, next, goal, terminal, true, does, init
- action(a) : a est une action du jeu
- base(p) : p est une proposition de base du jeu

Game Description Language

```
role(white)
role(black)

base(p)
base(q)
base(r)
base(s)

action(a)
action(b)
action(c)
action(d)

init(s)

legal(white,a)
legal(white,b)
legal(white,c)
legal(black,d)
```

Game Description Language

next(p) :- does(white,a) & ~true(p)

next(p) :- ~does(white,a) & true(p)

next(q) :- does(white,b) & true(p)

next(q) :- does(white,c) & true(r)

next(q) :- ~does(white,b) & ~does(white,c) & true(q)

next(r) :- does(white,c) & true(q)

next(r) :- ~does(white,c) & true(r)

goal(white,100) :- terminal

goal(white,0) :- ~terminal

goal(black,100) :- terminal

goal(black,0) :- ~terminal

terminal :- true(p) & true(q) & true(r)

Game Description Language

- Combien de rôles y a-t-il ?
- Combien de propositions y a-t-il ?
- Combien d'actions faisables y-a-t il ?
- Combien d'actions légales pour white dans l'état initial ?
- Combien de propositions vraies dans l'état initial ?
- Combien sont vraies dans l'état qui résulte de white a et black d ?
- Quel est le nombre minimal de coups pour que le jeu se termine ?

Game Description Language

Graphe d'états du problème suivant :

```
role(robot)
legal(robot,a)
legal(robot,b)
next(p) :- does(robot,a) & ~true(p)
next(p) :- does(robot,b) & true(q)
next(q) :- does(robot,a) & true(q)
next(q) :- does(robot,b) & true(p)
terminal :- true(p) & true(q)
goal(robot,100) :- true(p) & true(q)
```

Game Description Language

Graphe d'états du problème suivant :

```
role(player)

index(1)
index(2)
index(3)

base(on(X)) :- index(X)

legal(player,toggle(X)) :- index(X)

next(on(X)) :- ~true(on(X)) & does(player,toggle(X))
next(on(X)) :- true(on(X)) & ~does(player,toggle(X))

terminal :- true(on(1)) & true(on(2)) & true(on(3))
goal(player,100) :- true(on(1)) & true(on(2)) & true(on(3))
```

Game Description Language

Graphe d'états du problème suivant :

role (you)

opposite (left,right)

opposite (right,left)

init (you (left))

init (at (c, left))

init (at (g, left))

init (at (w, left))

legal (you, take (X)):- true (you (Side)) & true (at (X, Side))

legal (you , take (nothing))

Game Description Language

```
next (you (New)) :-
    true (you (Old)) &
    opposite (New, Old)

next (at (X, New)) :-
    does (you, take (X)) &
    true (at (X, Old)) &
    opposite (New, Old))

next (at (X, Old)) :-
    true (at (X, Old)) &
    ~does (you, take (X))
```

Game Description Language

terminal :- good

terminal :- bad

goal (you, 100) :- good

goal (you, 0) :- bad

Game Description Language

```
good :-
    true (at (c, right)) &
    true (at (g, right)) &
    true (at (w, right))

bad :-
    true (at (c, Side)) &
    true (at (g, Side)) &
    ~true (you (Side))
bad :-
    true (at (g, Side)) &
    true (at (w, Side)) &
    ~true (you (Side))
```

Game Description Language

- Dessiner le graphe d’états du problème
- Donner une solution qui permet d’atteindre la récompense de 100.

Game Description Language

- In competitions, game rules are usually obfuscated in order to hide their meaning.
- Try to figure out what the following game is about.
- Find a sequence of moves to win the game.

Game Description Language

role(you)

succ(1,2)
succ(2,3)
succ(3,4)
succ(4,5)
succ(5,6)
succ(6,7)
succ(7,8)

init(cell(1,1))
init(cell(Y,1)) <= succ(X,Y) ∧ init(cell(X,1))
init(step(1))

legal(you,jump(X,Y)) <= true(cell(X,1)) ∧ true(cell(Y,1)) ∧ (gillard(X,Y) ∨ gillard(Y,X))

oakeshott(X,Y) <= succ(X,Y)
oakeshott(X,Y) <= succ(X,Z) ∧ true(cell(Z,0)) ∧ oakeshott(Z,Y)

Game Description Language

abbott(X,Y) <= succ(X,Z) ∧ true(cell(Z,0)) ∧ abbott(Z,Y)
abbott(X,Y) <= succ(X,Z) ∧ true(cell(Z,1)) ∧ oakeshott(Z,Y)

gillard(X,Y) <= succ(X,Z) ∧ true(cell(Z,0)) ∧ gillard(Z,Y)
gillard(X,Y) <= succ(X,Z) ∧ true(cell(Z,1)) ∧ abbott(Z,Y)
gillard(X,Y) <= succ(X,Z) ∧ true(cell(Z,2)) ∧ oakeshott(Z,Y)

next(step(Y)) <= true(step(X)) ∧ succ(X,Y)
next(cell(X,0)) <= does(you,jump(X,Y))
next(cell(Y,2)) <= does(you,jump(X,Y))
next(cell(X,C)) <= does(you,jump(Y,Z)) ∧ true(cell(X,C)) ∧ distinct(X,Y) ∧ distinct(X,Z)

terminal <= ~continuable
continuable <= legal(you,M)

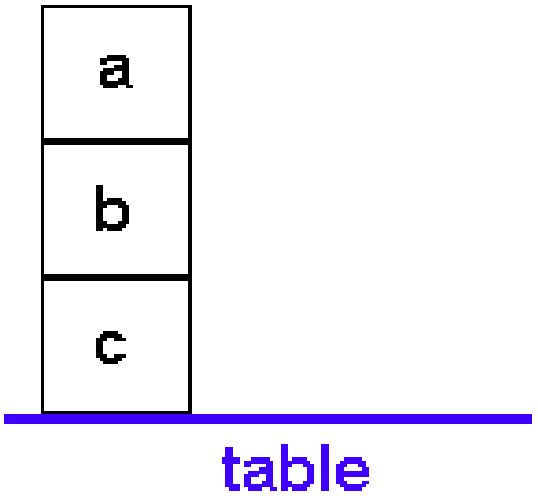
goal(you,100) <= true(step(5))
goal(you, 0) <= true(cell(X,1))

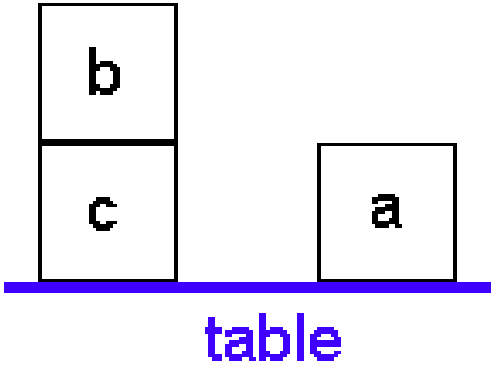
Game Description Language

- Misère Nim :
- On a un tas de 6 allumettes.
- Chaque joueur à son tour prend 1, 2 ou 3 allumettes.
- Le dernier qui joue a perdu.
- Ecrire le graphe d'états du jeu.
- Modéliser ce jeu en GDL.

Game Description Language

- Le monde des cubes :





Game Description Language

- Le monde des cubes :
- On a un état initial du problème qui consiste en une configuration des cubes sur la table.
- On souhaite arriver à un état final.
- Modéliser le problème pour 6 cubes que l'on souhaite empiler dans l'ordre et qui sont au départ tous isolés sur la table.

Game Description Language

- Un appartement comporte un salon, une cuisine et un placard. Les trois pièces sont reliées deux à deux. Un singe se trouve dans le salon et un régime de bananes est accroché au plafond de la cuisine. Une caisse est posée au sol dans le placard.
- Trouver une suite d'actions du singe affamé pour attraper les bananes sachant que pour atteindre son but il doit monter sur la caisse.

Game Description Language

- Aller dans le placard
- Prendre la caisse
- Aller dans la cuisine avec la caisse
- Monter sur la caisse
- Attraper les bananes

Game Description Language

- Ecrire la base de faits initiale sachant que le singe se nomme cheeta, la caisse ks et le régime de bananes split.
- Décrire le problème en GDL.