

Games in LTL Fragments

Salvatore La Torre

Dipartimento di Informatica ed Applicazioni
Università degli Studi di Salerno

Linear-time Temporal Logic (LTL)

□ Correctness requirements for reactive systems

“Every request is eventually granted”

$$\square (r \rightarrow \diamond g)$$

□ Most studied decision problem:

◆ model checking (closed systems)

Is M a model of φ ?

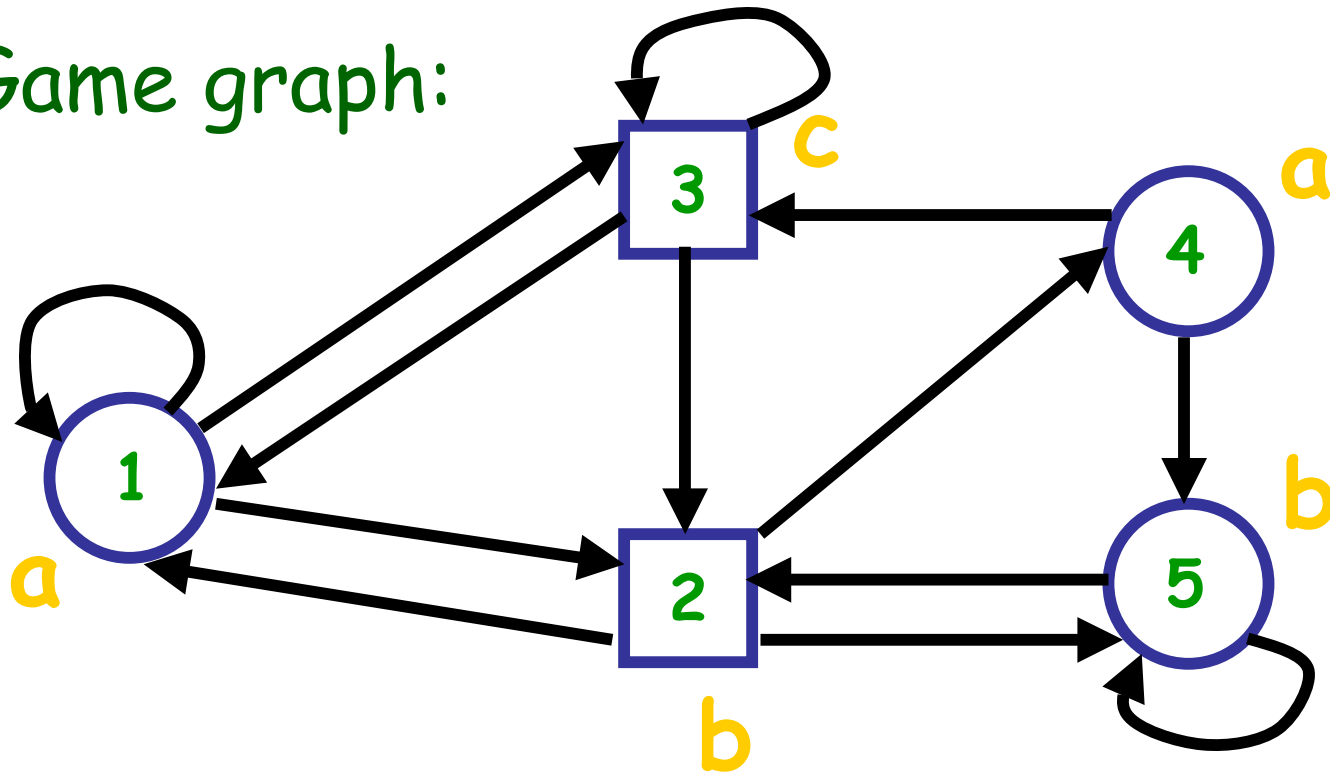
LTL specs in open systems

The system is a module interacting with the other modules (environment)

- ❑ Controller synthesis
- ❑ Realizability of specifications
- ❑ Verification of open systems
- ❑ Modular verification (module-checking)

LTL Games

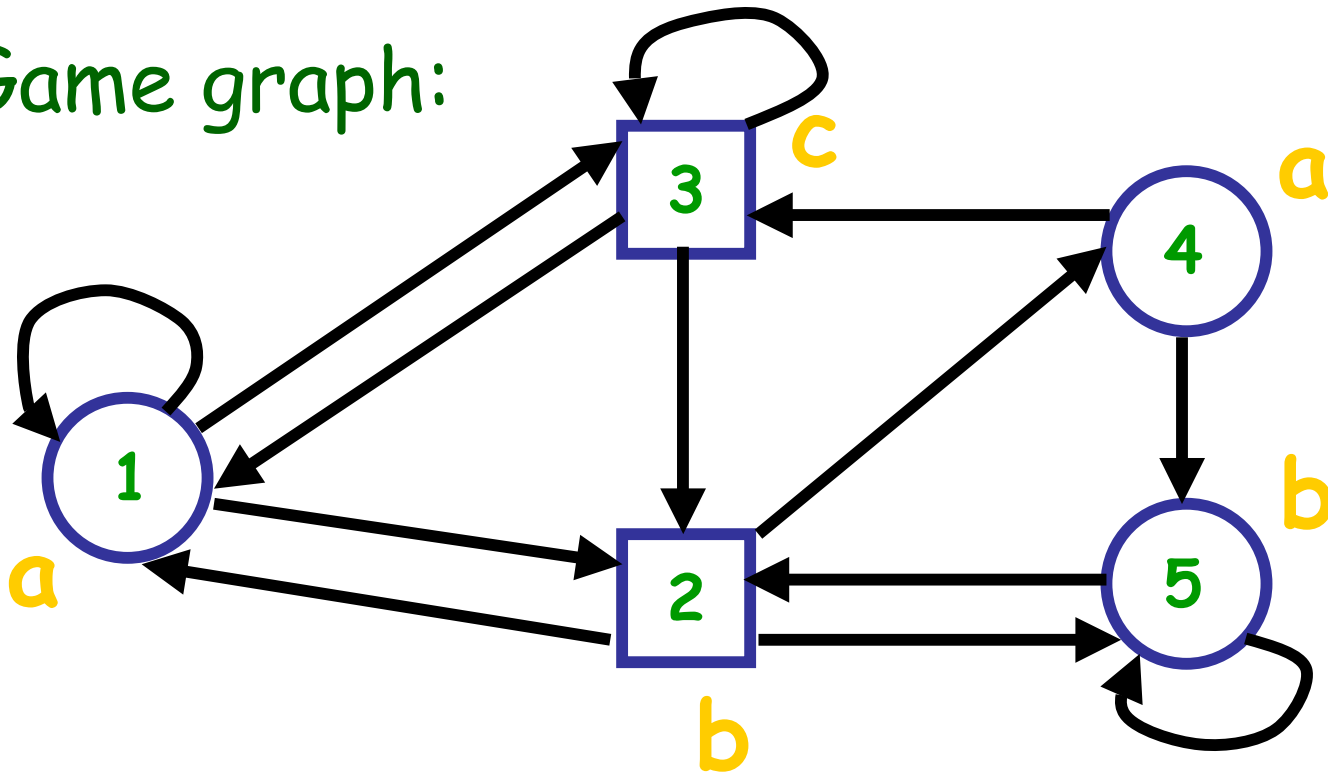
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Play

➤ Game graph:

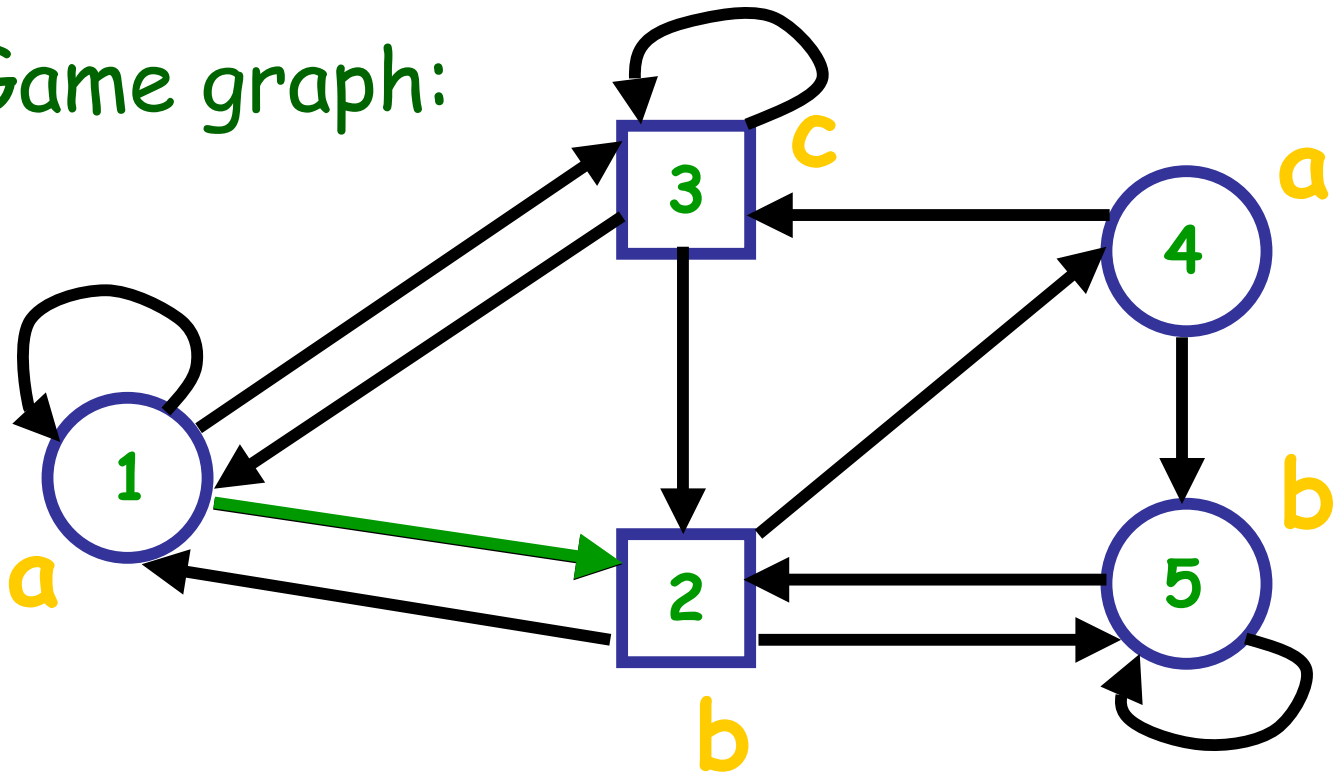


➤ Specification:

$\square (a \rightarrow \diamond b)$

Play

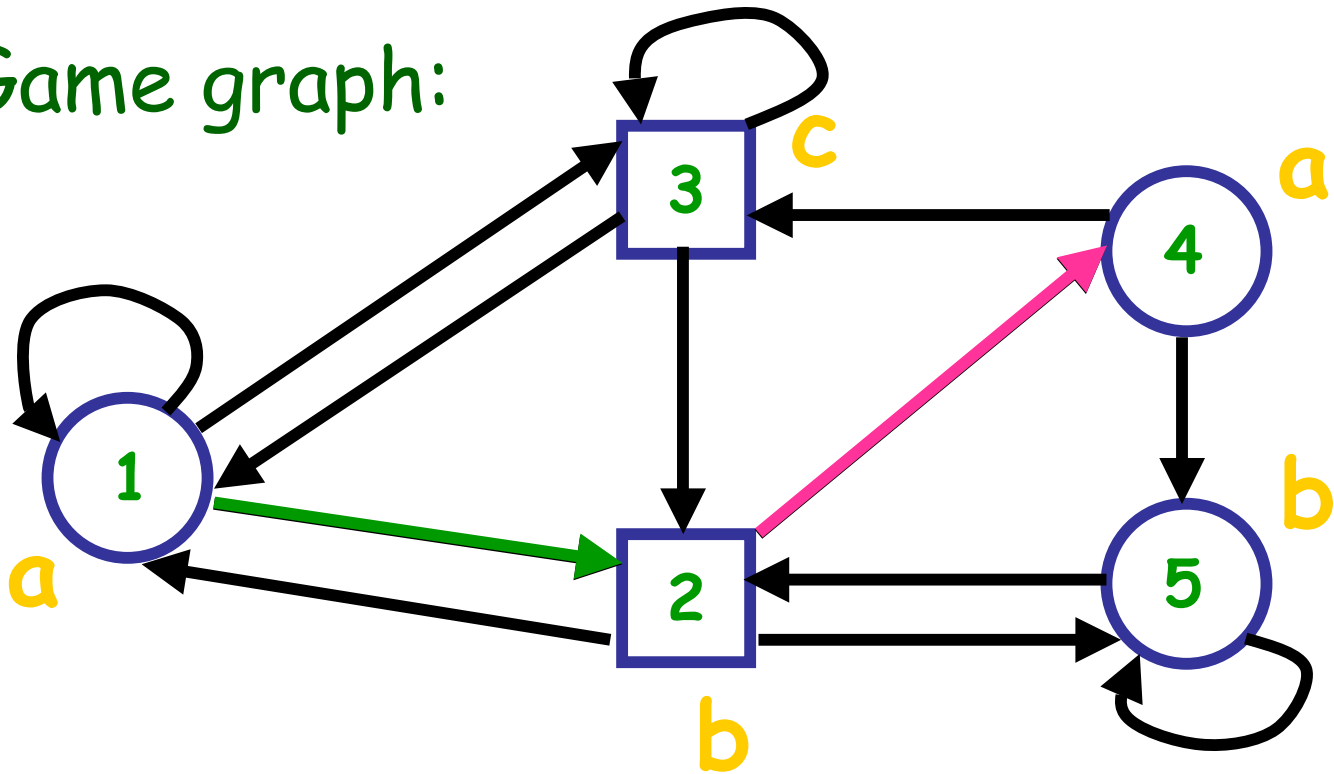
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Play

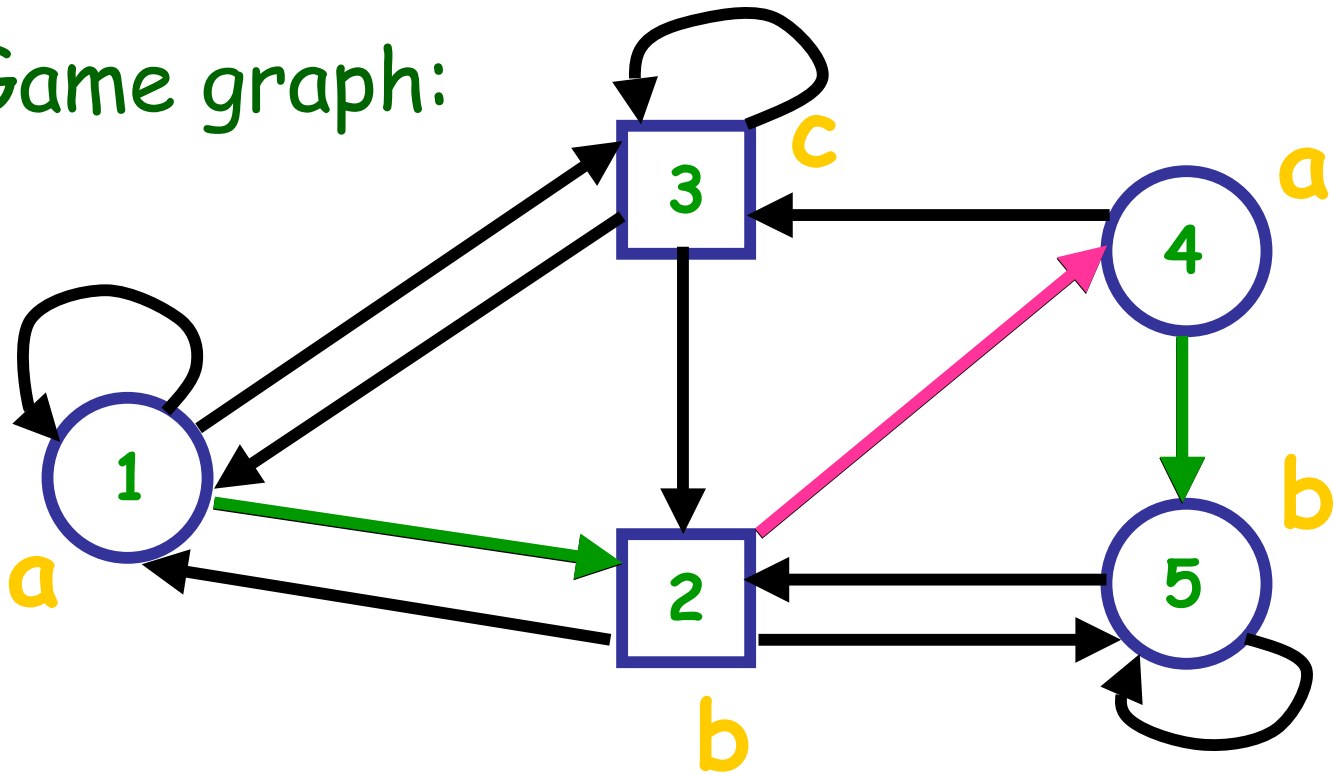
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Play

➤ Game graph:

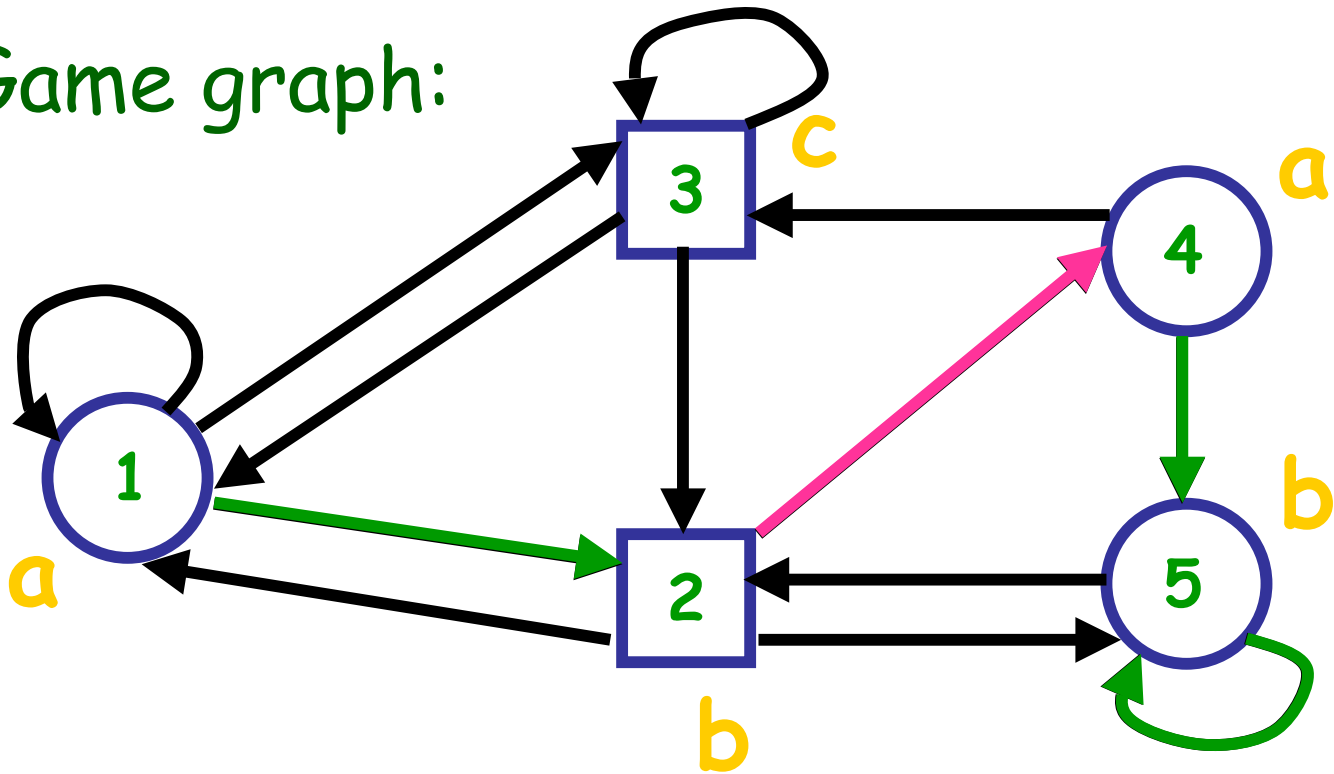


➤ Specification:

$\square (a \rightarrow \diamond b)$

Play

➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

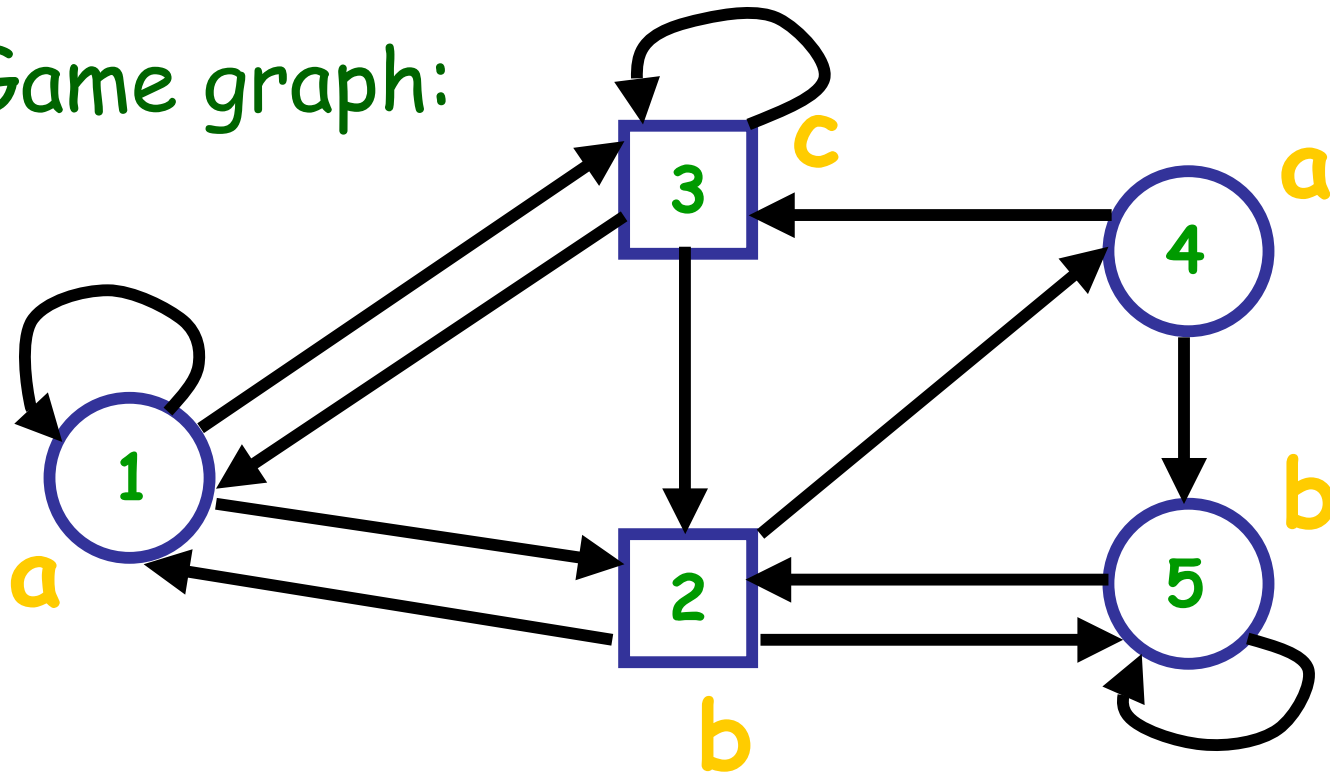
Decision Problem

- Strategy: function
from play ending at a system state s
to a successor of s
- Strategy is winning :
All plays constructed according to it
satisfy specification

Is there a winning strategy of the
protagonist?

Example: Strategy

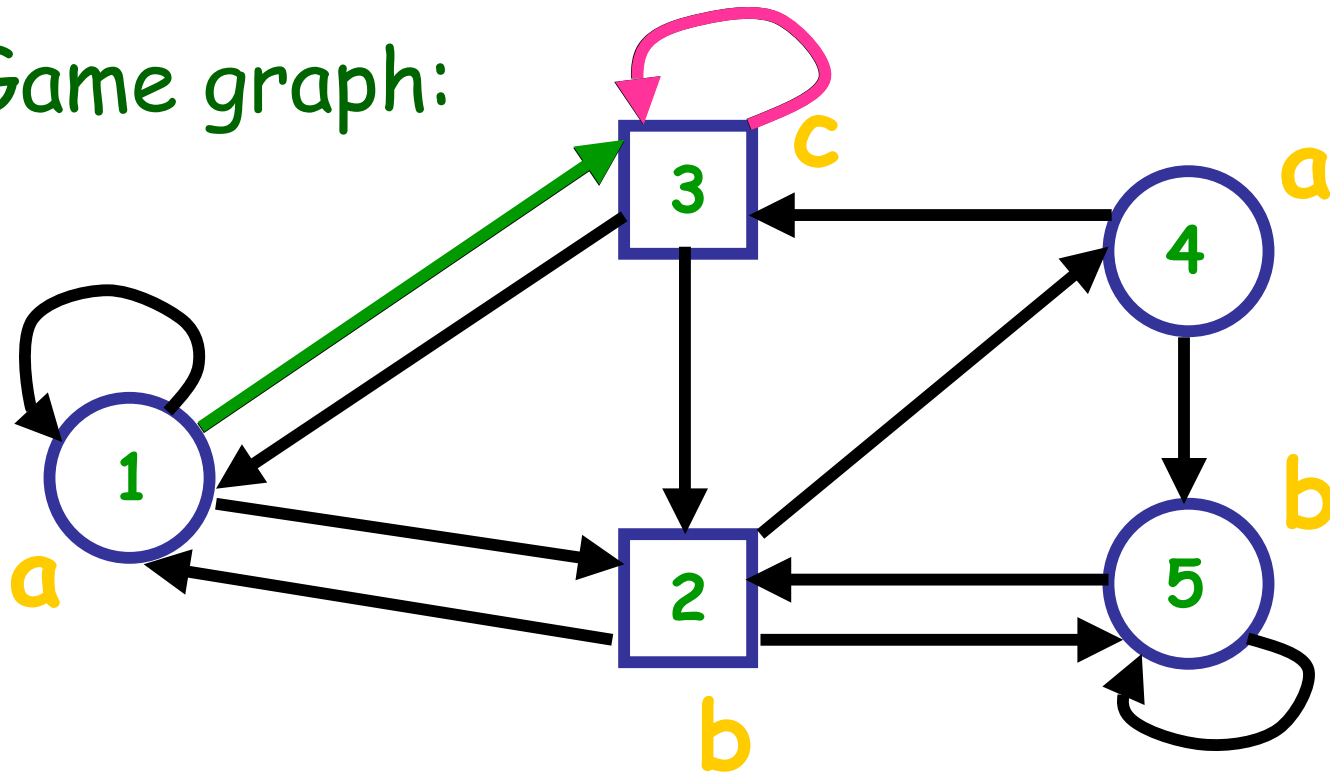
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

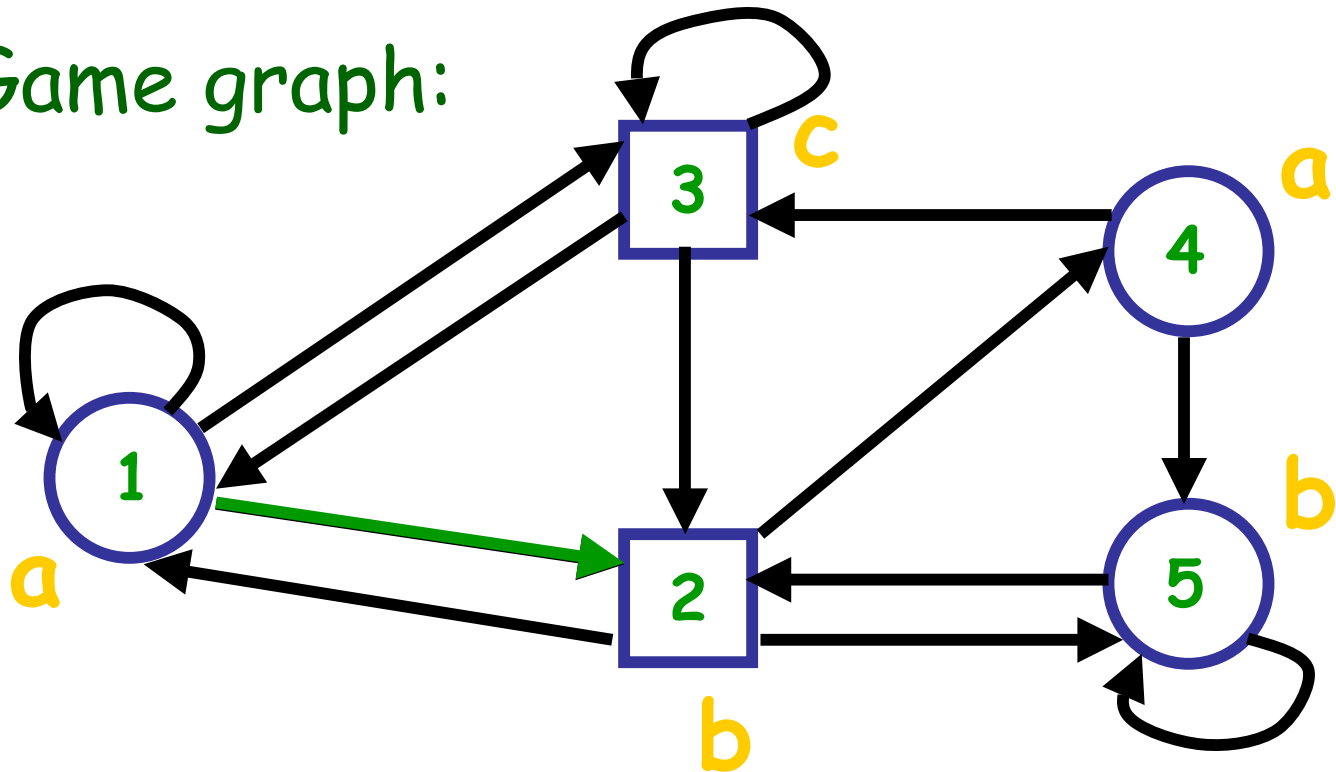
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

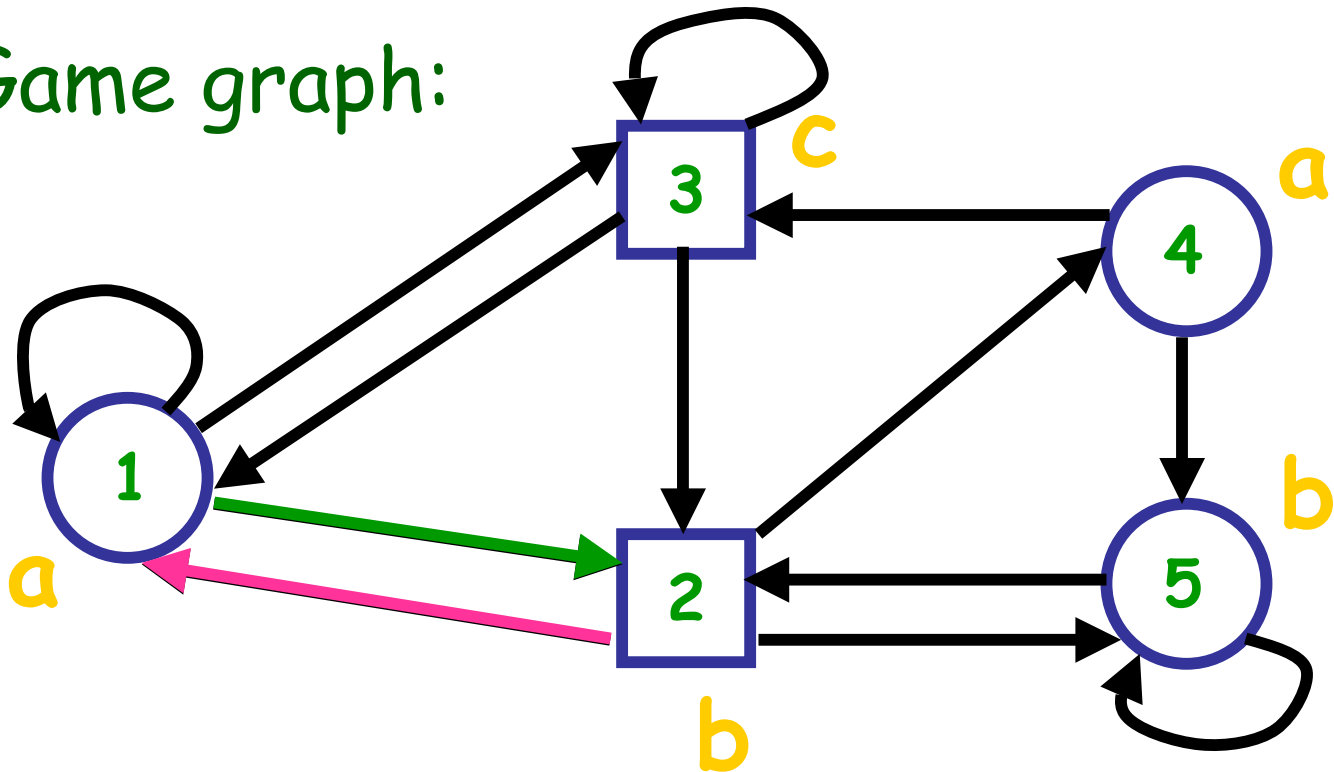
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

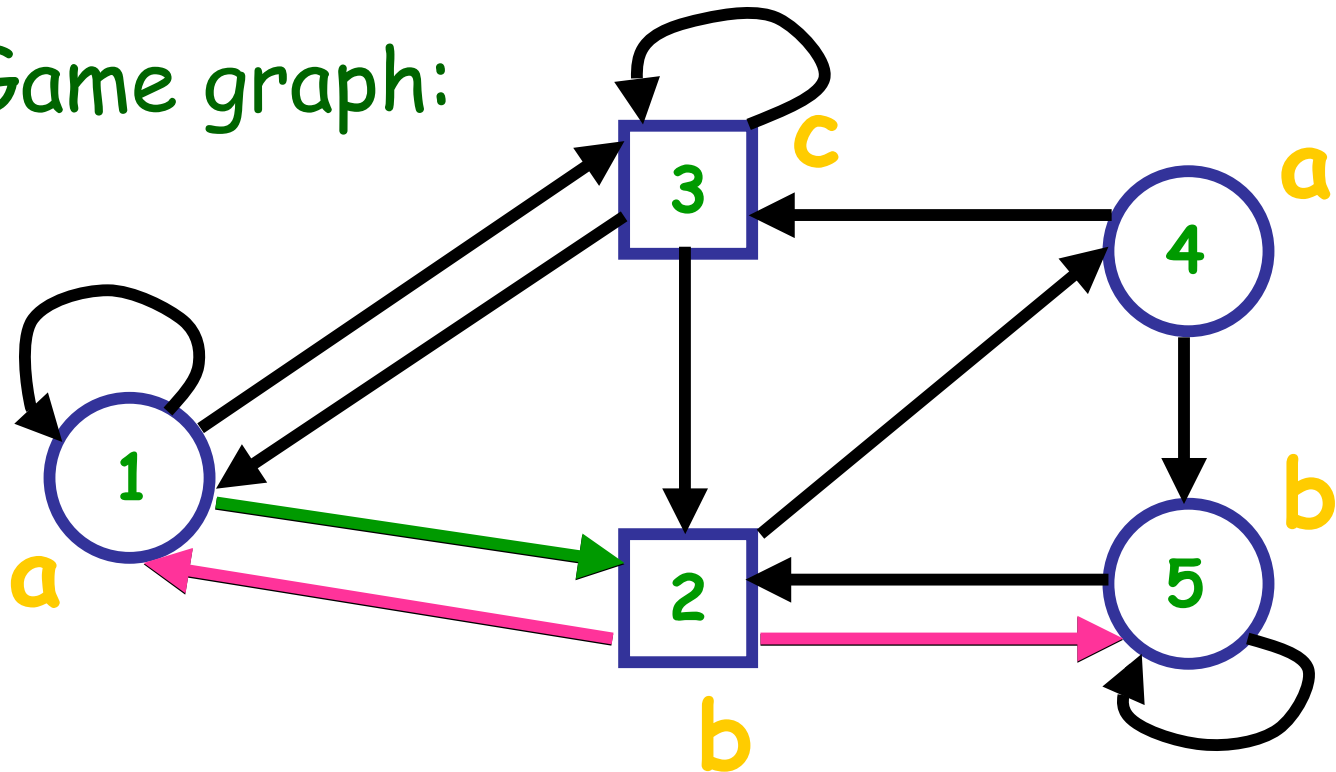
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

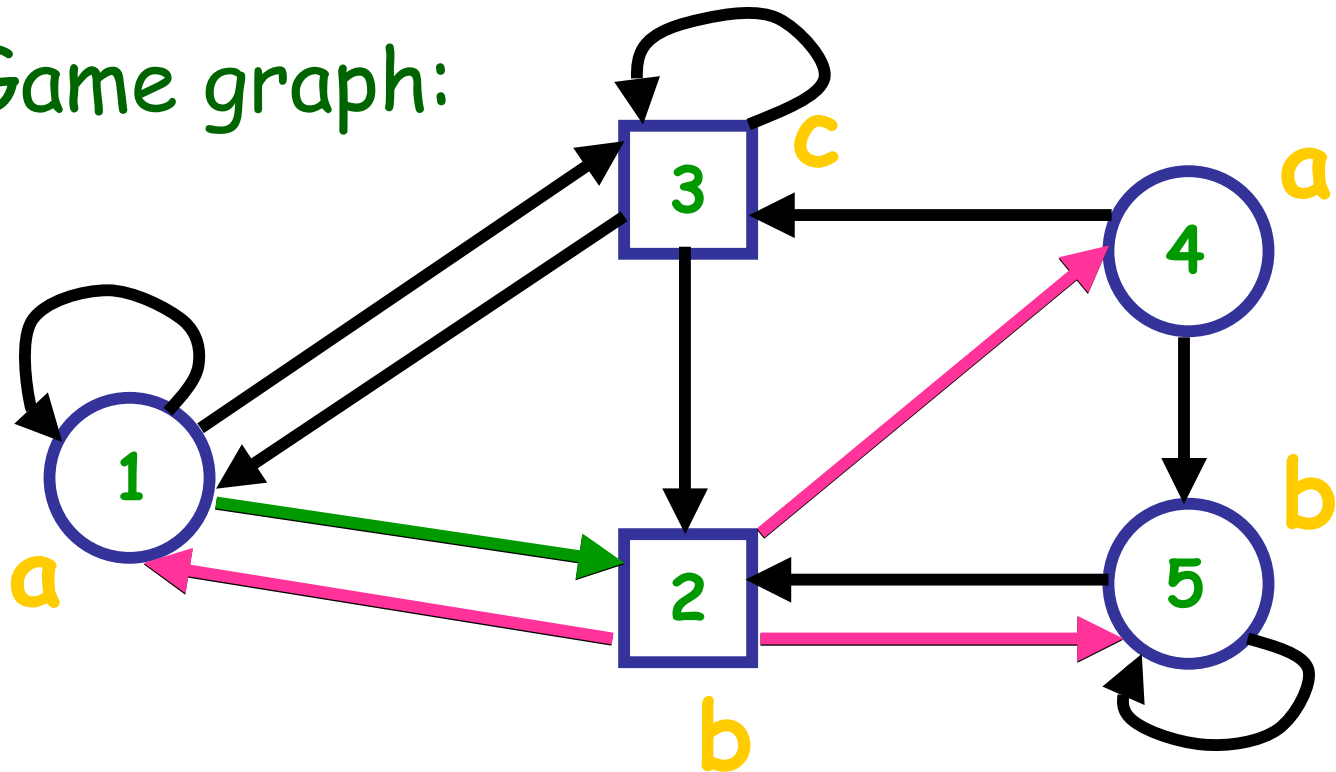
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

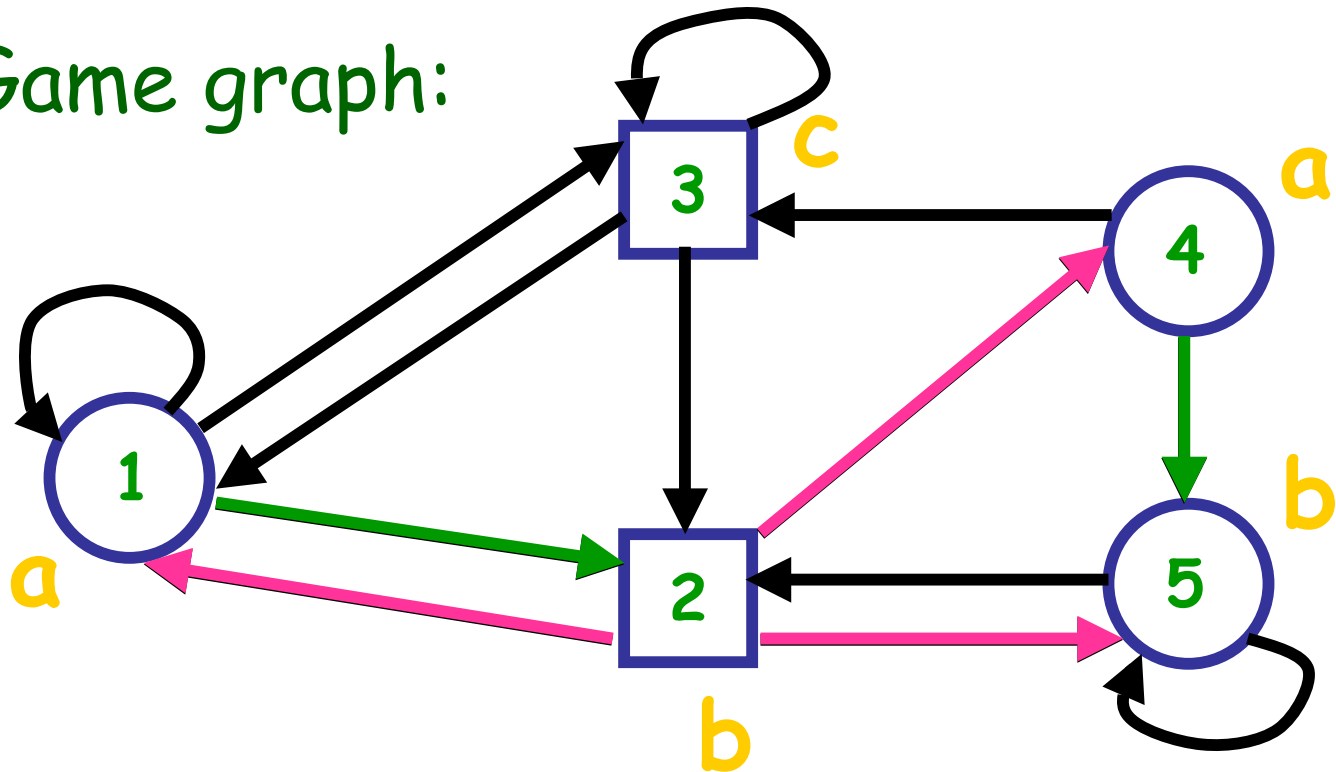
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

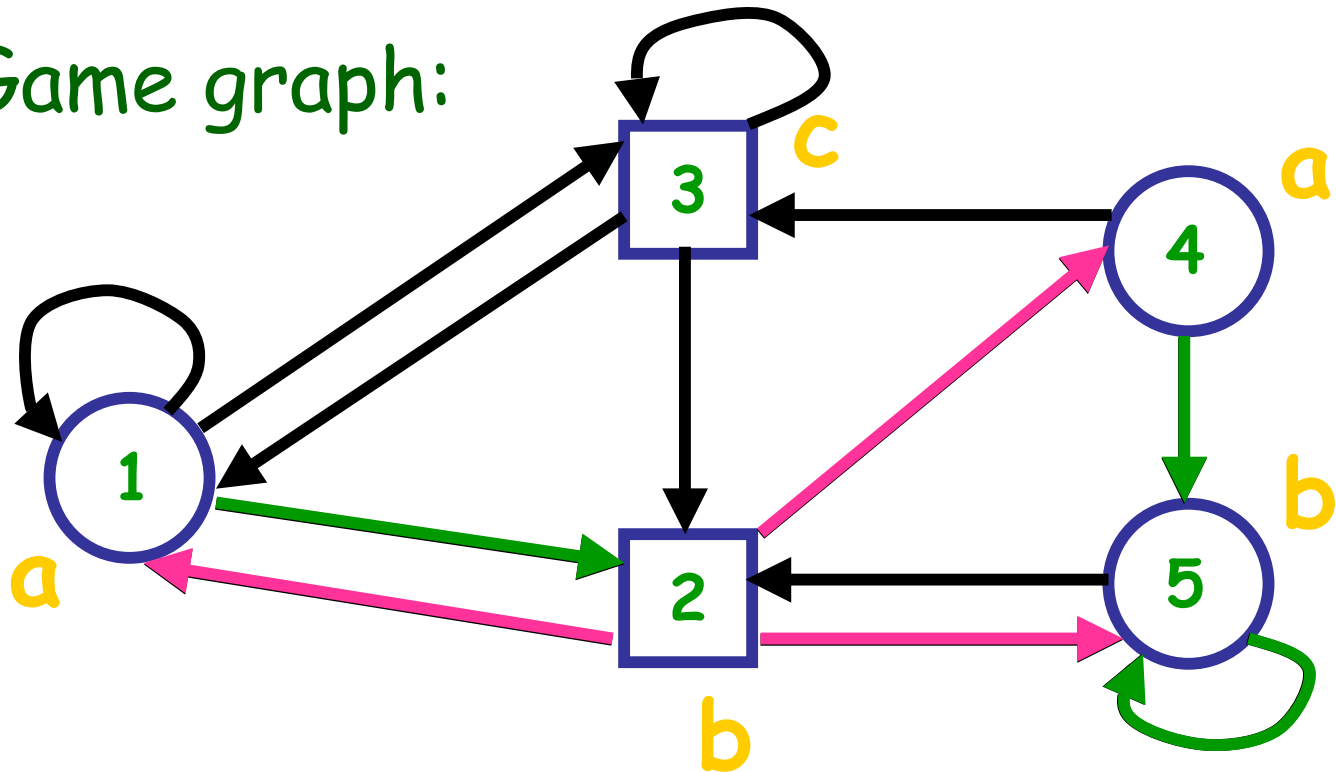
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Example: Strategy

➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Computational Complexity of LTL Games

- Deciding LTL games is 2Exptime-complete
[Pnueli-Rosner POPL'89]
- Complexity of games in LTL fragments:
 - ◆ Deterministic generators and games for LTL fragments [Alur - La Torre LICS'01]
 - ◆ Games for positive LTL fragments [Marcinkowski - Truderung CSL'02]
 - ◆ Games in fragments without "next" and "until"
[Alur - La Torre - Madhusudan CONCUR'03]

Other References

Realizability

[Abadi-Lamport-Wolper ICALP'89]

Module checking

[Kupferman-Vardi CAV'96 & '97]

Alternating Temporal Logic

[Alur-Henzinger-Kupferman JACM'02]

Talk Outline

✓ Overview

⇒ Notation and general solution to LTL games

□ Upper bounds: deterministic generators

□ Lower bounds

□ Encoding TMs without “next” and “until”

□ Expspace-hardness of $B(L_{\diamond, \wedge, \vee}(\Pi))$

□ 2Exptime-hardness of $L_{\square, \diamond, \wedge, \vee}(\Pi)$

□ Conclusions

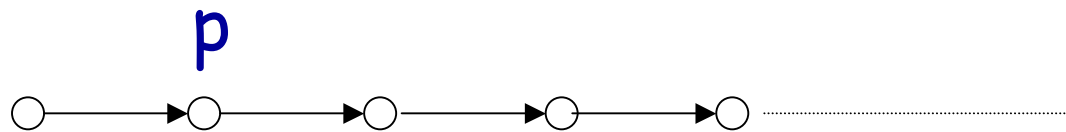
LTL

□ Syntax

$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \circ\varphi \mid \square\varphi \mid \diamond\varphi \mid \varphi \cup \varphi$

□ Semantics

- $\circ p$:

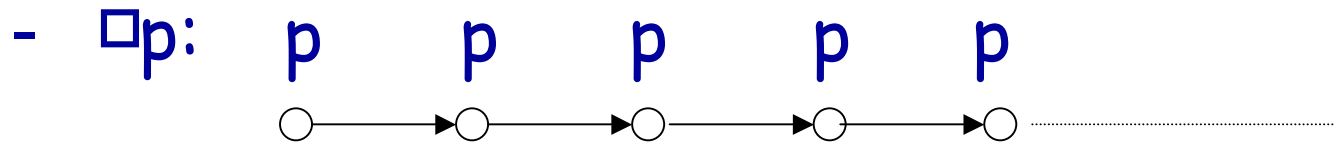


LTL

□ Syntax

$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \circ\varphi \mid \square\varphi \mid \diamond\varphi \mid \varphi \cup \varphi$

□ Semantics

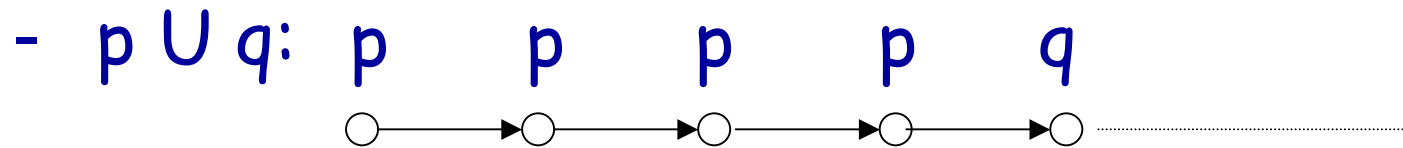


LTL

□ Syntax

$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \circ\varphi \mid \square\varphi \mid \diamond\varphi \mid \varphi \cup \varphi$

□ Semantics



Some Notation

□ $B(\Gamma)$ denotes:
◆ Boolean combinations of formulas from Γ

□ $L_{op_1, \dots, op_k}(\Gamma)$ denotes:
◆ formulas from Γ using only operators in the list op_1, \dots, op_k

□ For example:

$L_{\diamond, \wedge, \vee}(\Pi)$ denotes the LTL fragment

$$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \diamond \varphi, \quad p \in \Pi$$

More LTL fragments

□ $L_{\square, \diamond, \wedge, \vee}(\Pi)$: (usually $LTL(\square, \diamond)$)
 $\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \diamond \varphi \mid \square \varphi$

□ $B(L_{\diamond, \wedge}(\Pi))$: bool. combinations of
 $\varphi := p \mid \varphi \wedge \varphi \mid \diamond \varphi$

□ $B(L_{\diamond, \circ, \wedge}(\Pi))$: bool. combinations of
 $\varphi := p \mid \varphi \wedge \varphi \mid \diamond \varphi \mid \circ \varphi$

□ $B(L_{\diamond, \circ, \wedge, \vee}(\Pi))$: bool. combinations of
 $\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \diamond \varphi \mid \circ \varphi$

LTL Games

- Winning condition is an LTL formula
- Deciding LTL games is 2Exptime-complete
[Pnueli-Rosner'89]
- ◆ Construct Buchi generator (size $n=2^{O(\phi)}$)
[Vardi-Wolper'94]
- ◆ Determinize it = Rabin automaton with $2^{O(n)}$ states
and n pairs [Safra '88]
- ◆ Emptiness of Rabin tree automata with n states
and m pairs: $O((n \cdot m)^{c \cdot m})$ [Pnueli-Rosner'89]

Buchi Games

□ Winning condition:

- ◆ Some accepting state must repeat infinitely often

□ Decision algorithm:

- ◆ $O(d \log m)$ space

(d =longest simple distance,
 m =number of states)

Talk Outline

- ✓ Overview
- ✓ Notation and general solution to LTL games
- ⇒ Upper bounds: deterministic generators
- Lower bounds
 - Encoding TMs without “next” and “until”
 - Expspace-hardness of $B(L_{\diamond, \wedge, \vee}(\Pi))$
 - 2Exptime-hardness of $L_{\square, \diamond, \wedge, \vee}(\Pi)$
- Conclusions

LTL Deterministic Generators

- LTL formulas may not have Buchi deterministic generators
- Standard approach:
 - ◆ Construct nondeterministic generator
 - ◆ Determinize it
- LTL formulas have deterministic generators of size and longest distance $\leq 2\text{Exp}$ (matching lower bounds [KV'98])

Generators for $B(L_{\diamond, \wedge}(\Pi))$

□ There exists DBA of Exp size and linear longest distance

□ Construction is optimal:

◆ Ex. $\diamond p_1 \wedge \dots \wedge \diamond p_n$

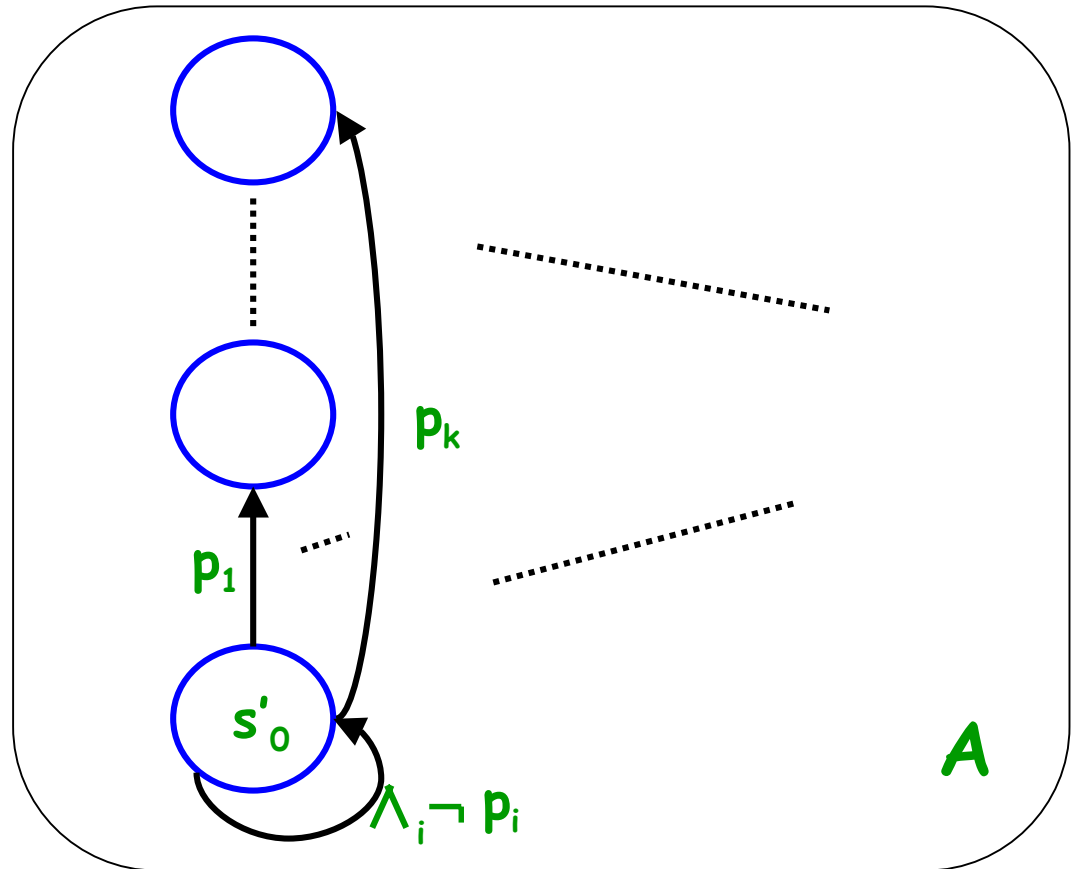
States store fulfilled predicates

Transition (non self-loop) required when new predicate is fulfilled

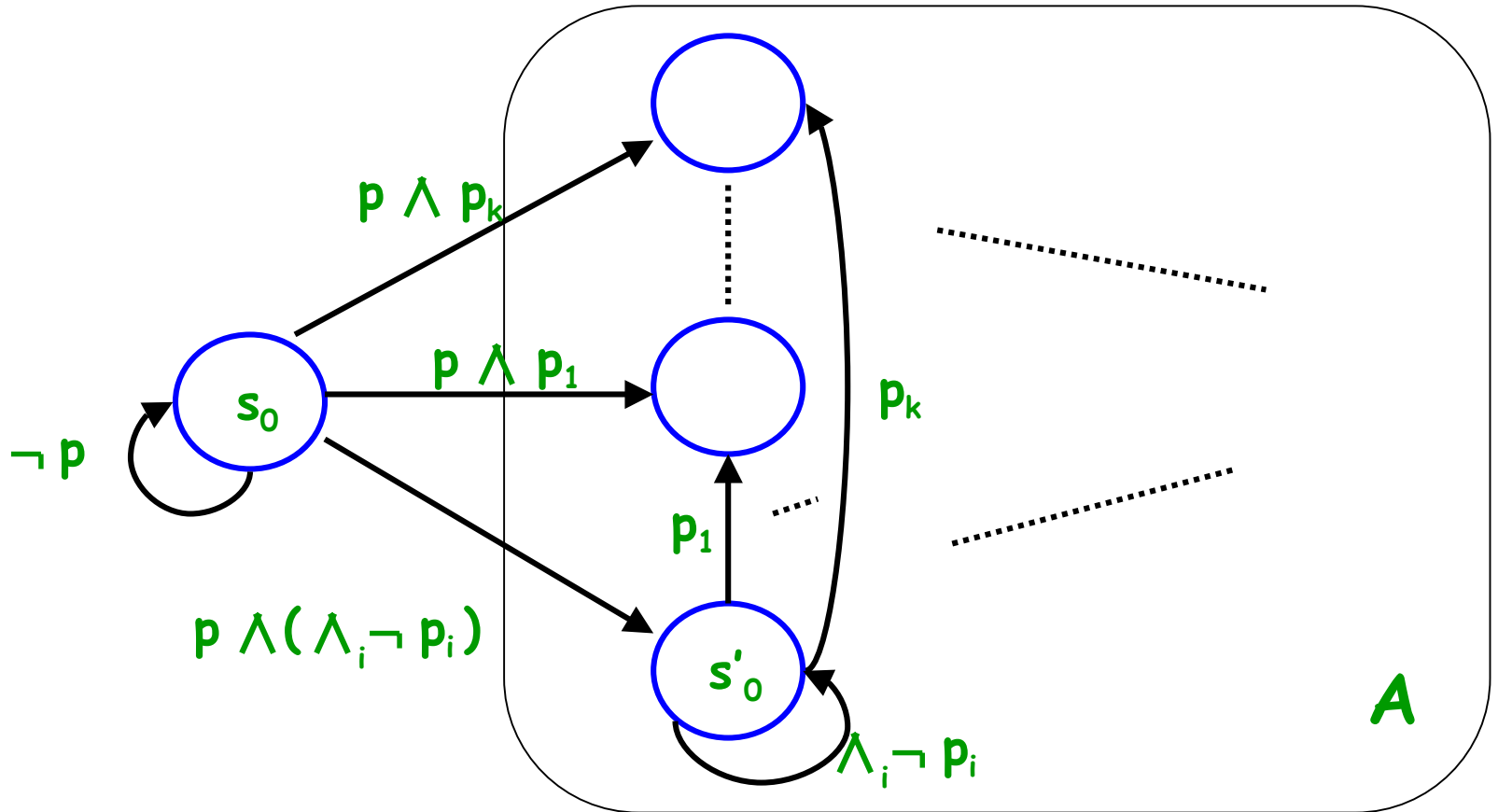
Partially-ordered Buchi Automata

- Transition graph is a DAG with self-loops
- Construction for intersection and union keeps linear longest distance (d_1+d_2)
- Complement is trivial
- Efficient construction for $\diamond(p \wedge \diamond \varphi)$ from POBD for $\diamond \varphi$

PODB Composition



PODB Composition



Generators for $B(L \diamond, \circ, \wedge(\Pi))$

- There exists DBA of Exp size and Exp longest distance
- Construction is optimal:
 - ◆ Ex. □ $(p \rightarrow \circ^n q)$
 - States store sequence of last n input
 - Exp-long path where the last n input are always different for each prefix

Automaton Construction

□ Push inside next operators ($O(n^2)$)

□ Interesting case:

$$\diamond (p \wedge O^k q \wedge \diamond \varphi)$$

◆ Use k copies of A' (det. gen. for $\diamond \varphi$)

◆ At h release copy started at $h-k$ if $p \wedge O^k q$ is not true at $h-k$, and release all the others, otherwise

(no more copies are started in this last case)

Generators for $B(L_{\diamond, \wedge, \vee}(\Pi))$ and $B(L_{\diamond, \circ, \wedge, \vee}(\Pi))$

□ There exists DBA of 2Exp size and Exp longest distance

□ Construction is optimal:

◆ Ex. $\diamond \bigwedge_{i=1}^n (p_i \vee \diamond q_i)$

States store sets of q 's :

if P then check if $\diamond q_i$ for $p_i \notin P$

Sequence of different sets of p 's

□ Push outside disjunctions

Generator for $L_{\square, \diamond, \wedge, \vee}(\Pi)$

□ $L_{\square, \diamond, \wedge, \vee}(\Pi)$ formulas may require det.
generator of size and longest distance

2Exp

◆ Ex. $\square (\diamond \bigwedge_{i=1}^n (a_i \vee \diamond b_i)) \longrightarrow \diamond \bigwedge_{i=1}^n (c_i \vee \diamond d_i)$

(States store for each set of b's
a list of sets of d's)

Generators Complexity

	Nondet. Gen.		Det. Generators	
	Size	L. Dist.	Size	L. Dist.
$B(L_{\diamond, \wedge}(\Pi))$	$\Theta(\text{Exp})$	$\Theta(\text{Linear})$	$\Theta(\text{Exp})$	$\Theta(\text{Linear})$
$B(L_{\diamond, \circ, \wedge}(\Pi))$	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$
$B(L_{\diamond, \wedge, \vee}(\Pi))$	$\Theta(\text{Exp})$	$\Theta(\text{Linear})$	$\Theta(2\text{Exp})$	$\Theta(\text{Exp})$
$B(L_{\diamond, \circ, \wedge, \vee}(\Pi))$	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$	$\Theta(2\text{Exp})$	$\Theta(\text{Exp})$
$L_{\square, \diamond, \wedge, \vee}(\Pi)$	$\Theta(\text{Exp})$	$\Theta(\text{Linear})$	$\Theta(2\text{Exp})$	$\Theta(2\text{Exp})$
LTL	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$	$\Theta(2\text{Exp})$	$\Theta(2\text{Exp})$

Solving LTL Games

- G = game graph, φ = LTL formula
 - ◆ Construct deterministic generator A of φ models
 - ◆ Solve the Buchi game $(G \times A, W)$
(W is the acceptance condition of A on $G \times A$)

- Complexity Buchi games: $O(d \log m)$ space
(d =longest simple distance, m =number of states)

Upper bounds

	Games	Det. Generators	
		Size	L. Dist.
$B(L_{\diamond, \wedge}(\Pi))$	PSPACE	$\Theta(\text{Exp})$	$\Theta(\text{Linear})$
$B(L_{\diamond, \circ, \wedge}(\Pi))$	EXPTIME	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$
$B(L_{\diamond, \wedge, \vee}(\Pi))$	EXPSPACE	$\Theta(2\text{Exp})$	$\Theta(\text{Exp})$
$B(L_{\diamond, \circ, \wedge, \vee}(\Pi))$	EXPSPACE	$\Theta(2\text{Exp})$	$\Theta(\text{Exp})$
$L_{\square, \diamond, \wedge, \vee}(\Pi)$	2EXPTIME	$\Theta(2\text{Exp})$	$\Theta(2\text{Exp})$
LTL	2EXPTIME	$\Theta(2\text{Exp})$	$\Theta(2\text{Exp})$

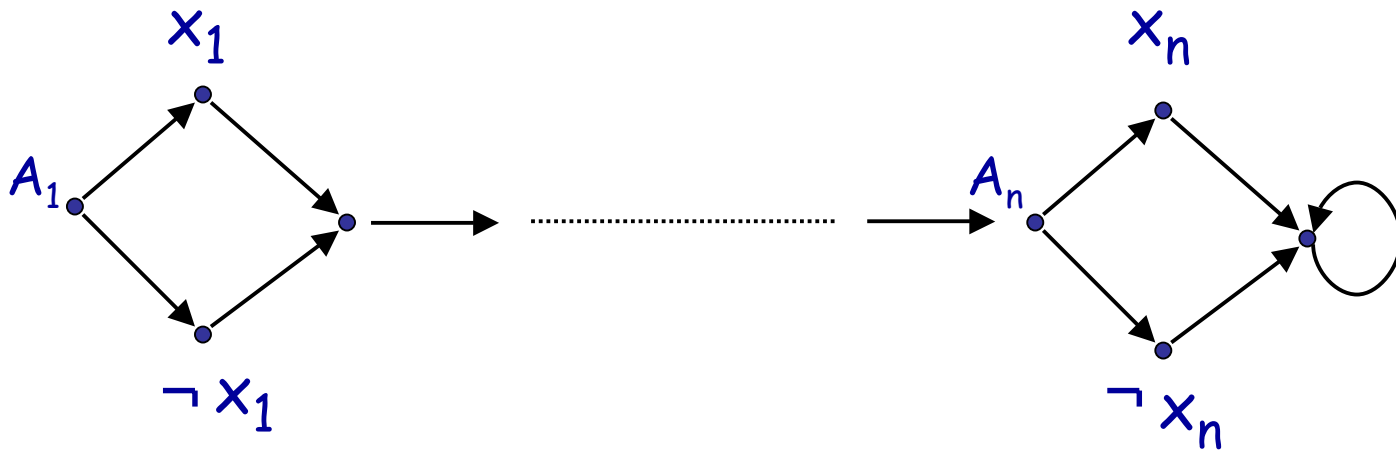
Talk Outline

- ✓ Overview
- ✓ Notation and general solution to LTL games
- ✓ Upper bounds: deterministic generators
- ⇒ Lower bounds
 - Encoding TMs without “next” and “until”
 - Expspace-hardness of $B(L_{\diamond, \wedge, \vee}(\Pi))$
 - 2Exptime-hardness of $L_{\square, \diamond, \wedge, \vee}(\Pi)$
- Conclusions

$B(L_{\diamond, \wedge}(\Pi))$: Pspace-hardness

QBF formula: $A_1 x_1 \dots A_n x_n \cdot \bigwedge_{i=1}^n c_i$

$B(L_{\diamond, \wedge}(\Pi))$ formula: $\bigwedge_{i=1}^n \diamond c_i$



$B(L \diamond, \circ, \wedge(\Pi))$: Exptime-hardness

□ Encoding from ALT-Pspace TM

□ System wins on plays either

encoding an accepting computation or
not encoding a computation

□ Encoding:

$$a_1 a_2 \dots a_{i-1} q a_i \dots a_n \longrightarrow a_1 a_2 \dots q' a_{i-1} a'_i \dots a_n$$

$$(a_1, 1) (a_2, 2) \dots (a_{i-1}, i-1) (a_i, i) \dots (a_n, n) (q, a_i, i) (q', a'_i, L)$$

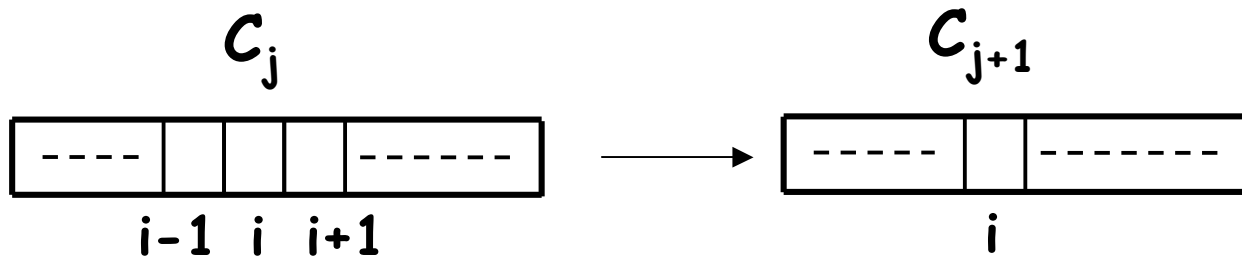
Talk Outline

- ✓ Overview
- ✓ Notation and general solution to LTL games
- ✓ Upper bounds: deterministic generators
- ⇒ Lower bounds
 - ⇒ Encoding TMs without “next” and “until”
 - Expspace-hardness of $B(L_{\diamond, \wedge, \vee}(\Pi))$
 - 2Exptime-hardness of $L_{\square, \diamond, \wedge, \vee}(\Pi)$
- Conclusions

Proving lower bounds

□ Encode acceptance problem for Turing Machines

□ Crucial point:



□ Problems:

◆ Zoom to a cell content

◆ Compare cells of consecutive configurations

With "until" and "next"

□ Zoom to cell $i = n(b_n \dots b_1)$:

◆ $b_n \dots b_1 a$ to encode "cell $b_n \dots b_1$ contains a "

◆ $\diamond O(b_n \wedge O(\dots \wedge O(b_1 \wedge O a) \dots))$ to check it

□ Compare across configurations:

◆ Modulo-2 counter to distinguish among consecutive configurations

◆ Constructs of type $O \cup (1 \wedge \varphi_1)$

New encoding of computations

\wedge \wedge \wedge

New encoding of computations

□◇ only checks for subsequences

Es. $\diamond(b_n \wedge \diamond(\dots \wedge \diamond(b_1 \wedge \diamond a) \dots))$,
("b_n...b₁ a" may not be consecutive)

□ $\langle a_0 \rangle_0 \langle a_1 \rangle_1 \dots \langle a_i \rangle_i \dots \langle a_{2n-1} \rangle_{2n-1}$ (proper sequence)

New encoding of computations

□◇ only checks for subsequences

Es. $\diamond(b_n \wedge \diamond(\dots \wedge \diamond(b_1 \wedge \diamond a) \dots))$,
("b_n...b₁ a" may not be consecutive)

□ $\langle a_0 \rangle_0 \langle a_1 \rangle_1 \dots \langle a_i \rangle_i \dots \langle a_{2n-1} \rangle_{2n-1}$ (proper sequence)

$p_n \dots p_1 a_i q_1 \dots q_n$

New encoding of computations

□◇ only checks for subsequences

Es. $\diamond(b_n \wedge \diamond(\dots \wedge \diamond(b_1 \wedge \diamond a) \dots))$,
("b_n...b₁ a" may not be consecutive)

□ $\langle a_0 \rangle_0 \langle a_1 \rangle_1 \dots \langle a_i \rangle_i \dots \langle a_{2^n-1} \rangle_{2^n-1}$ (proper sequence)

$p_n \dots p_1 a_i q_1 \dots q_n$

$p_n \dots p_1$: binary encoding for i

$q_n \dots q_1$: binary encoding for 2^n-1-

New encoding of computations

□◇ only checks for subsequences

Es. $\diamond(b_n \wedge \diamond(\dots \wedge \diamond(b_1 \wedge \diamond a) \dots))$,
("b_n...b₁ a" may not be consecutive)

□ $\langle a_0 \rangle_0 \langle a_1 \rangle_1 \dots \langle a_i \rangle_i \dots \langle a_{2^n-1} \rangle_{2^n-1}$ (proper sequence)

$p_n \dots p_1 a_i q_1 \dots q_n$

$p_n \dots p_1$: binary encoding for i

$q_n \dots q_1$: binary encoding for 2^n-1-i

$(p_j \in \{p_j^0, p_j^1\}, \quad q_j \in \{q_j^0, q_j^1\})$

Property of proper sequences

Property of proper sequences

□ For $\langle a_i \rangle_i = u a_i v$ (u-address, v-address):

- ◆ $\langle a_0 \rangle_0 \dots \langle a_{i-1} \rangle_{i-1} u$ is the shortest prefix containing u as a subsequence
- ◆ $v \langle a_{i+1} \rangle_{i+1} \dots \langle a_{2n-1} \rangle_{2n-1}$ is the shortest suffix containing v as a subsequence

□ Therefore:

- ◆ $u a v$ is a subseq of $\langle a_0 \rangle_0 \langle a_1 \rangle_1 \dots \langle a_{2n-1} \rangle_{2n-1}$ iff $a = a_i$

Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

$u=011$

000a111 001a011 010b101 011

$v=001$

01 100b110 101b010 110a100 111b000


Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

$u=011$

000a111 001a011 010b101 011

$v=001$

01 100b110 101b010 110a100 111b000

Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

u=011
┌──────────┬──┐
000a111 001a011 010b101 011

v=001

01 100b110 101b010 110a100 111b000

Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

u=011
┌───────────┐ ┌───────────┐
│ 000a111 001a011 010b101 011 │
└───────────┘ └───────────┘

v=001

01 100b110 101b010 110a100 111b000

Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

u=011
┌───────────┴───┐ ┌───────────┴───┐
000a111 001a011 010b101 011

v=001
└───┬───┘ └───┬───┘ └───┬───┘
01 100b110 101b010 110a100 111b000

Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

u=011
┌───────────┐ ┌───────────┐
│ 000a111 001a011 010b101 011 │
└───────────┘ └───────────┘

v=001

001 100b110 101b010 110a100 111b000

Example: proper sequences

□ 3-bits encoding of aababbab:

000a111 001a011 010b101 011a001

100b110 101b010 110a100 111b000

□ For $u=011$, $v=001$:

$u=011$

000a111 001a011 010b101 011

$v=001$

001 100b110 101b010 110a100 111b000

Talk Outline

- ✓ Overview
- ✓ Notation and general solution to LTL games
- ✓ Upper bounds: deterministic generators
- ⇒ Lower bounds
 - ✓ Encoding TMs without “next” and “until” U
 - ⇒ Expspace-hardness of $B(L_{\diamond, \wedge, \vee}(\Pi))$
 - ⇒ 2Exptime-hardness of $L_{\square, \diamond, \wedge, \vee}(\Pi)$
- Conclusions

Results

□ Th 1.

Deciding $L_{\square, \diamond, \wedge, \vee}(\Pi)$ games is 2Exptime-hard
(reduction from Alt. Expspace)

□ Th 2.

Deciding $B(L_{\diamond, \wedge, \vee}(\Pi))$ games is Expspace-hard
(reduction from Alt. Exptime)

Schema of our reductions

□ Protagonist (system)

- ◆ generates configurations
- ◆ picks transitions when TM in \exists -states

□ Adversary (environment)

- ◆ picks transitions when TM in \forall -states
- ◆ raises objections to check if the sequence of configurations is **proper** and **conforms** the **behaviour** of TM

Expspace-hardness

Expspace-hardness

- Protagonist generates sequences of positions $\langle a \rangle_i$
(i refers to configuration # and cell #)
- Plays:

Expspace-hardness

□ Protagonist generates sequences of positions $\langle a \rangle_i$
(i refers to configuration # and cell #)

□ Plays:

$u_0 a_0 v_0$

Expspace-hardness

□ Protagonist generates sequences of positions $\langle a \rangle_i$
(i refers to configuration # and cell #)

□ Plays:

$u_0 a_0 v_0$ ok

Expspace-hardness

- Protagonist generates sequences of positions $\langle a \rangle_i$
(i refers to configuration # and cell #)

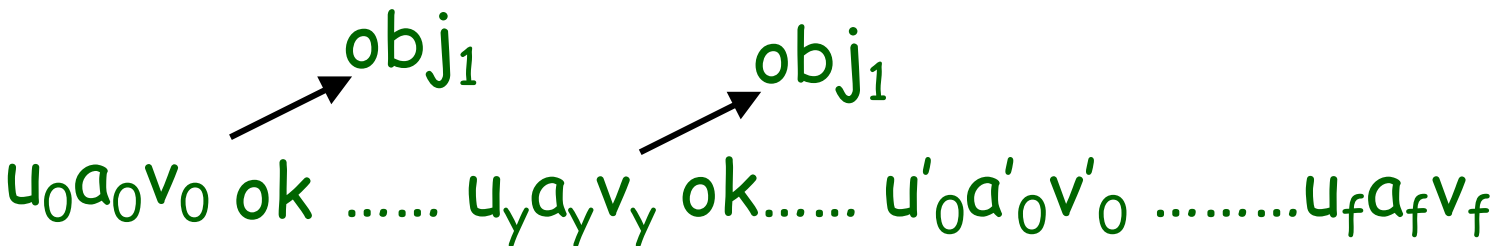
- Plays:

$u_0 a_0 v_0 \text{ ok}$ \nearrow obj_1

Expspace-hardness

- Protagonist generates sequences of positions $\langle a \rangle_i$
(i refers to configuration # and cell #)

- Plays:




Expspace-hardness

□ Protagonist generates sequences of positions $\langle a \rangle_i$

(i refers to configuration # and cell #)

□ Plays:

$u_0 a_0 v_0 \text{ ok} \dots u_y a_y v_y \text{ ok} \dots u'_0 a'_0 v'_0 \dots u_f a_f v_f \text{ ok ok} \dots$



The diagram shows two arrows pointing from the label 'obj1' to the configurations $u_y a_y v_y \text{ ok}$ and $u'_0 a'_0 v'_0$ in the sequence above.

Expspace-hardness

□ Protagonist generates sequences of positions $\langle a \rangle_i$

(i refers to configuration # and cell #)

□ Plays:

$u_0 a_0 v_0 \text{ ok} \dots u_y a_y v_y \text{ ok} \dots u'_0 a'_0 v'_0 \dots u_f a_f v_f \text{ ok ok} \dots$

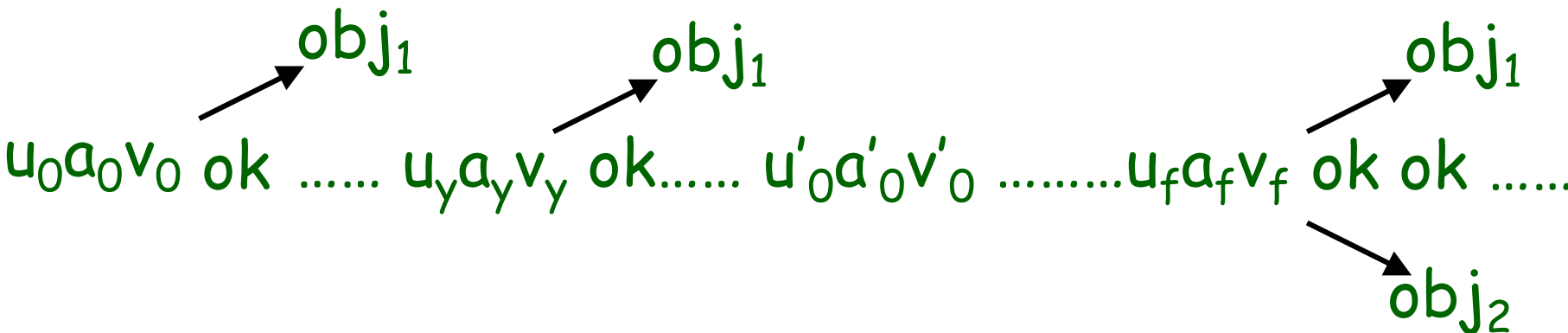
$\nearrow \text{obj}_1 \quad \nearrow \text{obj}_1 \quad \nearrow \text{obj}_1$

Expspace-hardness

□ Protagonist generates sequences of positions $\langle a \rangle_i$

(i refers to configuration # and cell #)

□ Plays:



Objection 1

Objection 1

□ Generation of proper sequences:

◆ verify $n(u_{j+1})=n(u_j)+1$ and $n(v_j)=2^n-1- n(u_j)$

... $p_n \dots p_1 a_j q_1 \dots q_n \dots \text{obj}_1 r_n \dots r_1 s_n \dots s_1$

Objection 1

□ Generation of proper sequences:

◆ verify $n(u_{j+1})=n(u_j)+1$ and $n(v_j)=2^n-1- n(u_j)$

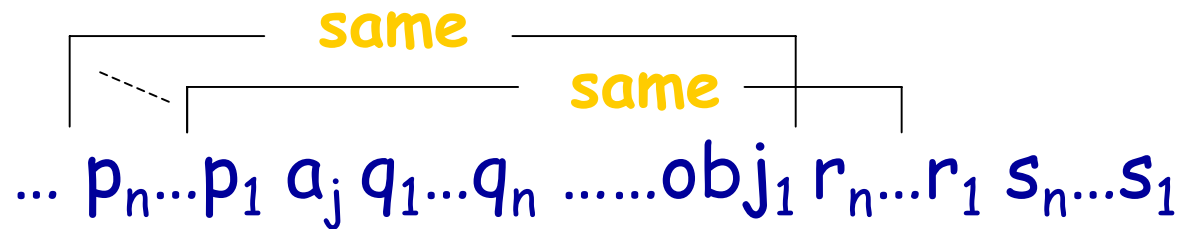
same

... $p_n \dots p_1$ a_j $q_1 \dots q_n$ obj_1 $r_n \dots r_1$ $s_n \dots s_1$

Objection 1

□ Generation of proper sequences:

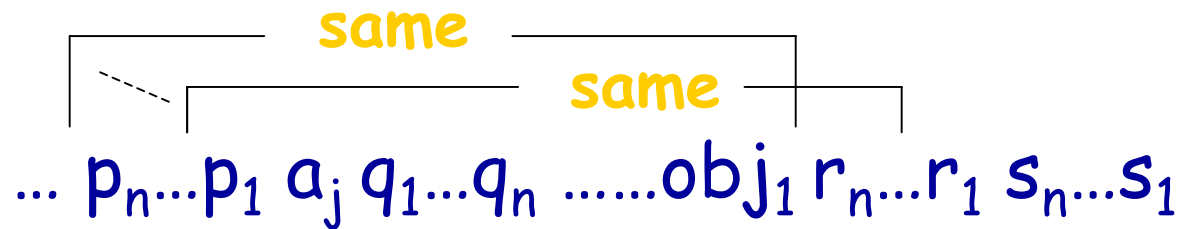
◆ verify $n(u_{j+1})=n(u_j)+1$ and $n(v_j)=2^n-1- n(u_j)$



Objection 1

□ Generation of proper sequences:

◆ verify $n(u_{j+1})=n(u_j)+1$ and $n(v_j)=2^n-1- n(u_j)$

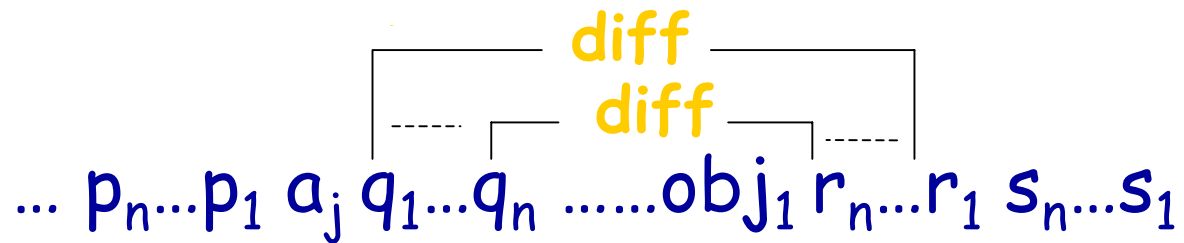


$$(p_j^0 \wedge \diamond r_j^0) \vee (p_j^1 \wedge \diamond r_j^1)$$

Objection 1

□ Generation of proper sequences:

◆ verify $n(u_{j+1})=n(u_j)+1$ and $n(v_j)=2^n-1- n(u_j)$



$$(q_j^0 \wedge \diamond r_j^1) \vee (q_j^1 \wedge \diamond r_j^0)$$

Formula for proper sequences

$$\diamond \text{obj}_1 \longrightarrow ($$
$$[(\text{succ}(r, s) \wedge \varphi_1) \longrightarrow \varphi_2] \wedge$$
$$(\varphi'_1 \longrightarrow \varphi'_2)$$
$$)$$

Formula for proper sequences

$$\diamond \text{obj}_1 \longrightarrow ($$
$$[(\text{succ}(r, s) \wedge \varphi_1) \longrightarrow \varphi_2] \wedge$$
$$(\varphi'_1 \longrightarrow \varphi'_2)$$
$$)$$

φ_1 = "p is same as r"

φ_2 = "p is same as r followed by p is same as s"

Formula for proper sequences

$$\diamond \text{obj}_1 \longrightarrow ($$
$$[(\text{succ}(r, s) \wedge \varphi_1) \longrightarrow \varphi_2] \wedge$$
$$(\varphi'_1 \longrightarrow \varphi'_2)$$
$$)$$

φ'_1 = "p is same as r"

φ'_2 = "p is same as r followed by q **diff** from r"

Formula for proper sequences

$$\diamond \text{obj}_1 \longrightarrow ([(\text{succ}(r,s) \wedge \varphi_1) \longrightarrow \varphi_2] \wedge (\varphi'_1 \longrightarrow \varphi'_2))$$

φ'_1 = "p is same as r"

φ'_2 = "p is same as r followed by q **diff** from r"

➤ Need only formulas in $B(L_{\diamond, \wedge, \vee}(\Pi))$

Objection 2

- Verify that sequences are TM outcomes
- Adversary picks $i-1, i, i+1$, and j , and checks if cell i of C_{j+1} can “follow” cells $i-1, i, i+1$ of C_j
- “Small” formulas from $B(L_{\diamond, \wedge, \vee}(\Pi))$ do the job
(property of proper sequences is crucial to match cell contents using only nested \diamond)
- TM computes in exptime:
 - ◆ at the end of a computation we can zoom to each position generating polynomially many bits

Results

□ Th 1.

Deciding $L_{\square, \diamond, \wedge, \vee}(\Pi)$ games is 2Exptime-hard
(reduction from Alt. Expspace)

□ Th 2.

Deciding $B(L_{\diamond, \wedge, \vee}(\Pi))$ games is Expspace-hard
(reduction from Alt. Exptime)

2 Exptime-hardness

2Exptime-hardness

- We cannot encode configuration #
- We can still use proper sequences to zoom to cells within a configuration
- Focus on 2 consecutive configurations at a time
(modulo-3 counter incremented every time a new configuration is entered)

Objections

- Objection 1 similar to previous case
- Objection 2 is allowed at the end of every configuration
- To check φ from the penultimate configuration use obj_2 along with:
 - $\bigvee_{j \in \{0,1,2\}} (\diamond(j \wedge \varphi \wedge \diamond(j+1)) \wedge \neg \diamond(j+2))$

Objections

- Objection 1 similar to previous case
- Objection 2 is allowed at the end of every configuration
- To check φ from the penultimate configuration use obj_2 along with:
 - $\bigvee_{j \in \{0,1,2\}} (\diamond(j) \wedge \varphi \wedge \diamond(j+1) \wedge \neg \diamond(j+2))$

Objections

- Objection 1 similar to previous case
- Objection 2 is allowed at the end of every configuration
- To check φ from the penultimate configuration use obj_2 along with:
 - $\bigvee_{j \in \{0,1,2\}} (\diamond(j \wedge \varphi \wedge \diamond(j+1) \wedge \neg \diamond(j+2)))$

(This is in $L_{\square, \diamond, \wedge, \vee}(\Pi)$)

Complexity

	Games	Det. Generators	
		Size	L. Dist.
$B(L_{\diamond, \wedge}(\Pi))$	Pspace-complete	$\Theta(\text{Exp})$	$\Theta(\text{Linear})$
$B(L_{\diamond, \circ, \wedge}(\Pi))$	Exptime-complete	$\Theta(\text{Exp})$	$\Theta(\text{Exp})$
$B(L_{\diamond, \wedge, \vee}(\Pi))$	ExpSPACE-complete	$\Theta(2\text{Exp})$	$\Theta(\text{Exp})$
$B(L_{\diamond, \circ, \wedge, \vee}(\Pi))$	ExpSPACE-complete	$\Theta(2\text{Exp})$	$\Theta(\text{Exp})$
$L_{\square, \diamond, \wedge, \vee}(\Pi)$	2Exptime-complete	$\Theta(2\text{Exp})$	$\Theta(2\text{Exp})$
LTL	2Exptime-complete	$\Theta(2\text{Exp})$	$\Theta(2\text{Exp})$

Talk Outline

- ✓ Overview
- ✓ Notation and general solution to LTL games
- ✓ Upper bounds: deterministic generators
- ✓ Lower bounds
 - ✓ Encoding TMs without "next" and "until" U
 - ✓ Expspace-hardness of $B(L_{\square, \diamond, \wedge, \vee}(\Pi))$
 - ✓ 2Exptime-hardness of $L_{\square, \diamond, \wedge, \vee}(\Pi)$
- ⇒ Conclusions

Fair safety-reachability games

□ Games with fairness:

◆ "(adv plays fair) \rightarrow (prot plays fair \wedge wins)

◆ "(prot plays fair) \wedge (adv plays fair \rightarrow wins)

□ $B(L_{\square\lozenge}(\Pi) \cup L_{\lozenge,\wedge}(\Pi)) : (B(L_{\lozenge,\wedge}(\Pi)))^F$
fair safety-reachability games

□ $B(L_{\lozenge,\wedge}(\Pi))^F$ games are Pspace-complete

Fair safety-reachability games

□ Games with fairness:

◆ "(adv plays fair) \rightarrow (prot plays fair \wedge wins)"

◆ "(prot plays fair) \wedge (adv plays fair \rightarrow wins)"

□ $B(L_{\square\lozenge}(\Pi) \cup L_{\lozenge,\wedge}(\Pi)) : (B(L_{\lozenge,\wedge}(\Pi)))^F$) fair safety-reachability games

□ $B(L_{\lozenge,\wedge}(\Pi))^F$ games are Pspace-complete

Decision algorithm uses Zielonka solution to Muller games along with det. generators for $L_{\lozenge,\wedge}(\Pi)$

Fair safety-reachability games

□ Games with fairness:

◆ "(adv plays fair) \rightarrow (prot plays fair \wedge wins)"

◆ "(prot plays fair) \wedge (adv plays fair \rightarrow wins)"

□ $B(L_{\square\lozenge}(\Pi) \cup L_{\lozenge,\wedge}(\Pi)) : (B(L_{\lozenge,\wedge}(\Pi)))^F$) fair safety-reachability games

□ $B(L_{\lozenge,\wedge}(\Pi))^F$ games are Pspace-complete

Hardness: games with "Streett V Rabin" winning conditions are Pspace-hard (from QBF)

More in PSPACE

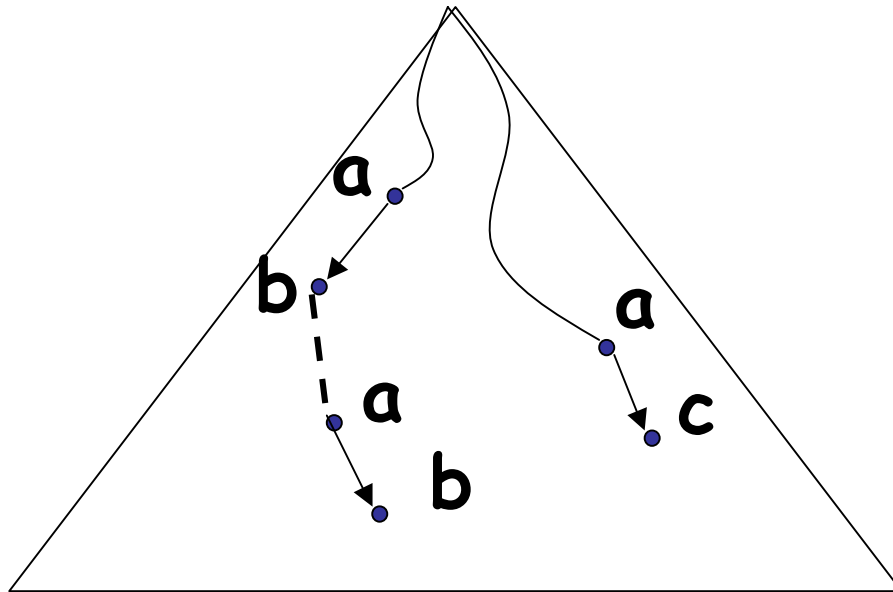
□ Persistent strategy:

On a play, the player picks always the same move visiting the same location
(weaker than memoryless)

More in PSPACE

□ Persistent strategy:

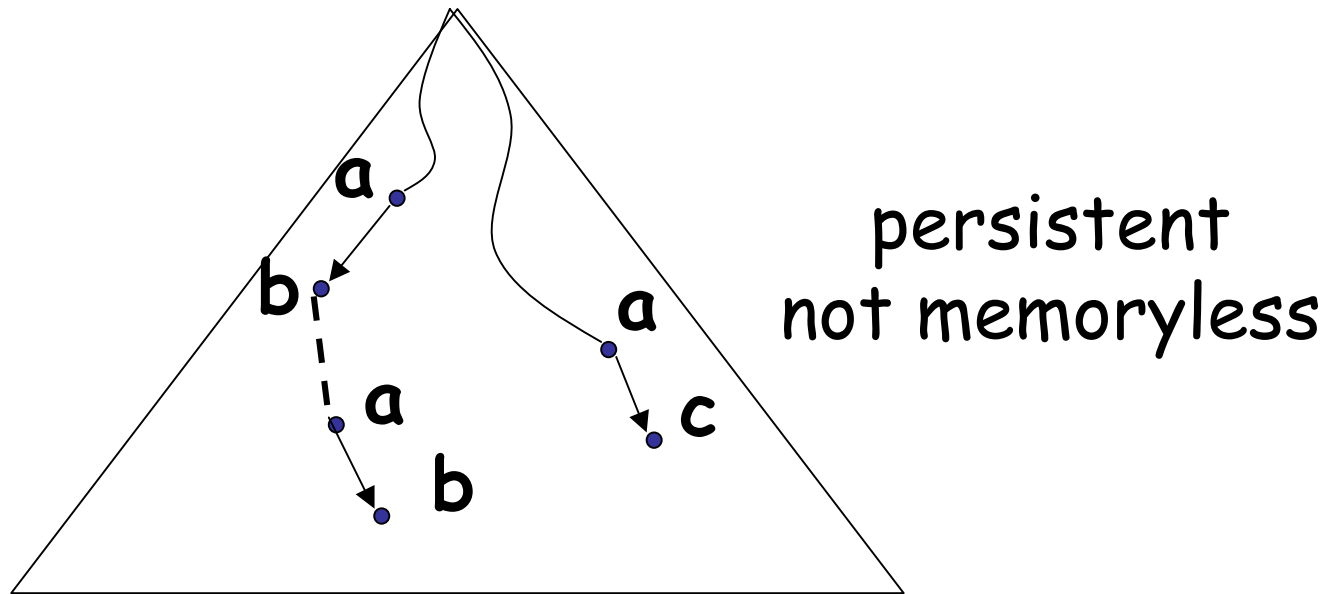
On a play, the player picks always the same move visiting the same location
(weaker than memoryless)



More in PSPACE

□ Persistent strategy:

On a play, the player picks always the same move visiting the same location
(weaker than memoryless)



Complexity of $L_{\diamond, \wedge, \vee}(\Pi)$

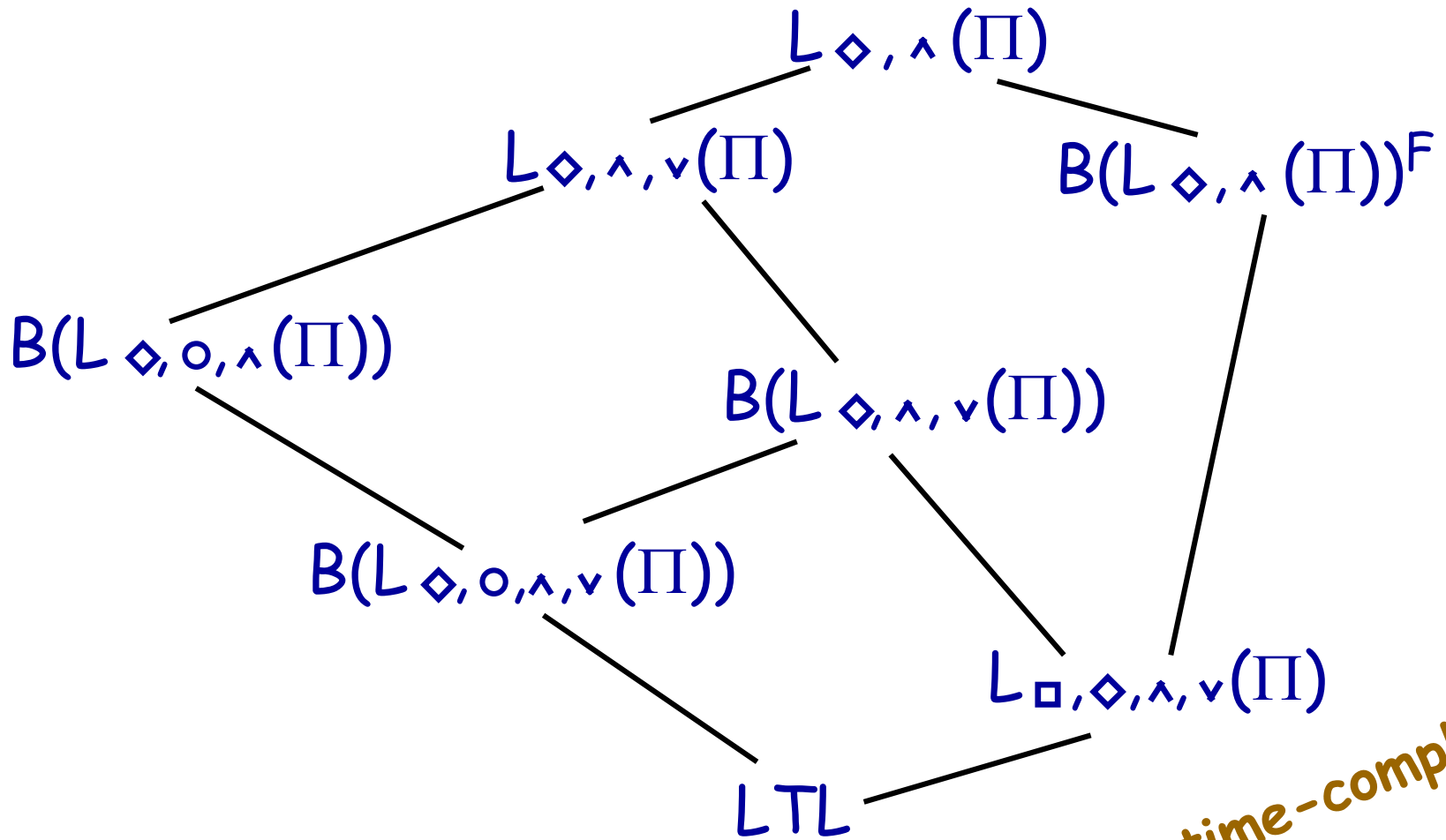
□ **Theorem:** [Marcinkowski - Truderung CSL'02]

For specs in $L_{\diamond, \wedge, \vee}(\Pi)$,

protagonist has a winning strategy **iff**
can win against an adversary that uses
only persistent strategies

□ $L_{\diamond, \wedge, \vee}(\Pi)$ games are in PSPACE

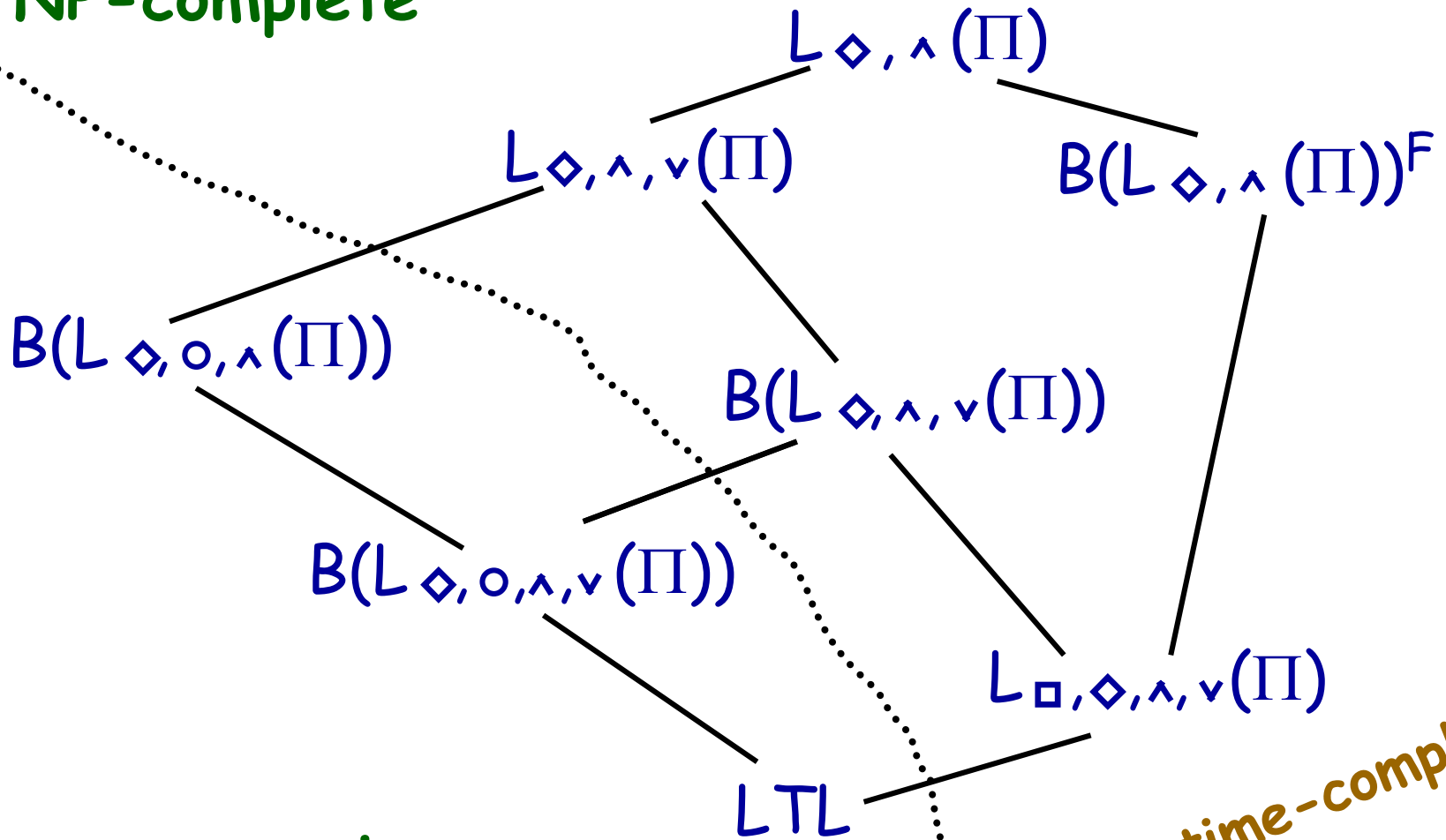
LTL fragments



2Exptime-complete

Complexity: Model-checking

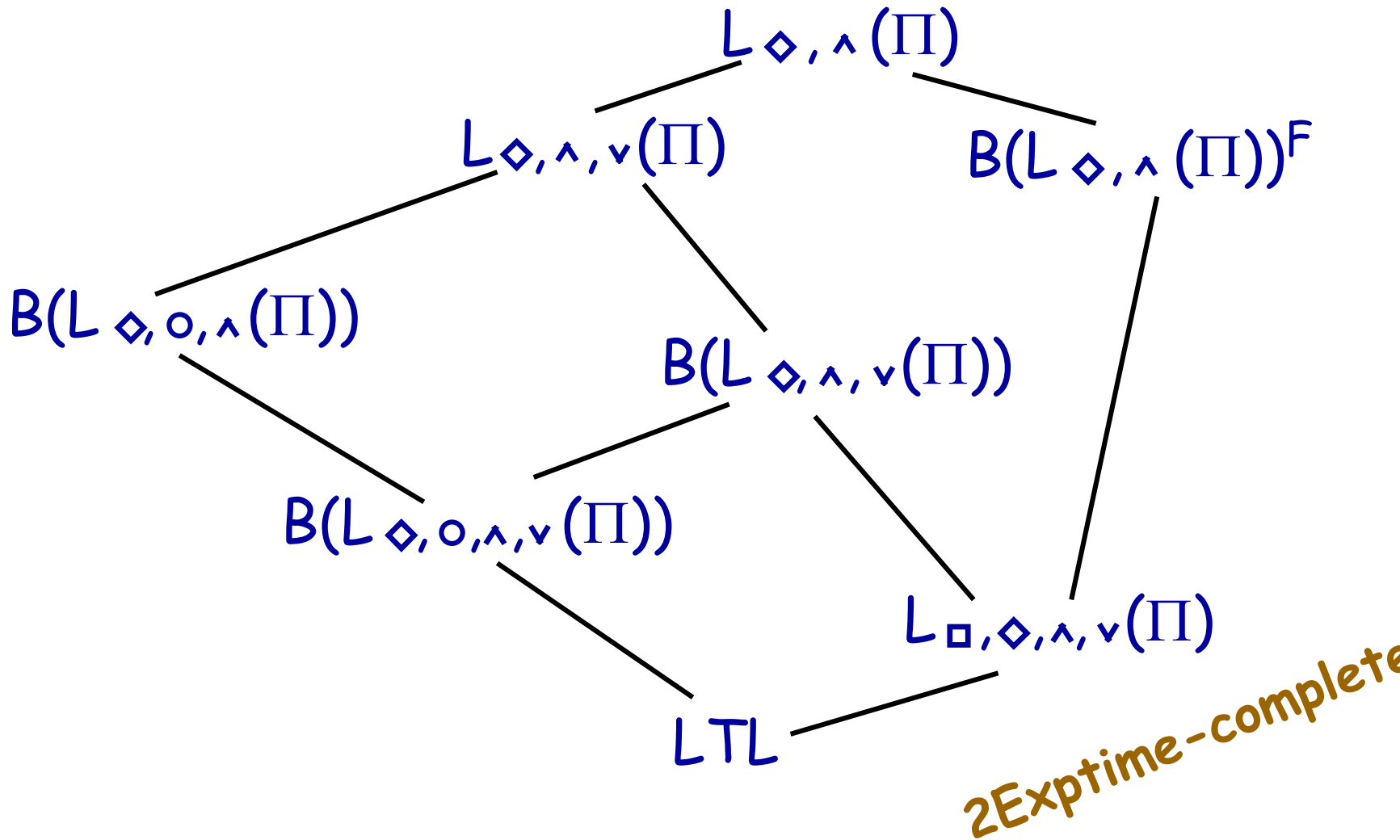
NP-complete



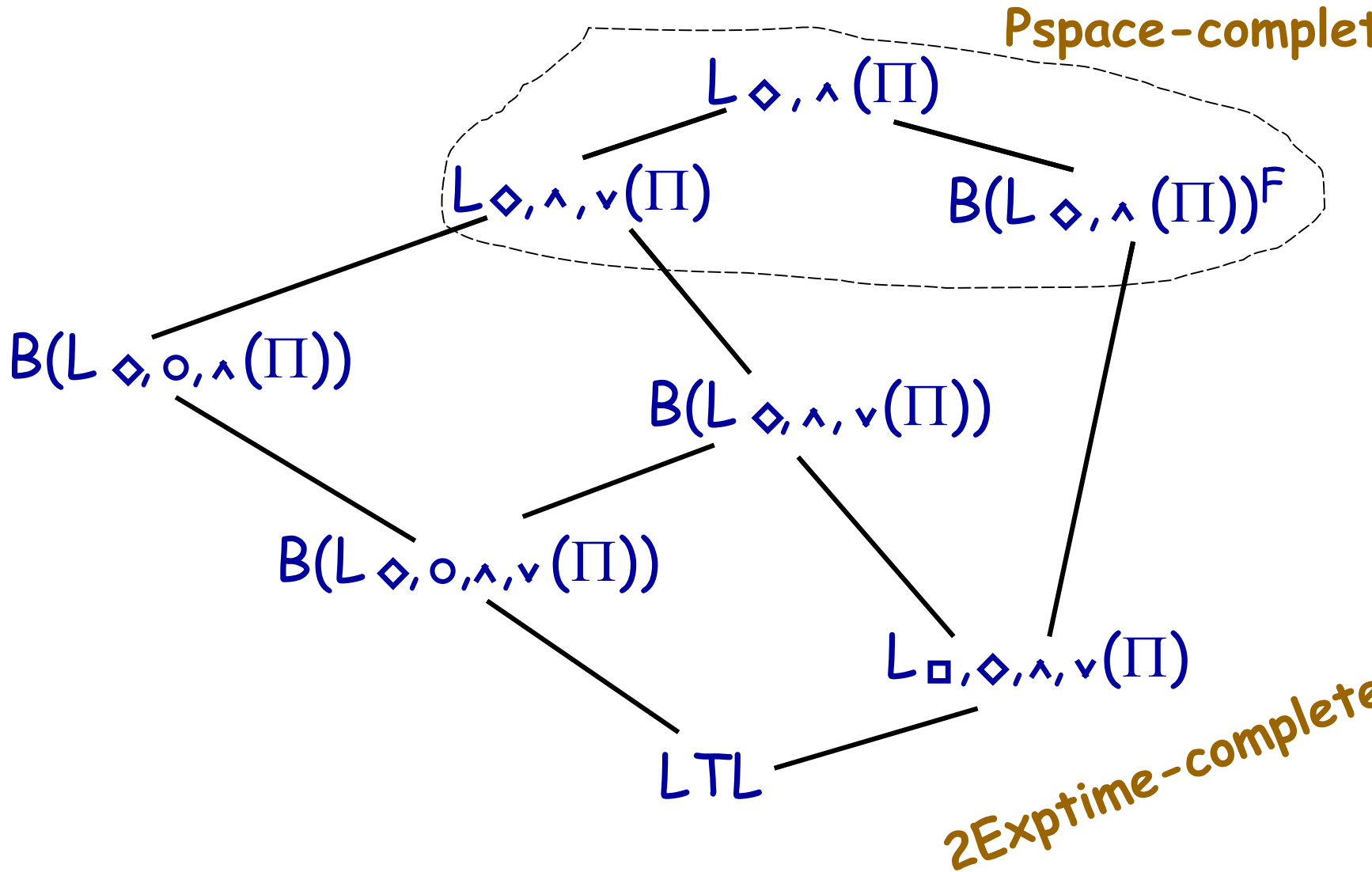
$2^{Exptime}$ -complete

Pspace-complete

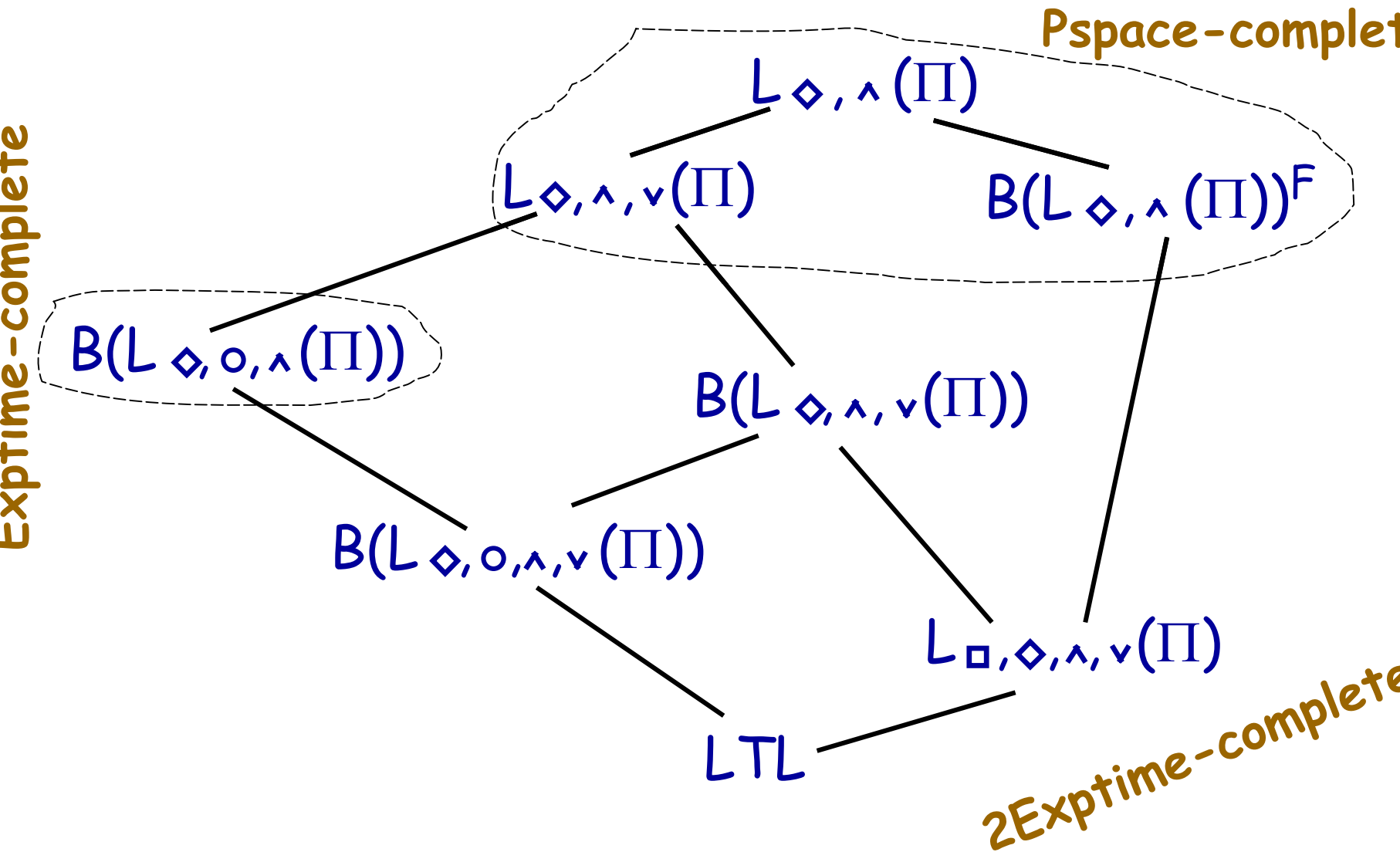
Complexity: Games



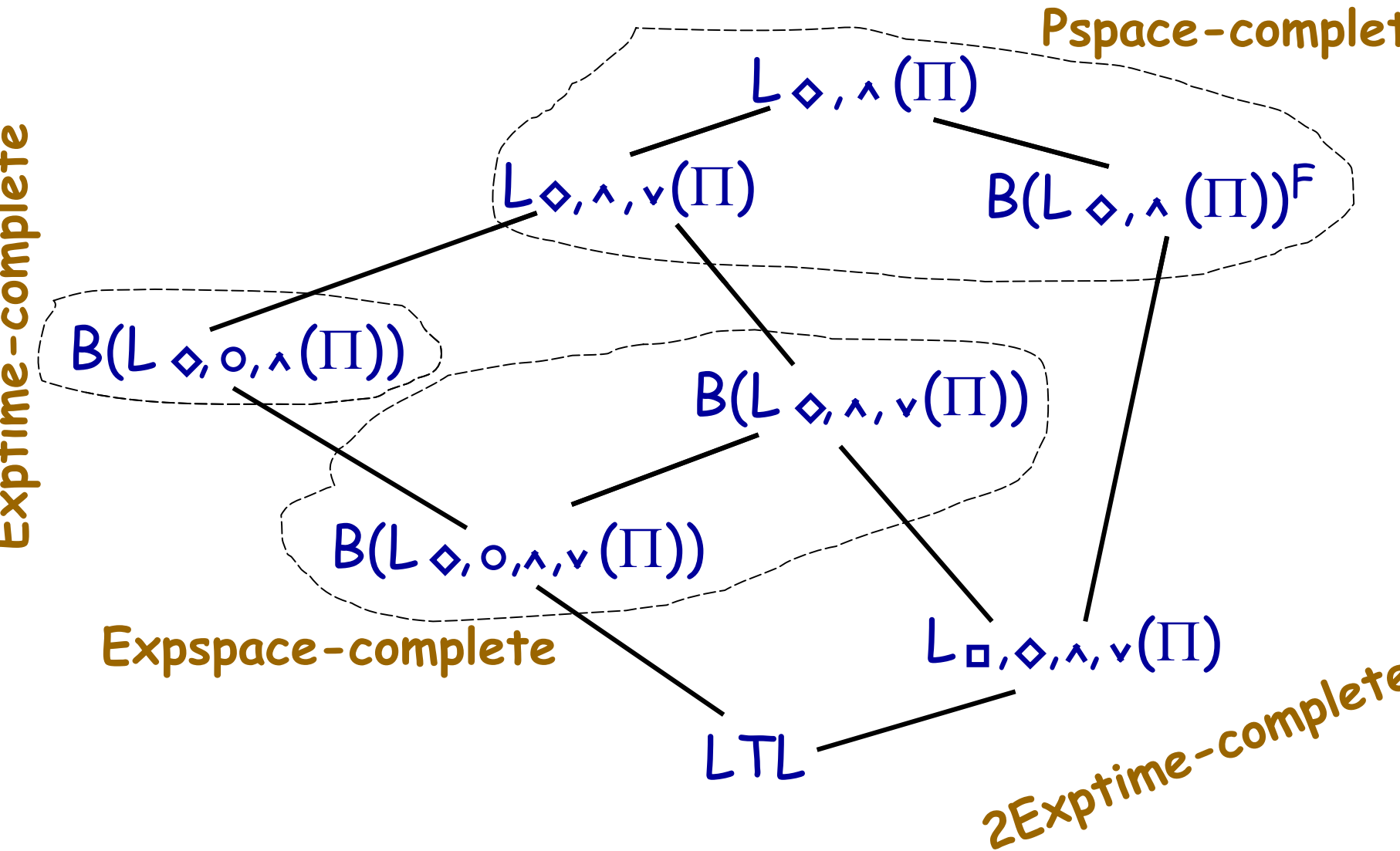
Complexity: Games



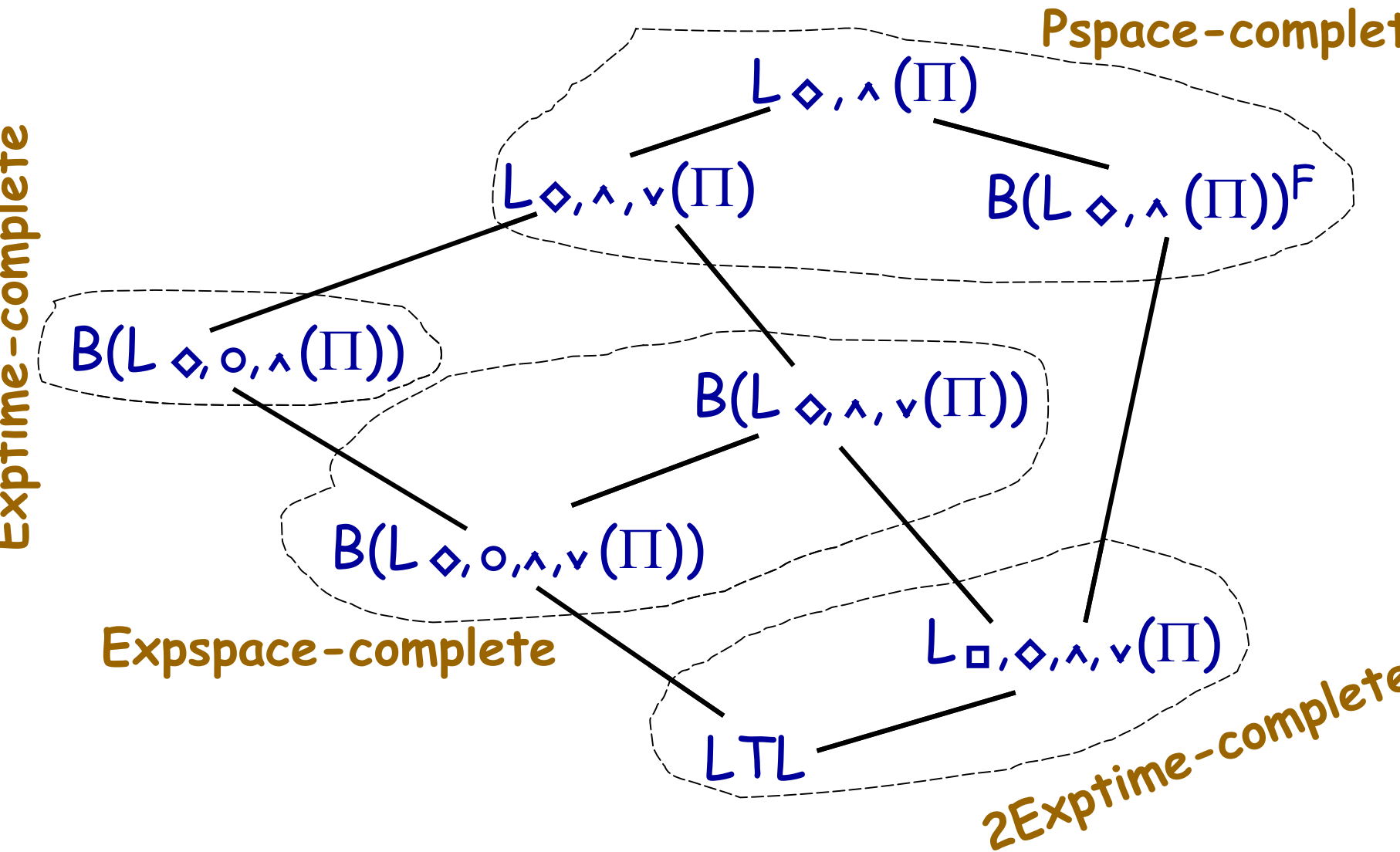
Complexity: Games



Complexity: Games



Complexity: Games

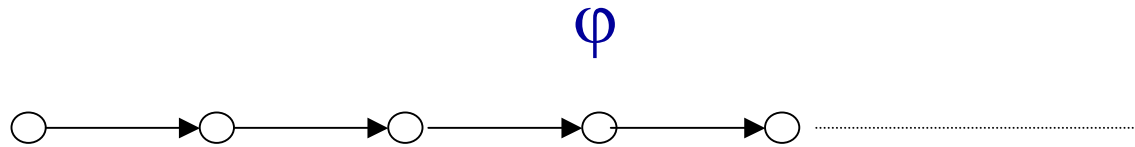


Computational Complexity

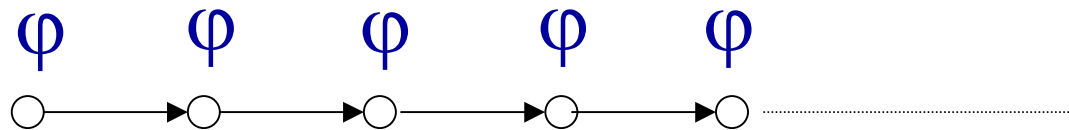
	Games	Model-checking
$L_{\diamond, \wedge}(\Pi)$	Pspace-complete	NP-complete
$B(L_{\diamond, \wedge}(\Pi))^F$	Pspace-complete	NP-complete
$L_{\diamond, \wedge, \vee}(\Pi)$	Pspace-complete	NP-complete
$B(L_{\diamond, \circ, \wedge}(\Pi))$	Exptime-complete	Pspace-complete
$B(L_{\diamond, \wedge, \vee}(\Pi))$	Expspace-complete	NP-complete
$B(L_{\diamond, \circ, \wedge, \vee}(\Pi))$	Expspace-complete	Pspace-complete
$L_{\square, \diamond, \wedge, \vee}(\Pi)$	2Exptime-complete	NP-complete
LTL	2Exptime-complete	Pspace-complete

Box and Diamond

$\square \diamond \varphi$ (eventually φ):



$\square \square \varphi$ (always φ):



"□-◇" fragments

□ $L_{\square, \diamond, \wedge, \vee}(\Pi)$: full "□ - ◇" LTL fragment

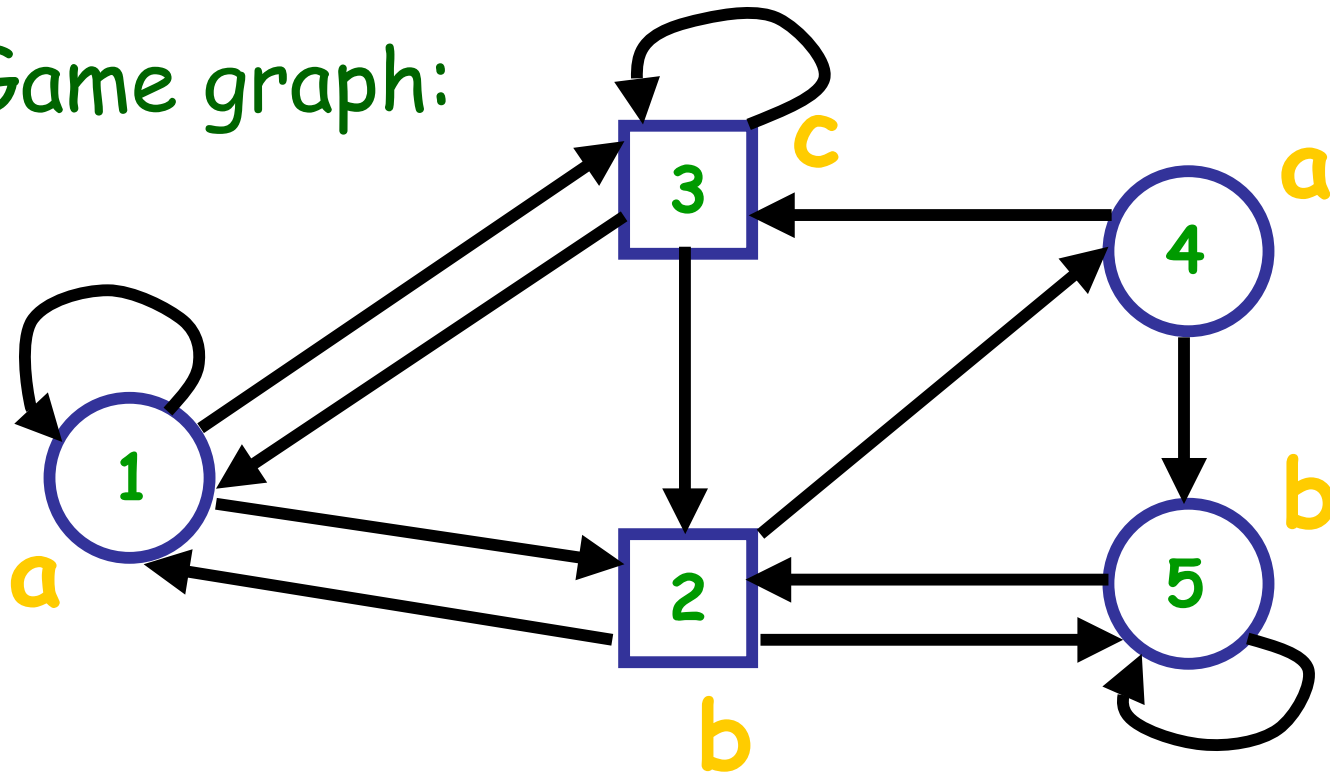
□ $B(L_{\diamond, \wedge, \vee}(\Pi))$: boolean combinations of

$$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \diamond \varphi, \quad p \in \Pi$$

(no □ in the scope of ◇ and vice-versa)

LTL Games

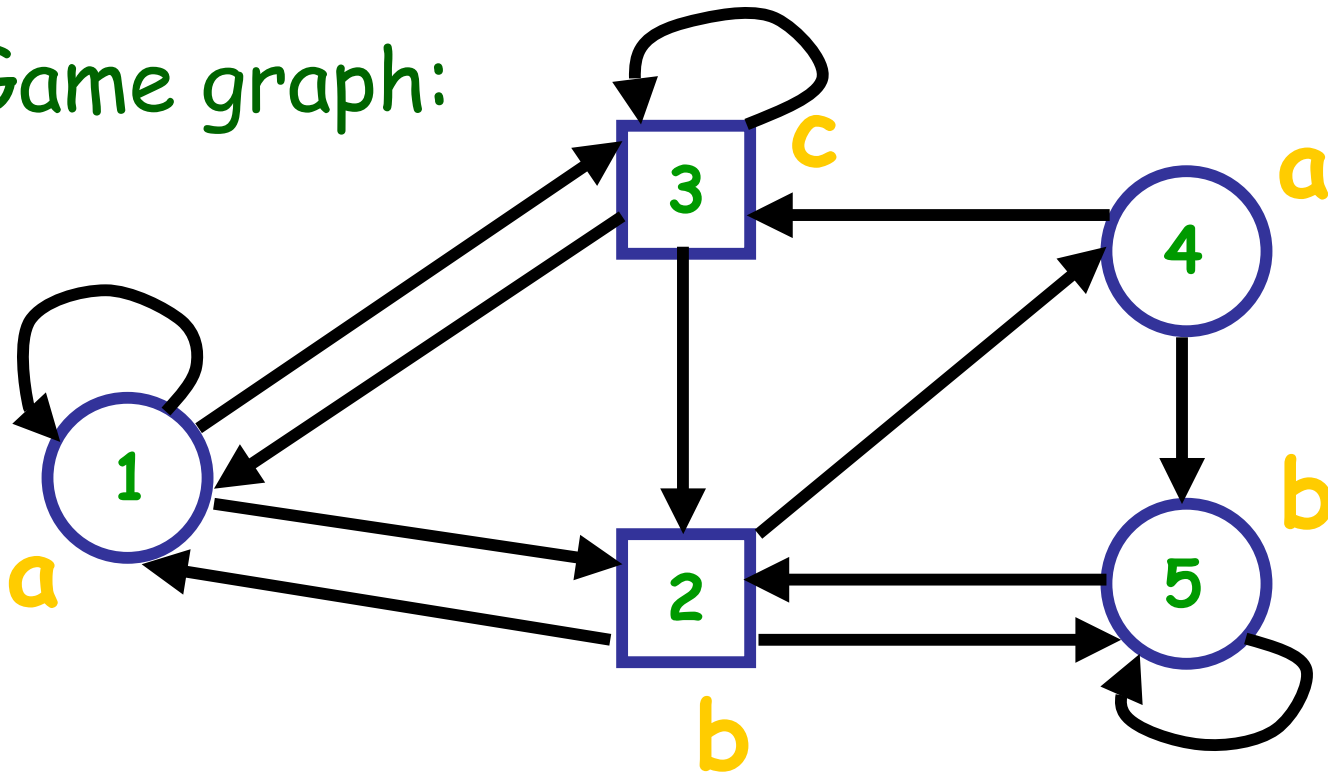
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

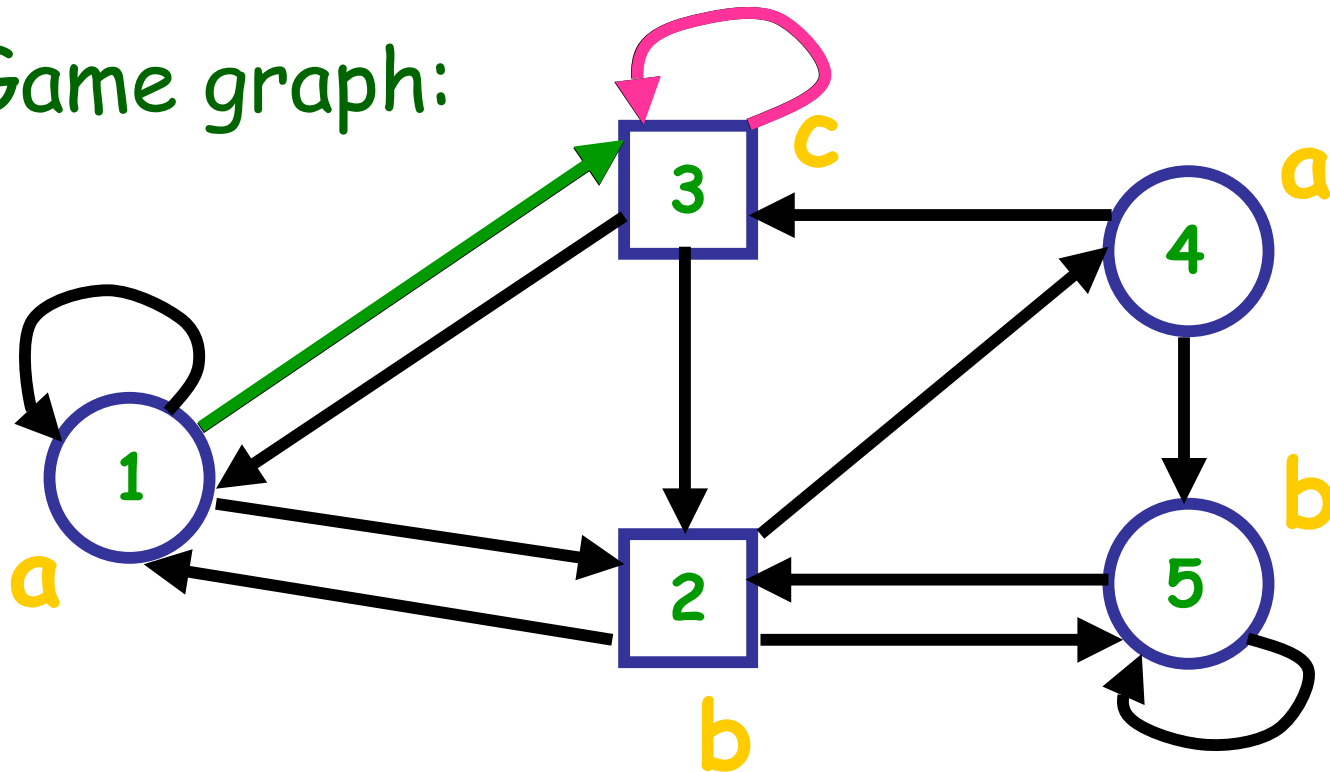
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

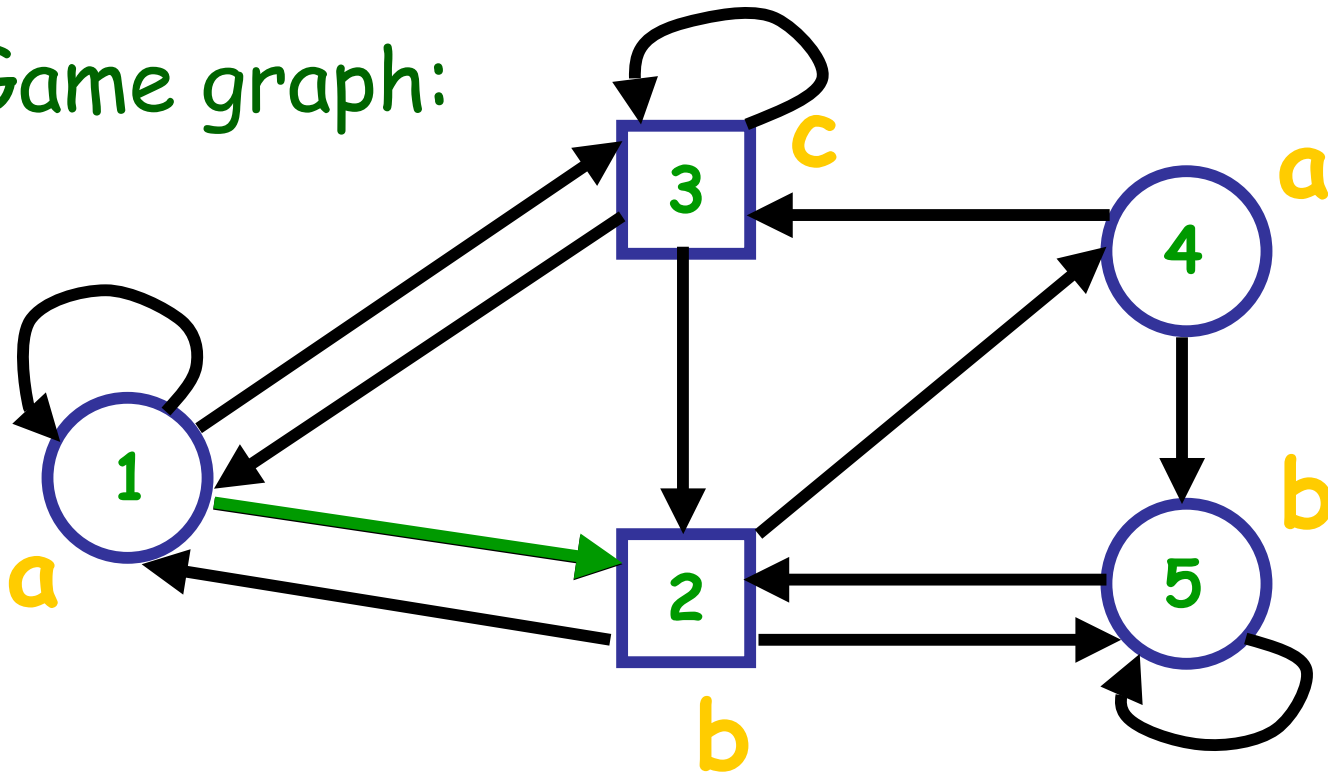
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

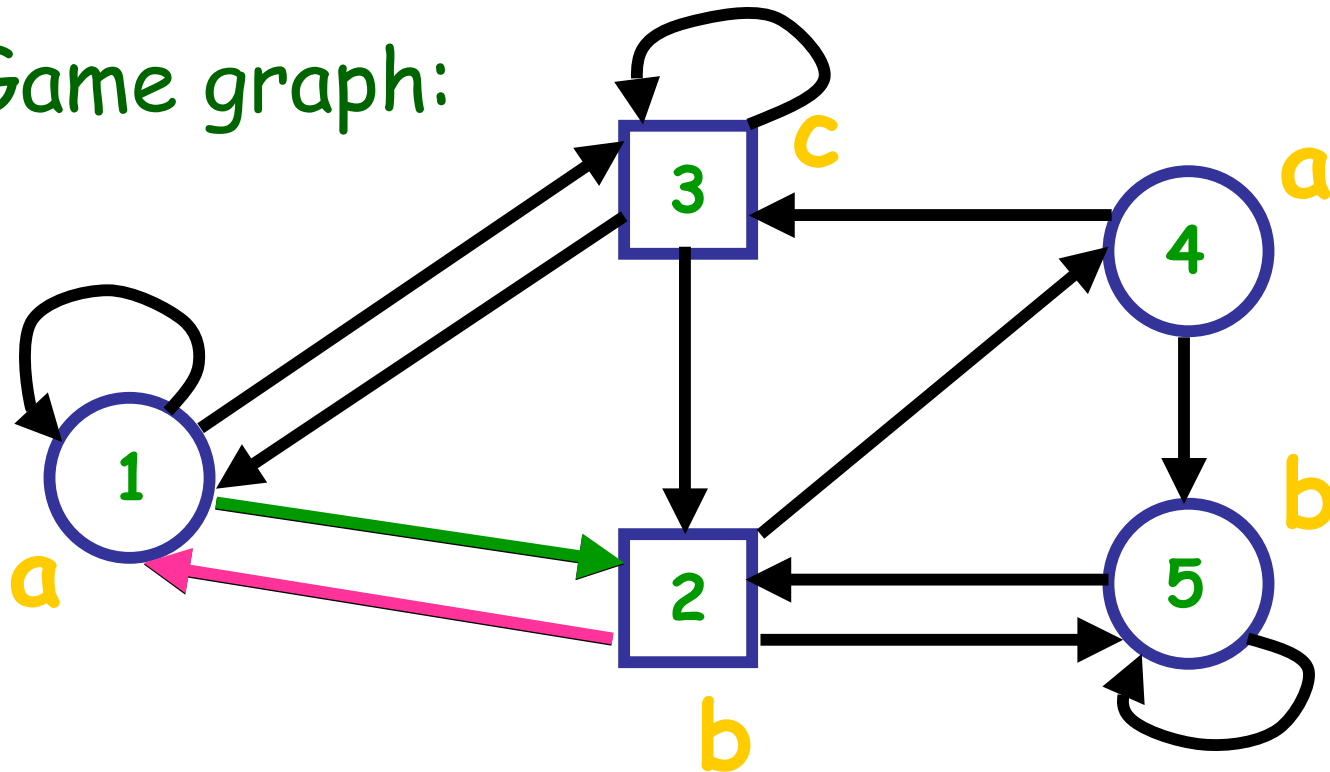
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

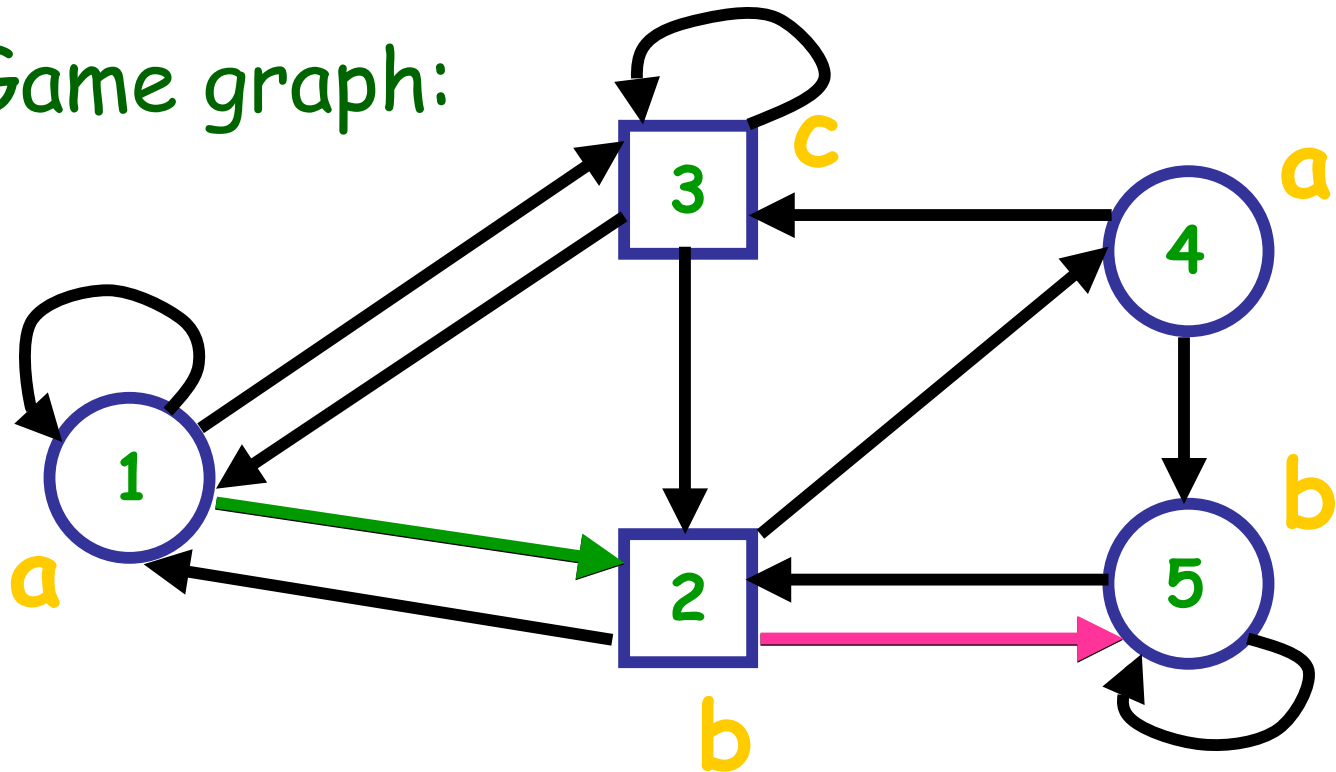
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

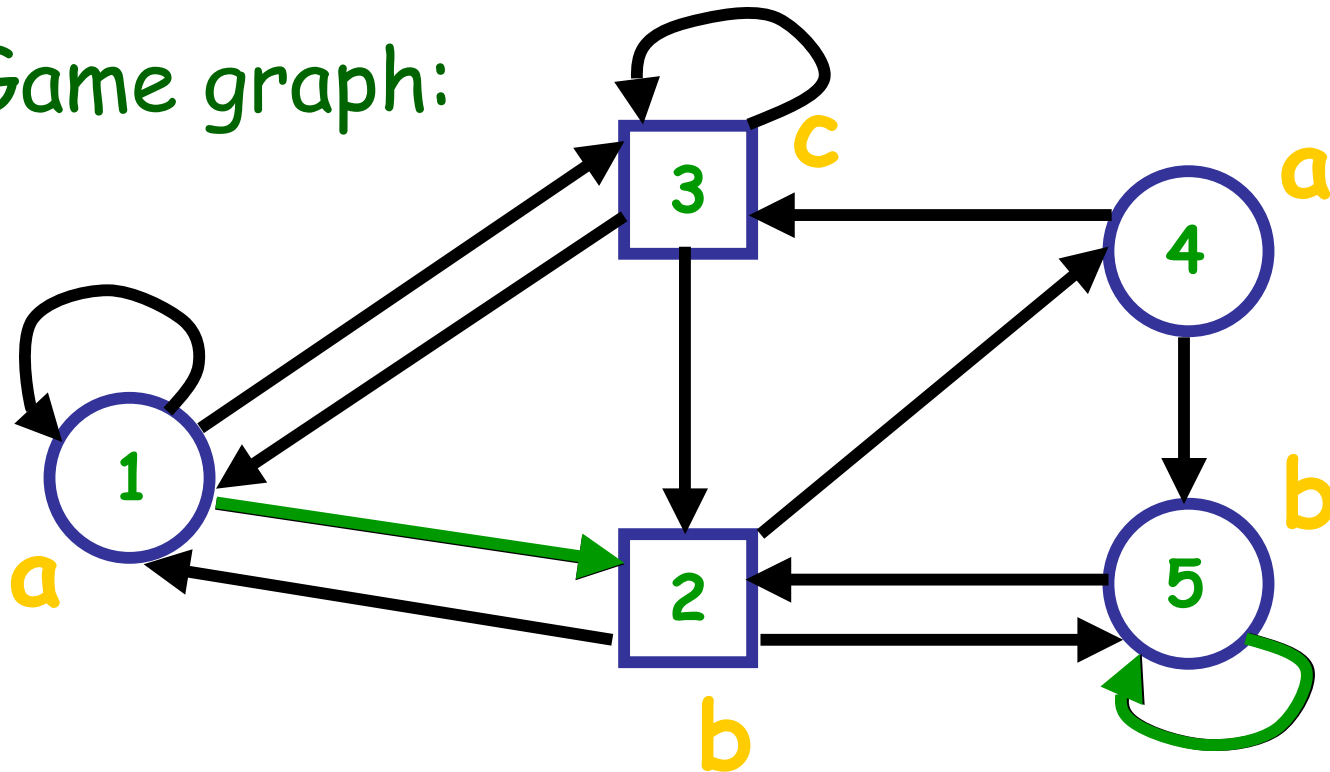
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

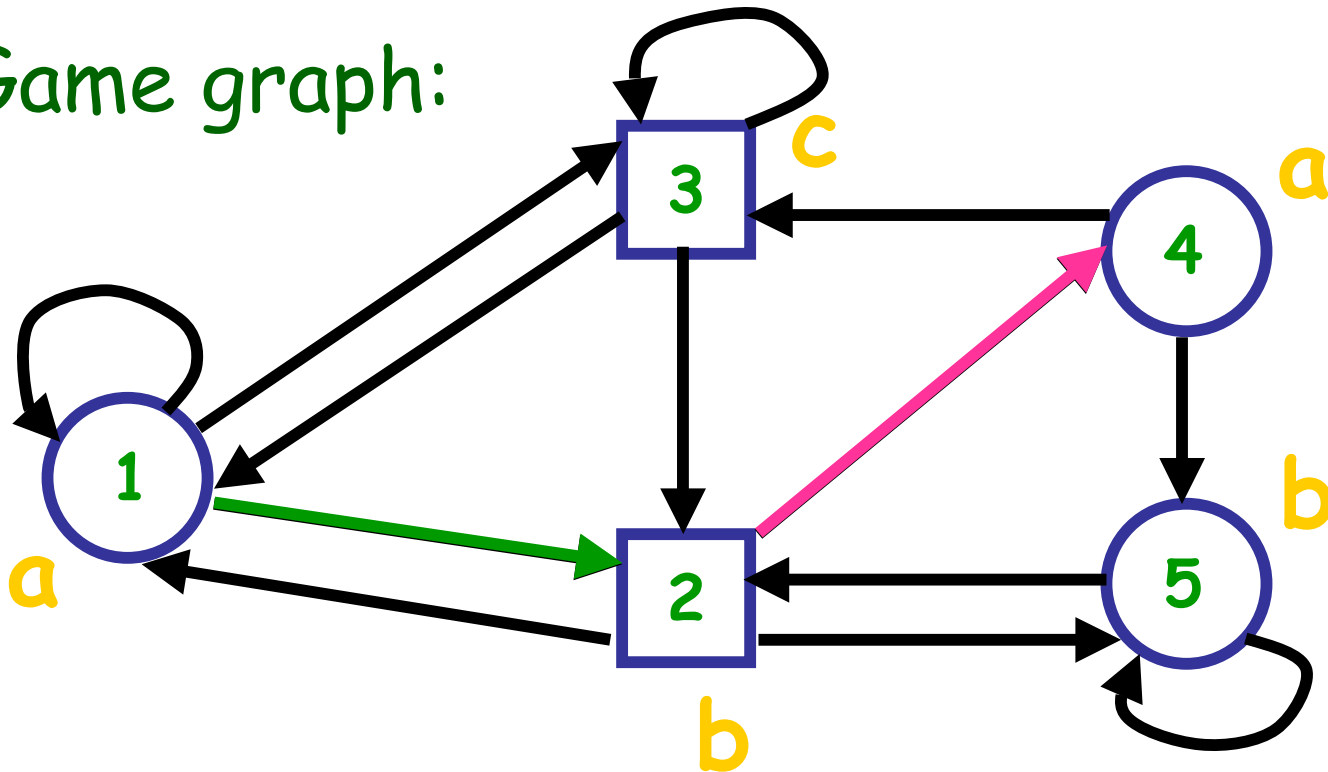
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

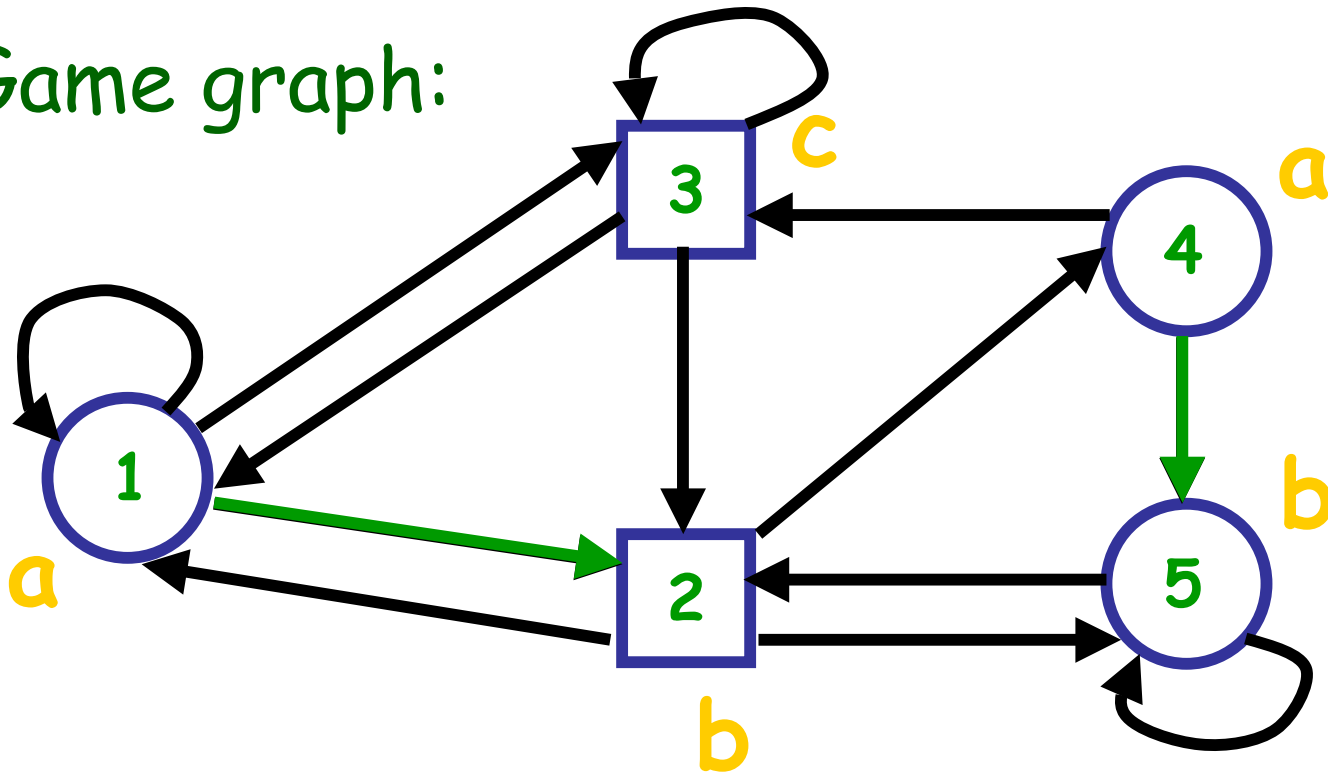
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

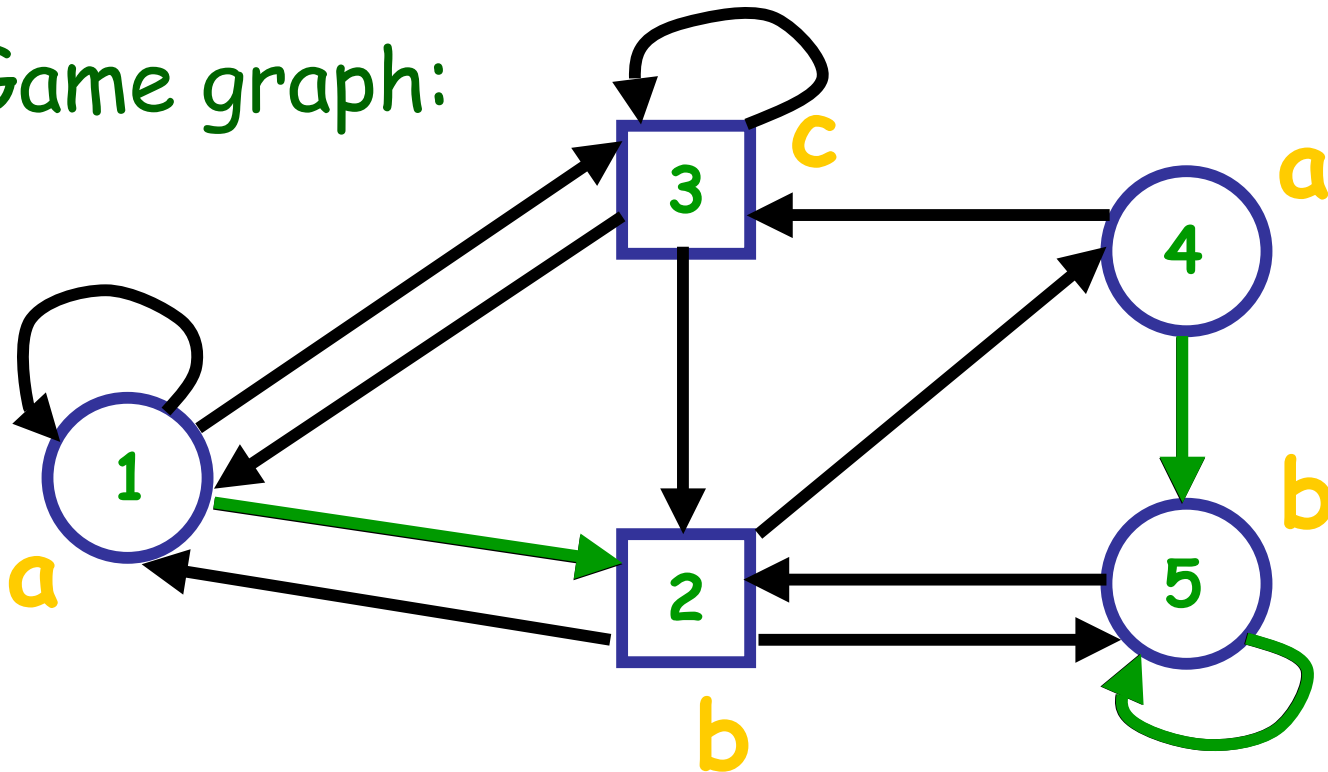
➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Decision problem: Is there a winning strategy of the protagonist?

➤ Game graph:



➤ Specification: $\square (a \rightarrow \diamond b)$

Computational Complexity of LTL Games

Computational Complexity of LTL Games

- Deciding LTL games is
2Exptime-complete [PR'89]
- What about games in LTL fragments?
- Previous research [AL'01] & [MT'02]
- Focus on fragments using only
"always" (\square) and "eventually" (\diamond)
(no "until" or "next" are allowed)

Our results

\wedge

\vee

\diamond

\square

Our results

□ Full “□ - ◇” LTL fragment

$$\varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \diamond \varphi \mid \square \varphi$$

□ Games are 2Exptime-hard as for LTL

□ Not allowing □ in the scope of ◇ and vice-versa games become Expspace-complete

◆ Expspace membership from [AL'01]

◆ Using only either □ or ◇ games are in Pspace [MT'02]

□ Games with safety and reachability specs augmented with fairness conditions are Pspace-complete

LTL Games

- Winning condition is LTL formula
- $G =$ game graph, $\varphi =$ LTL formula
 - ◆ Construct det. generator A of φ models
 - ◆ Solve the game $(G \times A, W)$
(W is the acceptance condition of A on $G \times A$)
- 2Exptime-complete [PR'89]

Motivation

- Game complexity is lower for Buchi, Rabin, and Streett games
- Model-checking is also easier in some LTL fragments
- What about games in LTL fragments?

Problem 2: consecutive configs

\wedge

\wedge

Problem 2: consecutive configs

□ If “until” (U) is allowed then:

- ◆ Modulo-2 counter to distinguish among consecutive configurations
- ◆ Constructs of type $(0 \wedge \varphi_0) \cup (1 \wedge \varphi_1)$

□ Without “next” and “until”?

- ◆ If # of configurations is $O(2^n)$, then number configurations (same as for cells)
- ◆ Otherwise, we need more ...

Linear Temporal Logic (LTL)

- Correctness requirements for reactive systems
- Game-based interpretation:
 - ◆ controller synthesis
 - ◆ compositionality requirements
 - ◆ verification of open systems
 - ◆ modular verification (module-checking)

Zoom to the last two configs

□ Configurations are counted with a modulo-3 counter

◆ use 3 new atomic propositions

◆ the same propositions hold true on all cells of a configuration

□ To check φ from the penultimate configuration use:

◆ $\bigvee_{j \in \{0,1,2\}} (\diamond(j \wedge \varphi \wedge \diamond(j+1) \wedge \neg \diamond(j+2)))$

Zoom to the last two configs

□ Configurations are counted with a modulo-3 counter

◆ use 3 new atomic propositions

◆ the same propositions hold true on all cells of a configuration

□ To check φ from the penultimate configuration use:

◆ $\bigvee_{j \in \{0,1,2\}} (\diamond(j \wedge \varphi \wedge \diamond(j+1) \wedge \neg \diamond(j+2)))$

Zoom to the last two configs

□ Configurations are counted with a modulo-3 counter

◆ use 3 new atomic propositions

◆ the same propositions hold true on all cells of a configuration

□ To check φ from the penultimate configuration use:

◆ $\bigvee_{j \in \{0,1,2\}} (\diamond(j \wedge \varphi \wedge \diamond(j+1) \wedge \neg \diamond(j+2)))$

Expspace-hardness

□ Objection 1:

- ◆ adversary selects 2 consecutive positions
- ◆ protagonist loses if these positions witness that the sequence is not proper

□ Objection 2:

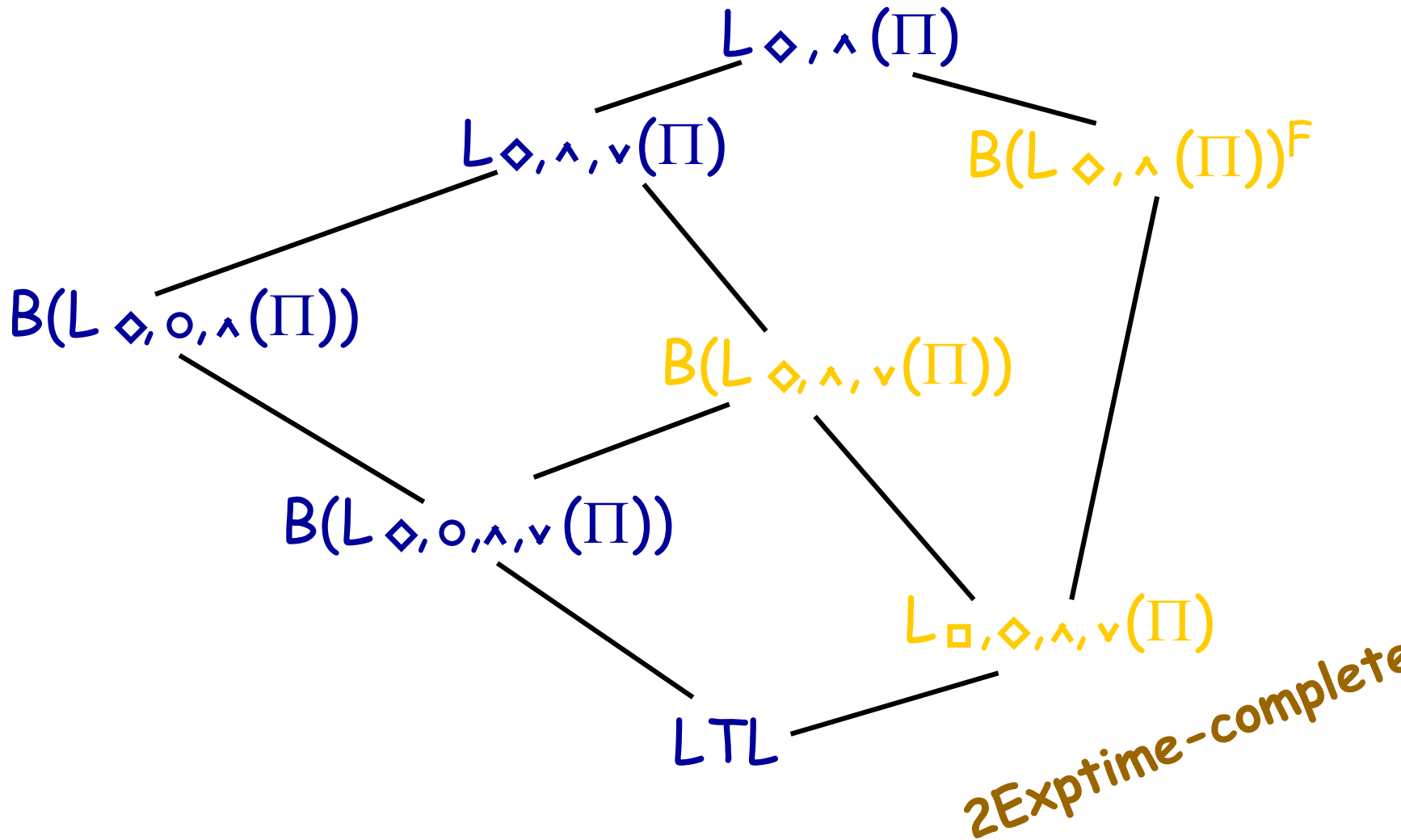
- ◆ adversary selects 4 positions to check that a position can derive from the positions of the previous configuration
- ◆ protagonist loses if these positions do not conform to TM behaviour

□ formulas similar to $\text{Match}(a,i)$

Match(a, i)

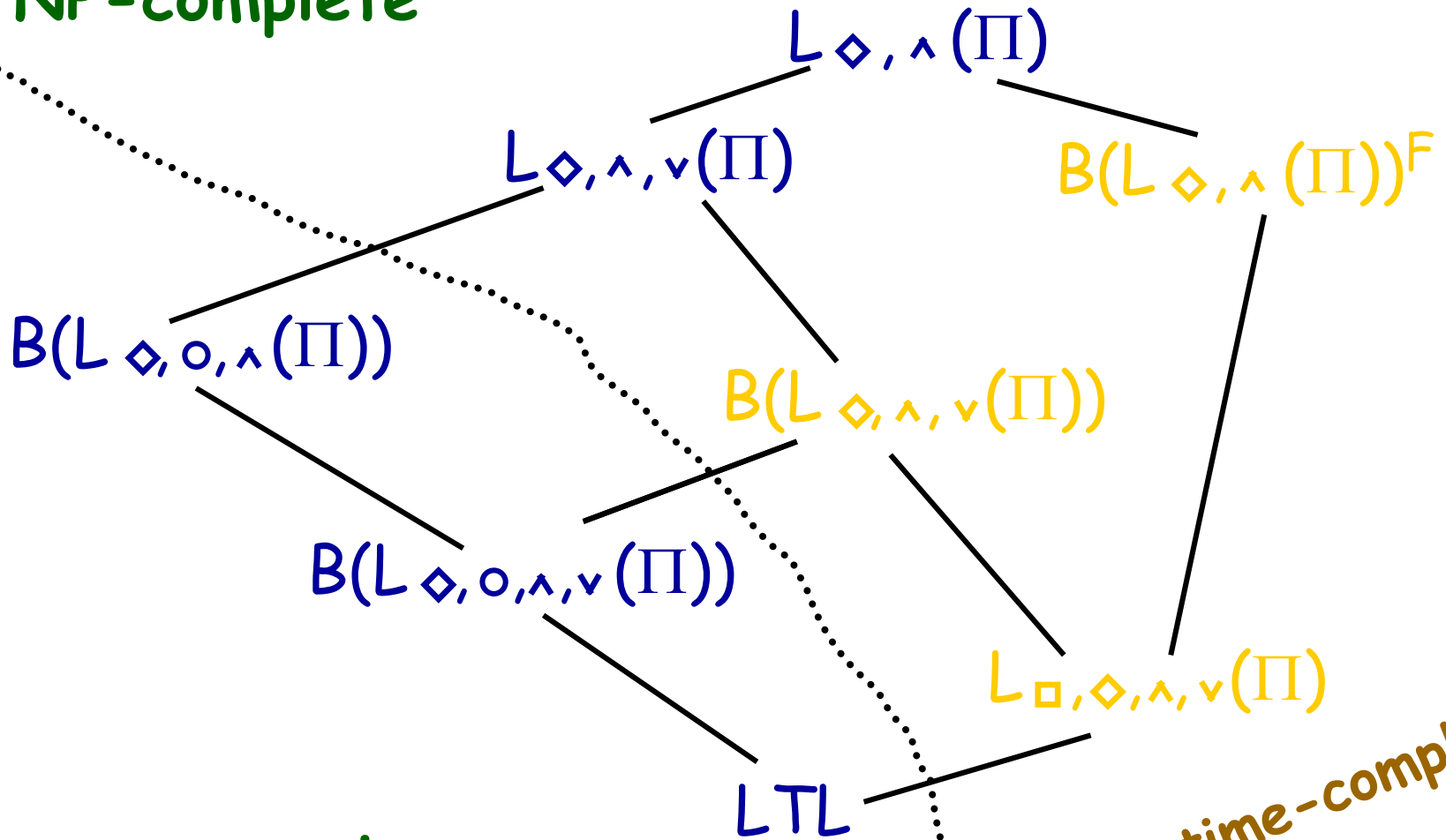
- $Seq(b_m, \dots, b_1) = \diamond(b_m \wedge \diamond(\dots \wedge \diamond b_1) \dots)$
- $Same(p_j, b_j) = (p_j^0 \wedge \neg b_j) \vee (p_j^1 \wedge b_j)$
- $Diff(q_j, b_j) = (q_j^0 \wedge b_j) \vee (q_j^1 \wedge \neg b_j)$
- $Match(a, i) = Seq(Same(p_n, b_n), \dots, Same(p_1, b_1), a, Diff(q_1, b_1), \dots, Diff(q_n, b_n))$
($b_n \dots b_1$ binary encoding of i)

LTL fragments



Complexity: Model-checking

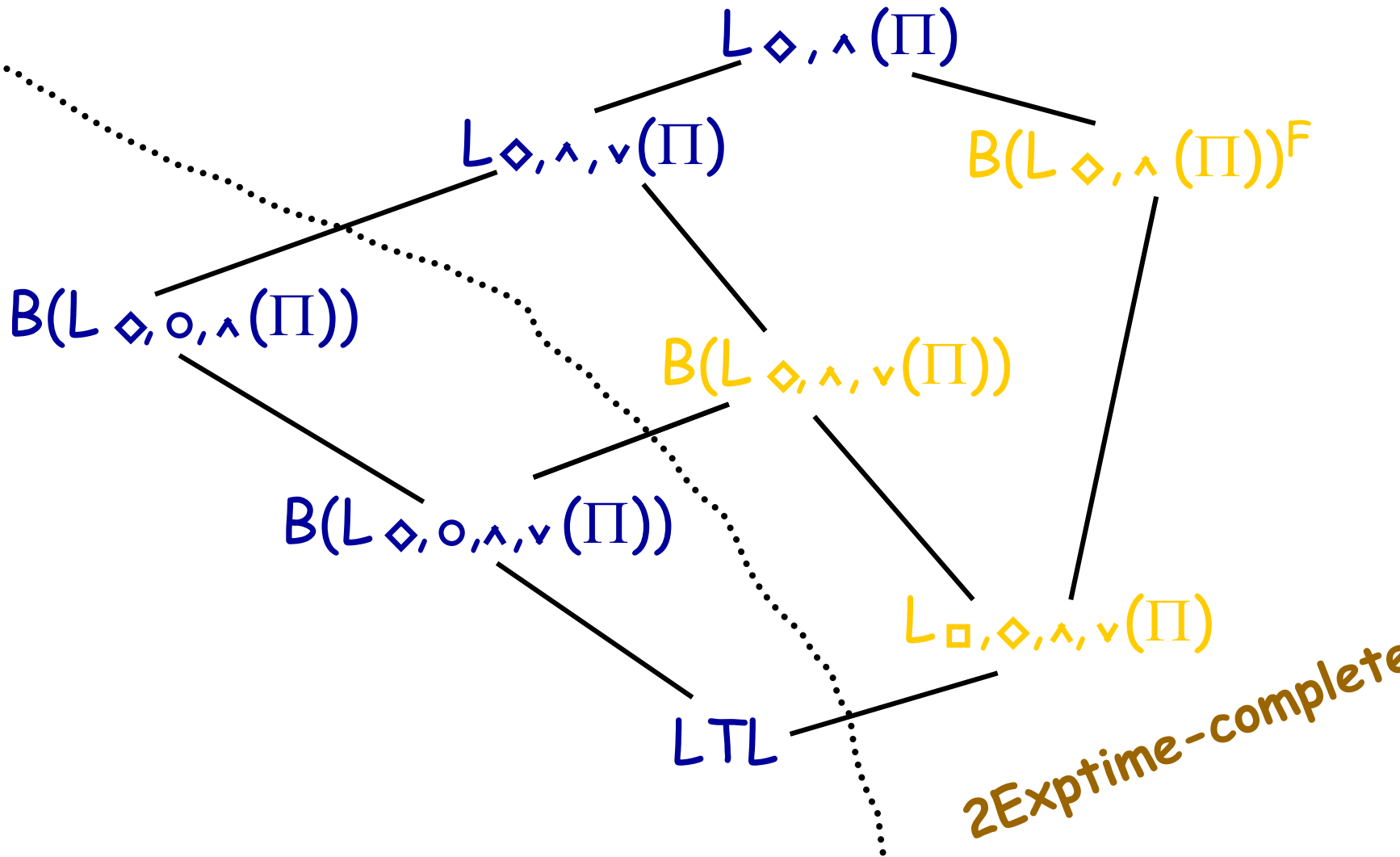
NP-complete



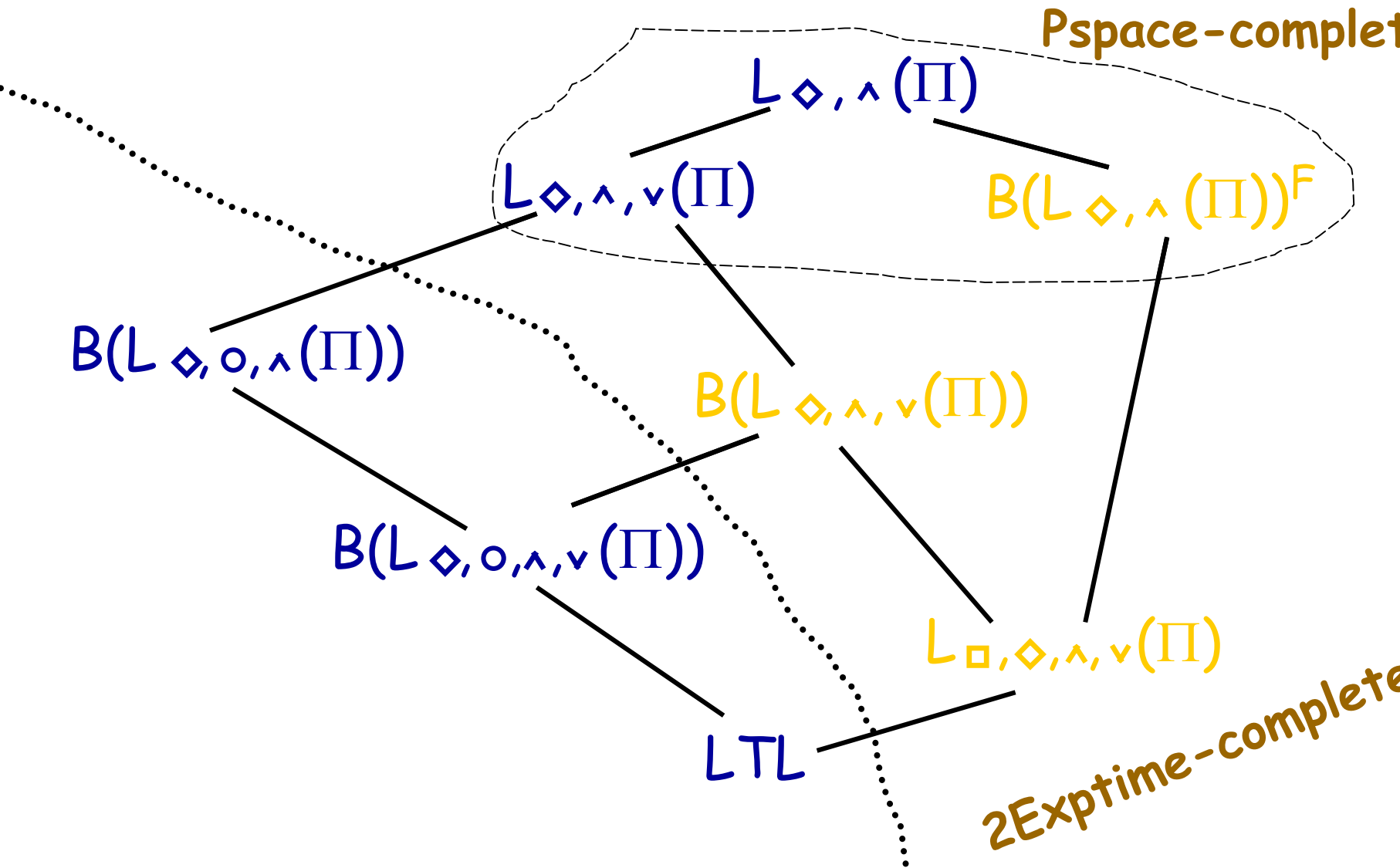
2Exptime-complete

Pspace-complete

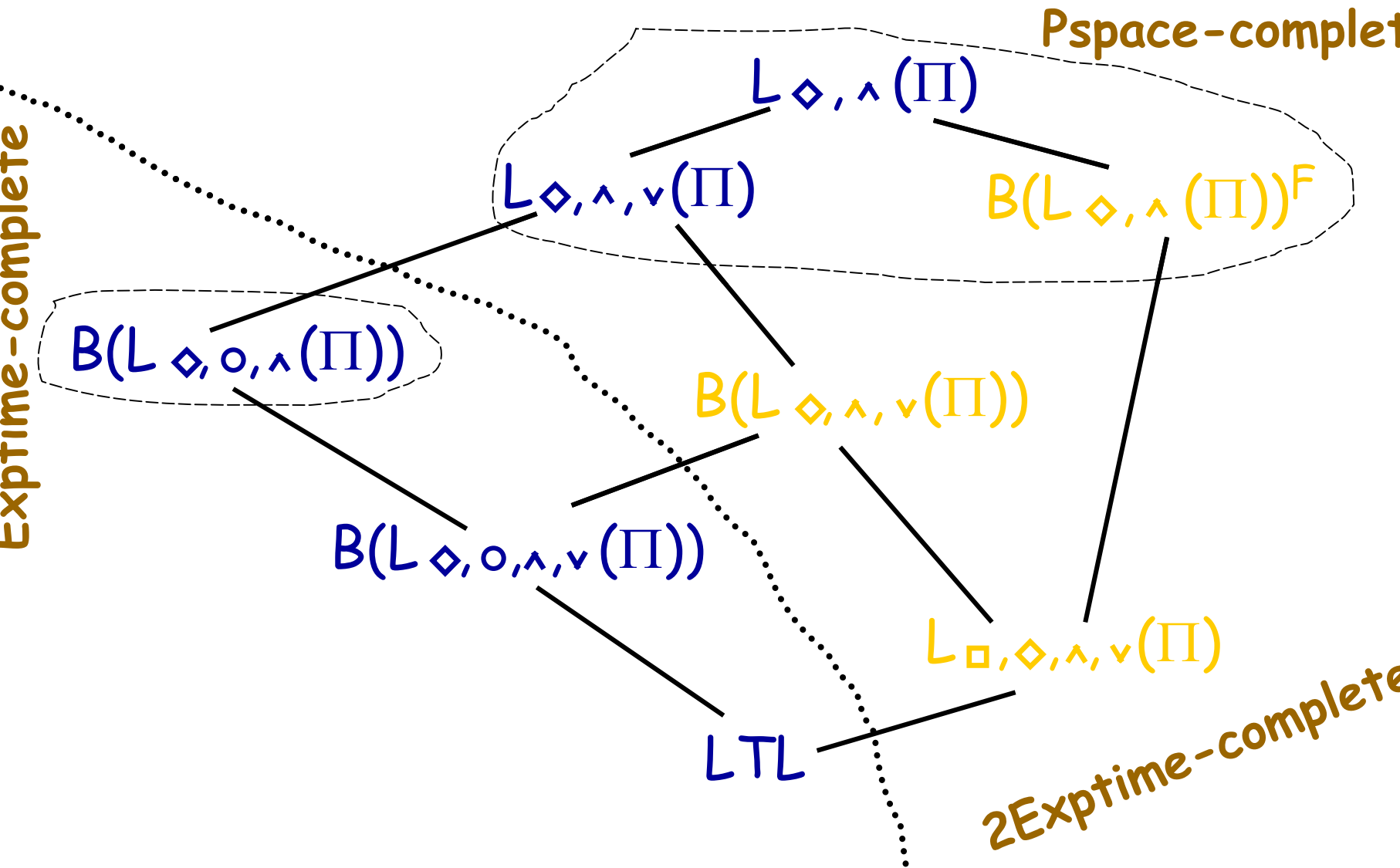
Complexity: Games



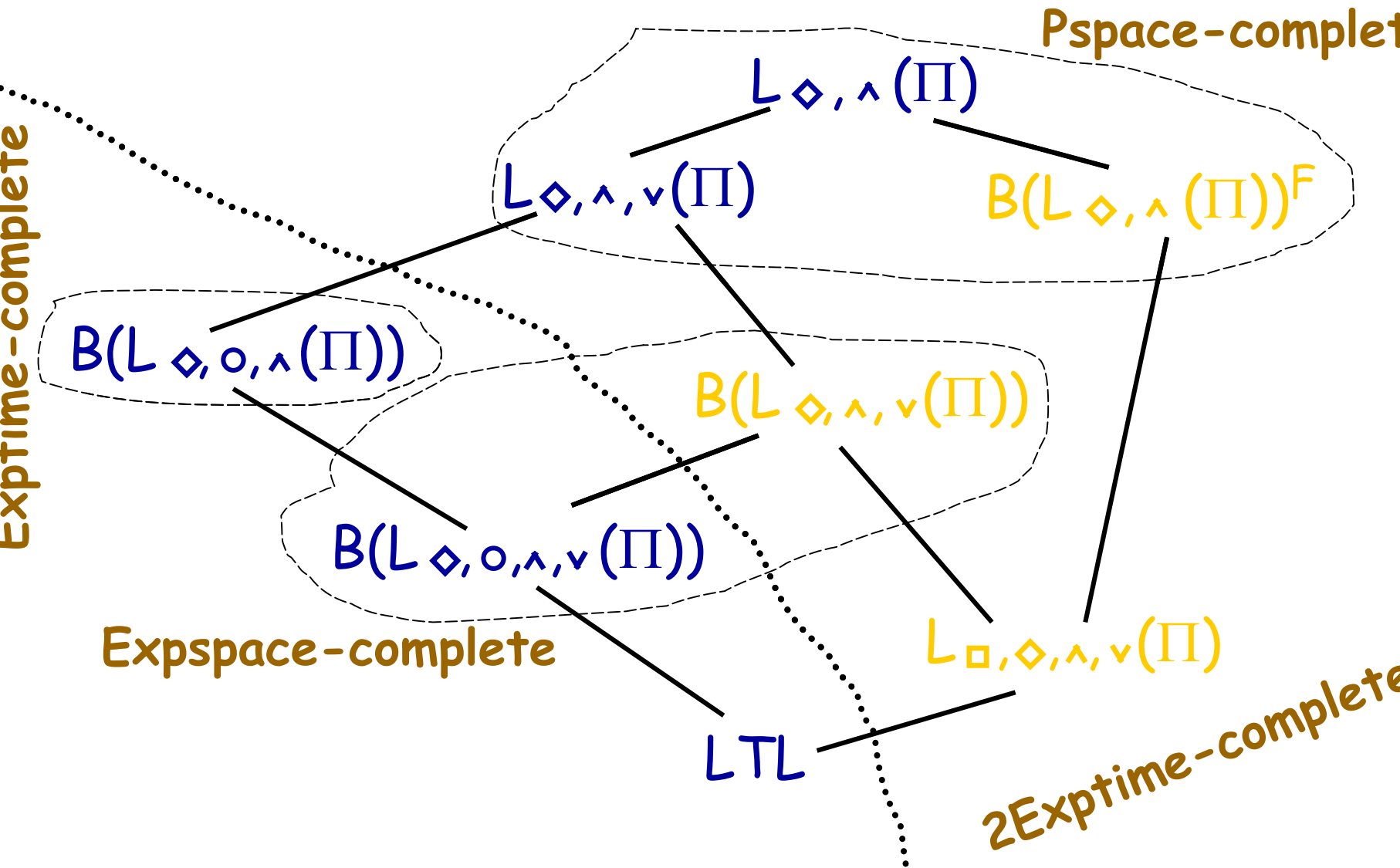
Complexity: Games



Complexity: Games



Complexity: Games



Complexity: Games

