



Wang, Yingjian

Think About Hanoi Game Declaratively

Dresden, 16.11.2021

Motivation

We want to achieve a model for Hanoi game

- disks: a, b, c
- size: $a > b > c$
- pillar: p_1, p_2, p_3
- initial state: $\text{on}(c, b), \text{on}(b, a), \text{on}(a, \text{ground}), \text{top}(c, p_1)$
- goal state: $\text{on}(c, b), \text{on}(b, a), \text{on}(a, \text{ground}), \text{top}(c, p_n)$ for $n \neq 1$
- constraint: $\text{on}(X_1, X_2) \rightarrow \text{bigger}(X_1, X_2)$ for $X_1, X_2 \in \{a, b, c\}$

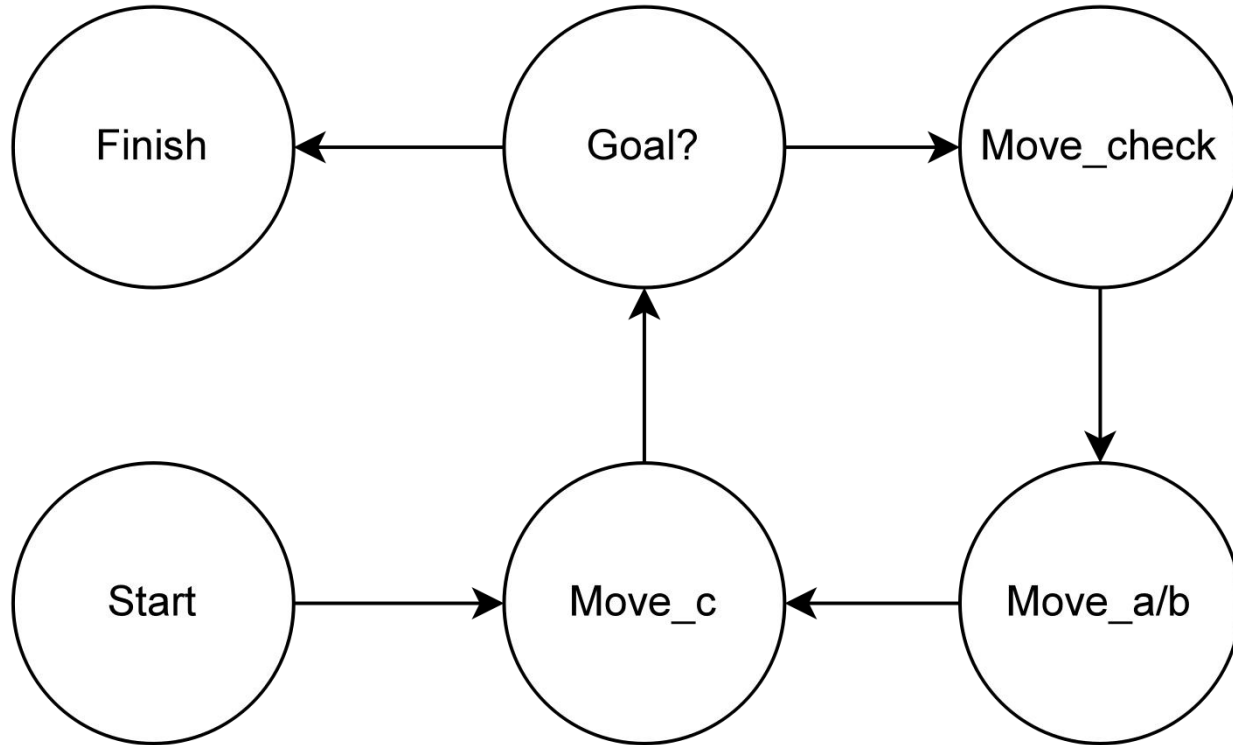
Hanoi Game

Non-recursive Algorithm

- Define move sequence $\text{order}(p_1, p_2, p_3)$ for c
 - take one move for c in order
 - take one move for a or b (always only one disk is able to move)
 - repeat the previous 2 steps till a, b, c moved to another pillar
- Eg. $\text{move}(a, p_2), \text{move}(b, p_3), \text{move}(a, p_3), \text{move}(c, p_2),$
 $\text{move}(a, p_1), \text{move}(b, p_2), \text{move}(a, p_2), \text{done}$

Hanoi Game

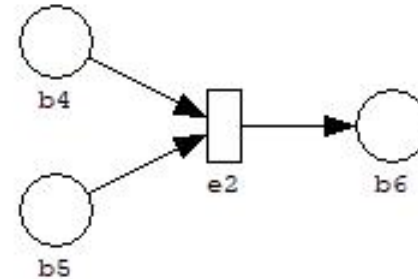
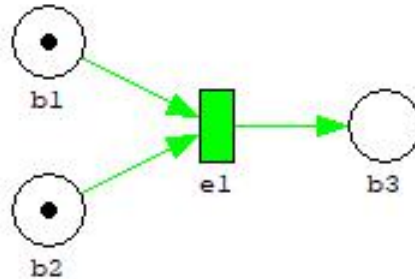
State machine



Hanoi Game

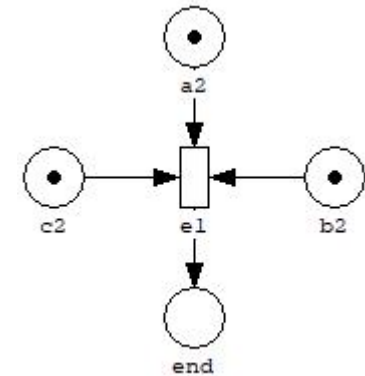
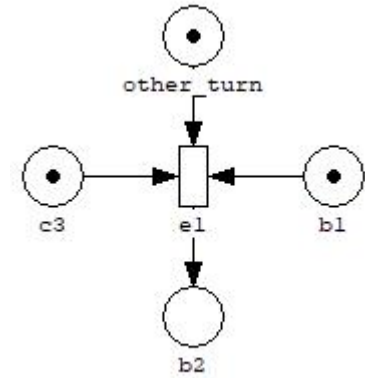
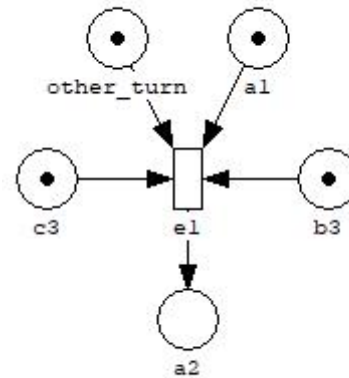
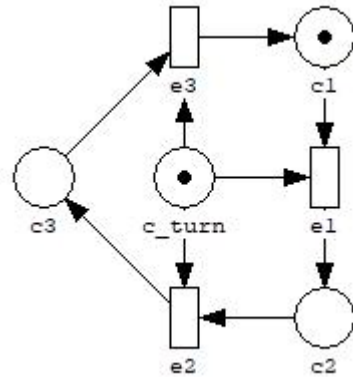
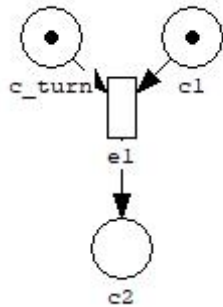
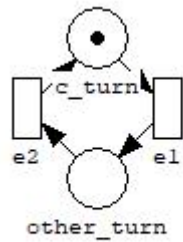
Petri-net(condition event net)

- condition: circle with token inside(or not)-->condition satisfied(or not)
- event: rectangle, fire when needed conditions are satisfied



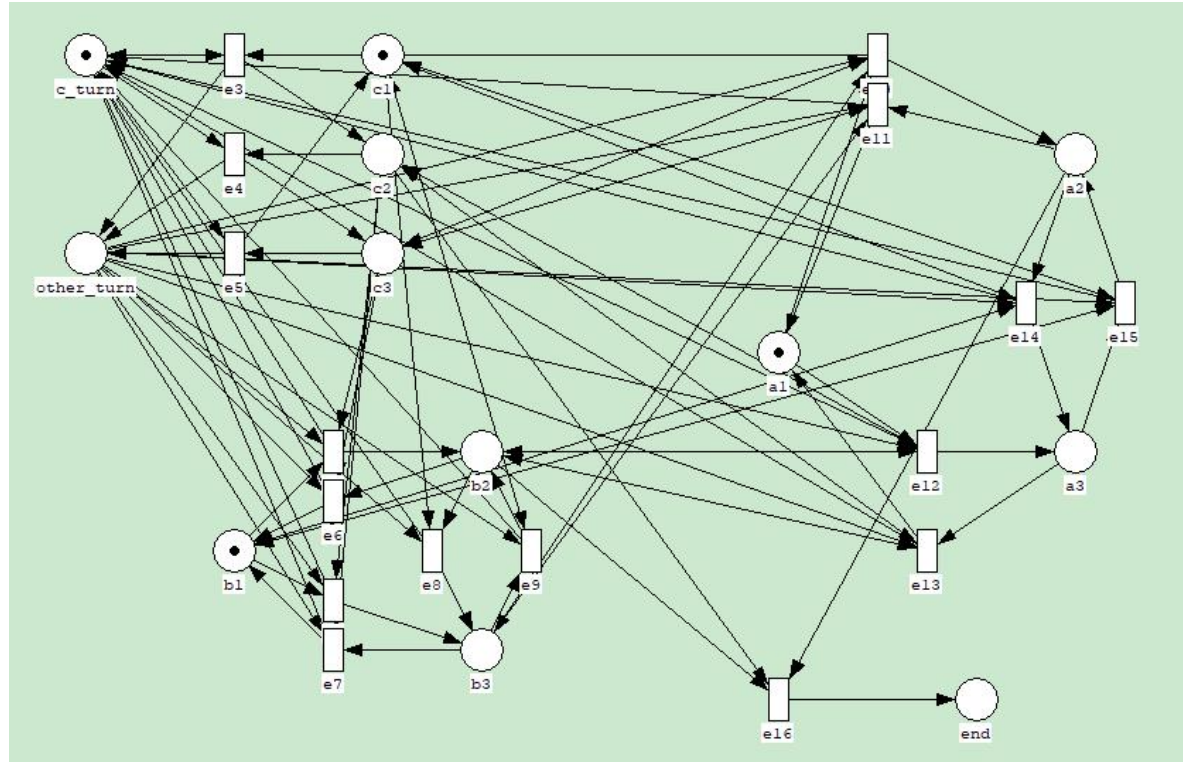
Hanoi Game

Petri-net solution(part)



Hanoi Game

Petri-net solution



Hanoi Game

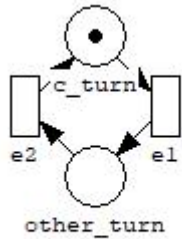
Prolog

- facts:
 - `disk(a)... pillar(p1)...bigger(a,b)... on(b,a)...`
 - `top(c,p1), top(ground,p2), top(ground,p3),`
 - `order(p1,p2), order(p2,p3), order(p3,p1).`
- rules:
 - `canmove(X,P):-disk(X),top(X,P1),top(Y,P),bigger(Y,X).`
 - `chec(P1,X2):-neighbour(P1,P2),top(X2,P2),disk(X2).`
 -

Hanoi Game

Prolog rules

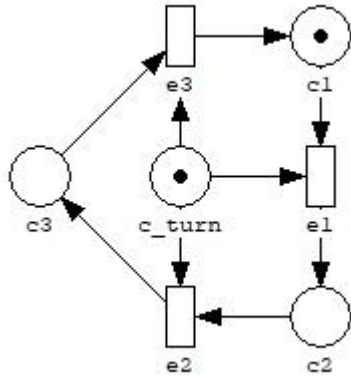
- `hanoi2(X):-move3(X),hanoi1(c).`
- `hanoi1(X):-move2(X),top(X,P1),chec(P1,X2),hanoi2(X2).`



Hanoi Game

Prolog rules

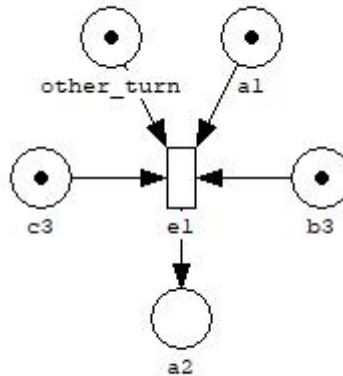
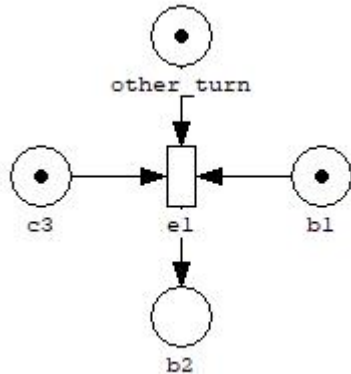
- `move2(X):-top(X,P1),order(P1,P2),move(X,P2).`
- `hanoi1(X):-move2(X),top(X,P1),chec(P1,X2),hanoi2(X2).`



Hanoi Game

Prolog rules

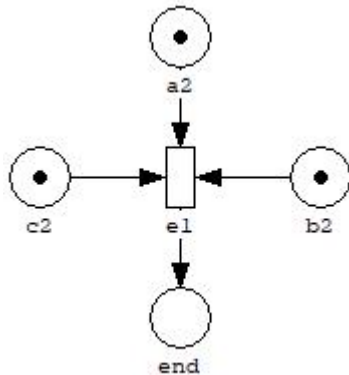
- `hanoi1(X):-move2(X),top(X,P1),chec(P1,X2),hanoi2(X2).`
- `chec(P1,X2):-neighbour(P1,P2),top(X2,P2),disk(X2).`
- `canmove(X,P):-disk(X),top(X,P1),top(Y,P),bigger(Y,X).`



Hanoi Game

Prolog rules

- `hanoi1(X):-move2(X),top(X,P1),chec(P1,X2),hanoi2(X2).`
- `hanoi1(X):-true.`



Hanoi Game

Prolog queries

• ?- on(X,Y).	?- top(X,P).		?- on(X,Y).	?- top(X,P).
• X = b,	X = c,		X = c,	X = c,
• Y = a ;	P = p1;		Y = b;	P = p2;
• X = c,	X = ground,	?- hanoi1(c).	X = b,	X = ground,
• Y = b ;	P = p2;	true.	Y = a;	P = p1;
• X = a,	X = ground,		X = a,	X = ground,
• Y = ground.	P = p3.		Y = ground;	P = p3.

Hanoi Game

Jastadd(more need to do)

- `Hanoi::=Disk*`;
- `Disk::=<Pillar:int> <Label:String> <Size:int> <Top:int> <On:String>;`

Thank you for your attention!
Vielen Dank für Ihre Aufmerksamkeit!