Decomposition of Multi-Player Games

Dengji ZHAO[2]
Supervisor:
Dipl.-Inf. Stephan SCHIFFEL[1]
Prof. Dr. Michael THIELSCHER[1]

[1] Computational Logic Group
Artificial Intelligence Institute

[2] European Master's Program in Computational Logic

Master Thesis

Outline

Outline

General Game Playing

- Time constraints **VS** Very large games

General Game Playing

- Time constraints **VS** Very large games
- Games contain subgames
- Solve a game by solving its subgames?

Previous Work

- *Decomposition of Single Player Games*,
  M. Günther, S. Schiffel and M. Thielscher, 2007

Previous Work

- *Decomposition of Single Player Games*,
  M. Günther, S. Schiffel and M. Thielscher, 2007
- What about multi-player games?

Properties of Multi-Player Games

- Alternating Move Games, *e.g. Nim, Chess, TicTacToe*
- Simultaneous Move Games, *e.g. Rock-paper-scissors*
- Impartial Games, *e.g. Nim*
- Partial Games, *e.g. TicTacToe, Double-TicTacToe*
- Parallel Games, *e.g. Parallel-TicTacToe*
- Serial Games, *e.g. Serial-TicTacToe*

Outline

Basic Definitions

Definition

(Game). A **game** is a tuple $G = (F, A, I, R)$ where

- F is a set of fluents,
- A is a set of actions,
- I is the initial state of the game, which is a set of ground instances of F,
- R is a set of roles.

Definition

(State). A **state** S of a game $G = (F, A, I, R)$ is a set of ground instances of F, and S can be reached from initial state I by playing G.

Basic Definitions

Definition

(Subgame). A game $G = (F, A, I, R)$ is a **subgame** of $G' = (F', A', I', R')$ iff $F \subseteq F'$, $A \subseteq A'$, $I \subseteq I'$, $R \subseteq R'$, and F, A, I and R are not empty.

Definition

(Subgame Independence). Two subgames $Gs = (Fs, As, Is, Rs)$ and $Gs' = (Fs', As', Is', Rs')$ of game G are **independent** each other iff $Fs \cap Fs' = \oslash$ and $As \cap As' = \oslash$.

Subgame Detection

- Dependency relations between fluents and actions
  - Precondition
    - e.g. if action M is legal then fluent F must be true
  - Positive Effect
    - e.g. fluent F is true (not true in current state) in next state if a player takes move M
  - Negative Effect
    - e.g. fluent F is not true (true in current state) in next state if a player takes move M

Subgame Detection

- Dependency relations between fluents and actions
  - Precondition
    - e.g. if action M is legal then fluent F must be true
  - Positive Effect
    - e.g. fluent F is true (not true in current state) in next state if a player takes move M
  - Negative Effect
    - e.g. fluent F is not true (true in current state) in next state if a player takes move M
- Independent subgames
  - connected components of fluents and actions with dependency relations

Fluent and Action Instantiation

- Why
  - subgames share fluent and action names
- When
  - if the value of one argument of a fluent does not change in the whole game

Fluent and Action Instantiation

- Why
  - subgames share fluent and action names
- When
  - if the value of one argument of a fluent does not change in the whole game

Fluent and Action Instantiation

- Why
  - subgames share fluent and action names
- When
  - if the value of one argument of a fluent does not change in the whole game

Example *(Nim with two heaps of size 1 and 2)*:
({heap(X,N)},{reduce(X,M)},{heap(a,1),heap(b,2)},{player1,player2})
fluent/action instantiation ⇒
({heap(a,N),heap(b,N)},{reduce(a,M),reduce(b,M)},...)
subgame detection ⇒
({heap(a,N)},{reduce(a,M)},{heap(a,1)},{player1,player2}),
({heap(b,N)},{reduce(b,M)},{heap(b,2)},{player1,player2})

Outline

GDL Game Rule Analysis

An impartial game is an alternating move game and the legal moves of the game only depend on the position (or state) of the game.

- Legal rules
  - if, given any state of the game, the legal rules give the same moves for every player if he has control in that state
- Next rules
  - if, given any state of the game and a move, the next rules give the same next state for every player if he does this move

GDL Game Rule Analysis

- Legal rules
  - if, given any state of the game, the legal rules give the same moves for every player if he has control in that state
- Next rules
  - if, given any state of the game and a move, the next rules give the same next state for every player if he does this move

Counter example (TicTacToe):

```
(<= (legal ?w (mark ?x ?y))
    (true (cell ?x ?y b))
    (true (control ?w)))
```

```
(<= (next (cell ?m ?n x))
    (does xplayer (mark ?m ?n))
    (true (cell ?m ?n b)))
(<= (next (cell ?m ?n o))
    (does oplayer (mark ?m ?n))
    (true (cell ?m ?n b)))
```

Game Nim and Nimber

Theorem

(Sprague Grundy theorem). Every impartial game under the normal play convention is equivalent to a nimber.

Definition

A **nimber** is a special game denoted by *n for some integer n and n $\geq$ 0. We define *0 = {}, and *(n+1) = *n $\cup$ {*n}.
Given two nimbers G and H, nim addition

$$G \oplus H = \{G \oplus h | h \in H\} \cup \{g \oplus H | g \in G\}.$$

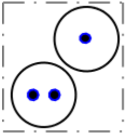
Game Nim and Nimber

Definition

A **nimber** is a special game denoted by *n for some integer n and n $\geq$ 0. We define *0 = {}, and *(n+1) = *n $\cup$ {*n}.
Given two nimbers G and H, nim addition

$$G \oplus H = \{G \oplus h | h \in H\} \cup \{g \oplus H | g \in G\}.$$

Nim example:



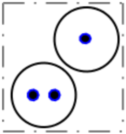
$$*1 = *0 \cup \{*0\} = \{\} \cup \{\{\}\} = \{*0\},$$
$$*2 = *1 \cup \{*1\} = \{\{\}\} \cup \{\{\{\}\}\} = \{*0, *1\}$$

Game Nim and Nimber

Definition

A **nimber** is a special game denoted by *n for some integer n and $n \geq 0$. We define $*0 = \{\}$, and $*(n{+}1) = *n \cup \{*n\}$.
Given two nimbers G and H, nim addition

$$G \oplus H = \{G \oplus h | h \in H\} \cup \{g \oplus H | g \in G\}.$$



$*1 = *0 \cup \{*0\} = \{\} \cup \{\{\}\} = \{*0\}$,
$*2 = *1 \cup \{*1\} = \{\{\}\} \cup \{\{\{\}\}\} = \{*0, *1\}$

Winning Conditions of Nim

1. the player to make the last move wins (normal play game)
2. the player to make the last move loses (misère game)

Theorem

A game is impartial iff its all subgames are impartial.

Decomposition Search

- Subgame search
  - calculate the nimber of each subgame
- Global game search
  - use nim-addition to find optimal strategies

Experimental Results

For Nim with 4 heaps:
time cost(second) for finding the first optimal strategy

Time Cost(*s*)	Normal Play			Misère
	Heaps Size			
	1,5,4,2	2,2,10,10	11,12,15,25	12,12,20,20
Normal Search	0.4	3.5	6607	10797
Decomposition Search	0.01	0.01	0.07	0.06

Outline

Local Concepts

Definition

A **local goal (resp. terminal) concept** (local concept for short) is a ground predicate call that occurs in the body of the goal (resp. terminal) predicate's definition.

Local Concepts

Definition

A **local goal (resp. terminal) concept** (local concept for short) is a ground predicate call that occurs in the body of the goal (resp. terminal) predicate's definition.

Example

```
(<= (goal xplayer 100) (line1 x) (line2 x))
```

there are two local goal concepts, *(line1 x)* and *(line2 x)*.

Turn-Move Sequences

Definition

A **turn-move sequence** is a tuple $Seq = (Ts, Ms, Es)$ where

- Ts is a list of player names, indicated by $T_1 \circ T_2 \circ \ldots \circ T_n$,
- Ms is a list of moves, indicated by $M_1 \circ M_2 \circ \ldots \circ M_n$,
- Es is a set of evaluations of local concepts, where $n \geq 0$.

Definition

Turn-move sequence $Seq_1 = (Ts_1, Ms_1, Es_1)$ is **evaluation dominated** by turn-move sequence $Seq_2 = (Ts_2, Ms_2, Es_2)$ **under** a set of local concepts Cs iff

- $Ts_1 = Ts_2$,
- $\forall_{C \in Cs}(Seq_1 \models C \Rightarrow Seq_2 \models C)$.

Decomposition Search II

Subgame Search

For each subgame state,

- expand all legal moves of all players
- return all simplified turn-move sequences

Global Game Search

Using normal search methods, for each global game state,

- choose legal moves from turn-move sequences returned from subgame search

Experimental Results

Subgame Search Results

One Subgame of Double-Tictactoe					
Search Depth	1	2	3	4	5
All Sequences	18	288	4032	47328	483840
Simplified Seqs	2	4	10	26	64
Time Cost(s)	0.17	2	10	28	55

One Subgame of Double-Tictactoe				
Search Depth	6	7	8	9
All Sequences	3870720	23224320	92897280	185794560
Simplified Seqs	148	324	674	912
Time Cost(s)	127	469	678	790

Experimental Results

Global Game Search Results

Time Cost(s)	**Search Depth**				
	1*2	2*2	3*2	4*2	5*2
Decomposition Search	0.36	4.36	24	80	179
Normal Search	< 1800			$> 3600 * 4$	

Time Cost(s)	**Search Depth**			
	6*2	7*2	8*2	9*2
Decomposition Search	301	1022	1530	1793
Normal Search	$> 3600 * 4$			

Outline

Additional Work for Subgame Detection

Properties of Parallel Games:

- At least two independent subgames
- A player has to move in all subgames on his turn
- All subgames share move names (called compound moves)

Additional Work for Subgame Detection

Properties of Parallel Games:

- At least two independent subgames
- A player has to move in all subgames on his turn
- All subgames share move names (called compound moves)

Example

Parallel-TicTacToe has two tictactoe subgames, these two subgames share one move name *mark*, e.g. (mark ?x1 ?y1 ?x2 ?y2)

Additional Work for Subgame Detection

Properties of Parallel Games:

- At least two independent subgames
- A player has to move in all subgames on his turn
- All subgames share move names (called compound moves)

Example

Parallel-TicTacToe has two tictactoe subgames, these two subgames share one move name *mark*, e.g. (mark ?x1 ?y1 ?x2 ?y2)

We have to find and split compound moves

Additional Work for Subgame Detection

Properties of Parallel Games:

- At least two independent subgames
- A player has to move in all subgames on his turn
- All subgames share move names (called compound moves)

Example

Parallel-TicTacToe has two tictactoe subgames, these two subgames share one move name *mark*, e.g. (mark ?x1 ?y1 ?x2 ?y2)

We have to find and split compound moves

- by analyzing next and legal rules

Compound Move Detection Example

```
Next Rules:
(1).(<= (next (cell1 ?x1 ?y1 x))
        (does xplayer (mark ?x1 ?y1 ?x2 ?y2)))
(2).(<= (next (cell1 ?x1 ?y1 o))
        (does oplayer (mark ?x1 ?y1 ?x2 ?y2)))
(3).(<= (next (cell1 ?x ?y ?mark))
        (true (cell1 ?x ?y ?mark))
        (does xplayer (mark ?x1 ?y1 ?x2 ?y2))
        (distinctcell ?x ?y ?x1 ?y1))
(4).(<= (next (cell1 ?x ?y ?mark))
        (true (cell1 ?x ?y ?mark))
        (does oplayer (mark ?x1 ?y1 ?x2 ?y2))
        (distinctcell ?x ?y ?x1 ?y1))

Legal Rules:
(<= (legal ?player (mark ?x1 ?y1 ?x2 ?y2))
    (true (control ?player))
    (true (cell1 ?x1 ?y1 b))
    (true (cell2 ?x2 ?y2 b)))
```

Decomposition Search

- Subgame Search
  - normal alternating move or simultaneous move game search methods
- Global Game Search
  - make sure subgame search gets equal length plans in all subgames

Decomposition Search

- Subgame Search
  - normal alternating move or simultaneous move game search methods
- Global Game Search
  - make sure subgame search gets equal length plans in all subgames

Decomposition Search

- Subgame Search
  - normal alternating move or simultaneous move game search methods
- Global Game Search
  - make sure subgame search gets equal length plans in all subgames

Additional Work for Subgame Detection

Properties of Serial Games

- At least two independent subgames
- All subgames are ordered and played one after another
- Only one subgame is played in each turn

Additional Work for Subgame Detection

Properties of Serial Games

- At least two independent subgames
- All subgames are ordered and played one after another
- Only one subgame is played in each turn

We have to find the order between subgames

Additional Work for Subgame Detection

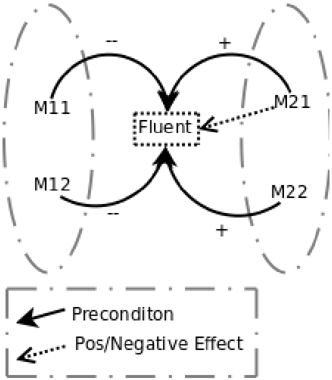
Properties of Serial Games

- At least two independent subgames
- All subgames are ordered and played one after another
- Only one subgame is played in each turn

We have to find the order between subgames

- by analyzing dependency relations between actions and fluents

Subgame Order Detection



Decomposition Search

- Subgame Search
  - normal alternating move or simultaneous move game search methods
- Global Game Search
  - control subgame search in terms of the order

Decomposition Search

- Subgame Search
  - normal alternating move or simultaneous move game search methods
- Global Game Search
  - control subgame search in terms of the order

Decomposition Search

- Subgame Search
  - normal alternating move or simultaneous move game search methods
- Global Game Search
  - control subgame search in terms of the order

Outline

Done and ToDo

What We Have Done:

1. subgame detection algorithm and
2. decomposition search algorithms for different classes of games

Done and ToDo

What We Have Done:

1. subgame detection algorithm and
2. decomposition search algorithms for different classes of games

What Can be Improved:

1. apply pruning techniques in decomposition search of partial games, e.g. alpha-beta pruning
2. use local concept evaluations more efficiently in global game search

Done and ToDo

Thank You!!

For Further Reading I

- John H. Conway
  On Numbers and Games.
  Academic Press, 1976.
- Elwyn R. Berlekamp, John H. Conway, Richard K. Guy
  Winning Ways 2nd Edition.
  2001.
- Martin Müller
  Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames
  1999.

For Further Reading II

M. Günther, S. Schiffel and M. Thielscher
Decomposition of Single Player Games
2007.

Eric Schkufza
Decomposition of Games for Efficient Reasoning
2008.

Time Complexity Comparison I

Impartial and Partial Games

Assume that a game G has n subgames, G_1, G_2, ..., G_n with V_1, V_2, ..., V_n states respectively,

- normal search: $O(V_1 * V_2 * ... * V_n)$
- decomposition search: $O(V_1 + V_2 + ... + V_n)$

Example

For double-tictactoe, the number of states is about 18!(including revisited states), while the state for each subgame is about $\prod_{n=1}^{9}(2n)$ which is $\prod_{n=1}^{9}(2n-1)$ times smaller than 18!

Time Complexity Comparison II

Parallel Games

Assume that a parallel game G has n subgames, G_1, G_2, ..., G_n with V_1, V_2, ..., V_n states respectively,

- normal search: $O(V_1 * V_2 * ... * V_n)$
- decomposition search: $O(V_1 + V_2 + ... + V_n)$

Serial Games

Assume that a serial game G has n subgames, for subgame i there are V_i states and T_i terminal states

- normal search: $O(V_1 + T_1 * V_2 + T_1 * T_2 * V_3 + ... + T_1 * T_2 * ... * T_{n-1} * V_n)$
- decomposition search: $O(V_1 + V_2 + ... + V_n)$