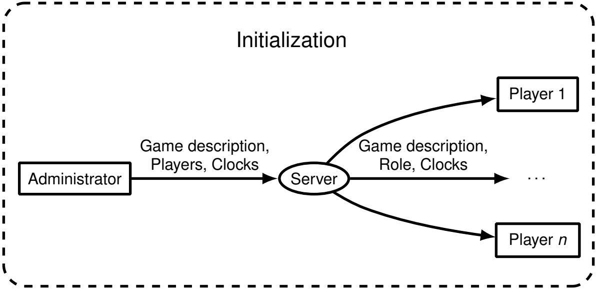
General Game Playing and the Game Description Language

Abdallah Saffidine

University of New South Wales

December, 16

GENERAL
GAME
PLAYING



Initialization

Administrator

Game description, Players, Clocks

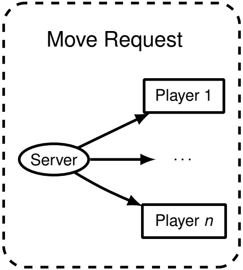
Server

Game description, Role, Clocks

Player 1

. . .

Player *n*



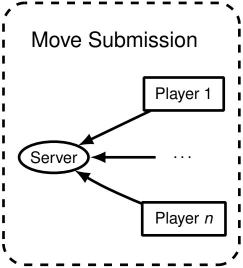
Move Request

Server

Player 1

. . .

Player *n*



Move Submission

Server

Player 1

. . .

Player *n*

Another Kind of Logic Language

GDL is not Prolog

- No cuts
- Hypotheses are not ordered
- Forward and backward chaining are possible
- Grounding

GDL is not Datalog

- Nested function symbols
- Recursion restriction

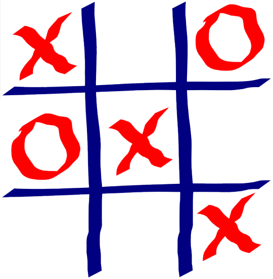
- Dynamic keywords: `init`, `true`, `next`
- Semantics = Transition system

Initial State

```
(init (cell 1 1 b))
 ...
(init (cell 3 3 b))
(init (ctrl xplayer))
```

Legal Actions

```
(← (legal P (mark M N))
   (true (ctrl P))
   (true (cell M N b)))
```



Next State

```
(← (next (cell M N P))
   (does P (mark M N)))
(← (next (ctrl oplayer))
   (true (ctrl xplayer)))
(← (next (ctrl xplayer))
   (true (ctrl oplayer)))
```

```
(← (next (cell M N C))
   (true (cell M N C))
   (does P (mark M′ N′))
   (or (distinct M M′)
       (distinct N N′)))
```

Auxiliary predicates

```
(← (line P) (true (cell M 1 P)) (true (cell M 2 P)) (true (cell M 3 P)))
(← (line P) (true (cell 1 N P)) (true (cell 2 N P)) (true (cell 3 N P)))
(← emptycell (true (cell M N b)))
```

Objective

```
(← (goal xplayer 100) (line xplayer))
(← (goal oplayer   0) (line xplayer))
```

Termination

```
(← terminal (line P))
(← terminal (not emptycell))
```

Typical competition time limits

- Startclock: 600 sec per match
- Playclock: 30 sec per move

Making the most out of the competition settings

- GDL is not Prolog → Hypotheses are not ordered
- Startclock → Optimize the compiler's parameters

Into perspective

- Classical compilers: globally-tune default flags.
- Automatic empirical optimization (ATLAS/FFTW3): tune to the hardware/installed software.
- Here: tune to the instance (also [Keller *et al.*, 2008]).

Hypotheses Ordering

Rule	Input	Restriction
(← (legal white (move X_1 Y_1 X_2 Y_2))		
(succ Y_1 Y_2)	Y_1, Y_2	
(succ X_1 X_2)	X_1, X_2	
(cell X_1 Y_1 w)		X_1, Y_1
(not (cell X_2 Y_2 w)))		X_2, Y_2

Hypotheses Ordering

Rule	Input	Restriction
(← (legal white (move X_1 Y_1 X_2 Y_2))		
(succ Y_1 Y_2)	Y_1, Y_2	
(succ X_1 X_2)	X_1, X_2	
(cell X_1 Y_1 w)		X_1, Y_1
(not (cell X_2 Y_2 w)))		X_2, Y_2
(← (legal white (move X_1 Y_1 X_2 Y_2))		
(succ Y_1 Y_2)	Y_1, Y_2	
(cell X_1 Y_1 w)	X_1	Y_1
(succ X_1 X_2)	X_2	X_1
(not (cell X_2 Y_2 w)))		X_2, Y_2

Knowledge Base

{(succ 1 2), (succ 2 3), (succ 3 4), (cell 1 1 w), (cell 1 2 w), (cell 2 2 w), . . .}

Steps	Satisfying assignments: {[X_1 X_2 Y_1 Y_2]}
(succ Y_1 Y_2)	{[_ _ 1 2], [_ _ 2 3], [_ _ 3 4]}
(succ X_1 X_2)	{[1 2 1 2], [2 3 1 2], [3 4 1 2], [1 2 2 3], [2 3 2 3], [3 4 2 3], [1 2 3 4], [2 3 3 4], [3 4 3 4]}
(cell X_1 Y_1 w)	{[1 2 1 2], [1 2 2 3], [2 3 2 3]}
(not (cell X_2 Y_2 w)))	{[1 2 2 3], [2 3 2 3]}
(succ Y_1 Y_2)	{[_ _ 1 2], [_ _ 2 3], [_ _ 3 4]}
(cell X_1 Y_1 w)	{[1 _ 1 2], [1 _ 2 3], [2 _ 2 3]}
(succ X_1 X_2)	{[1 2 1 2], [1 2 2 3], [2 3 2 3]}
(not (cell X_2 Y_2 w)))	{[1 2 2 3], [2 3 2 3]}

Knowledge Base

{(succ 1 2), (succ 1 3), (cell 1 1 w), (cell 2 1 w), (cell 3 1 w), (cell 4 1 w), . . .}

Steps	Satisfying assignments: {[X_1 X_2 Y_1 Y_2]}
(succ Y_1 Y_2)	{[_ _ 1 2], [_ _ 1 3]}
(succ X_1 X_2)	{[1 2 1 2], [1 3 1 2], [1 2 1 3], [1 3 1 3]}
(cell X_1 Y_1 w)	{[1 2 1 2], [1 3 1 2], [1 2 1 3], [1 3 1 3]}
(not (cell X_2 Y_2 w)))	{[1 2 1 2], [1 3 1 2], [1 2 1 3], [1 3 1 3]}
(succ Y_1 Y_2)	{[_ _ 1 2], [_ _ 1 3]}
(cell X_1 Y_1 w)	{[1 _ 1 2], [2 _ 1 2], [3 _ 1 2], [4 _ 1 2], [1 _ 1 3], [2 _ 1 3], [3 _ 1 3], [4 _ 1 3]}
(succ X_1 X_2)	{[1 2 1 2], [1 3 1 2], [1 2 1 3], [1 3 1 3]}
(not (cell X_2 Y_2 w)))	{[1 2 1 2], [1 3 1 2], [1 2 1 3], [1 3 1 3]}

Empirical hypotheses ordering

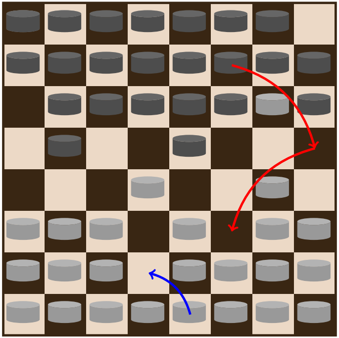
- Get the rules
- Naive compilation
- Collect data via random games
- Infer a good hypotheses ordering
- → Smart compilation

Empirical hypotheses ordering

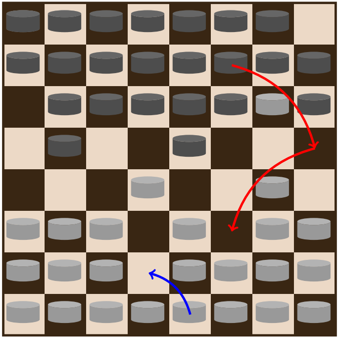
- Get the rules
- Naive compilation
- Collect data via random games
- Infer a good hypotheses ordering
- → Smart compilation

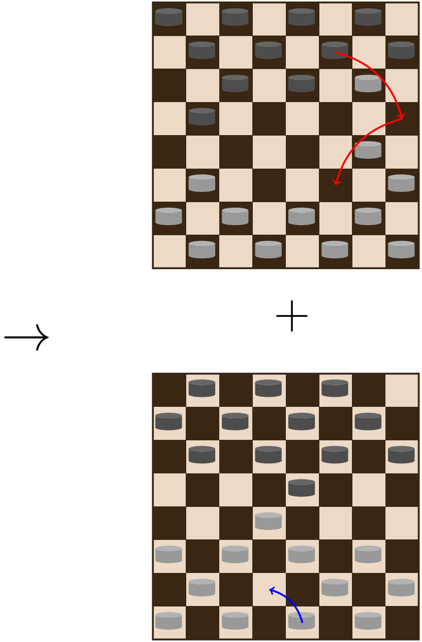
Experiments: engine speed (higher is better)

Game	Original	Inferred	Improvement
Peg solitaire	3.6	139	3800%
Connect 4	78.7	111	141%
Mini-chess	62.7	74.8	119%
Ro-sham-bo	1,480	1,610	109%
Sheep and Wolf	101	94.3	93%



Language-level: Decomposition into sub-games





AI level: Complexity reduction

$(b_1 b_2)^n \to b_1^n + b_2^n$

Reachability Analysis

What for?

- Deadlock analysis
- Termination analysis
- Precise static typing
- Dead-code detection

Reachability Analysis

What for?

- Deadlock analysis
- Termination analysis
- Precise static typing
- Dead-code detection

- Prop: no variables
- Mono: facts are never removed from the KB
- Bounded: stronger recursion restriction

Fragment	Reachability
Prop, Mono	NP-c
Prop	PSPACE-c
Mono	NEXPTIME-c
Bounded	EXPSPACE-c
Full	UNDEC

Conclusion

In the AI community: a Hot Topic

- Yearly international competition
- Journal special issues, workshops
- Masters students, PhDs
- Massive OO Course ($\geq$ 50,000 students)

Conclusion

In the AI community: a Hot Topic

- Yearly international competition
- Journal special issues, workshops
- Masters students, PhDs
- Massive OO Course ($\geq$ 50,000 students)

In the PL community: We need You!

- Low hanging fruits
- Some interesting problems?