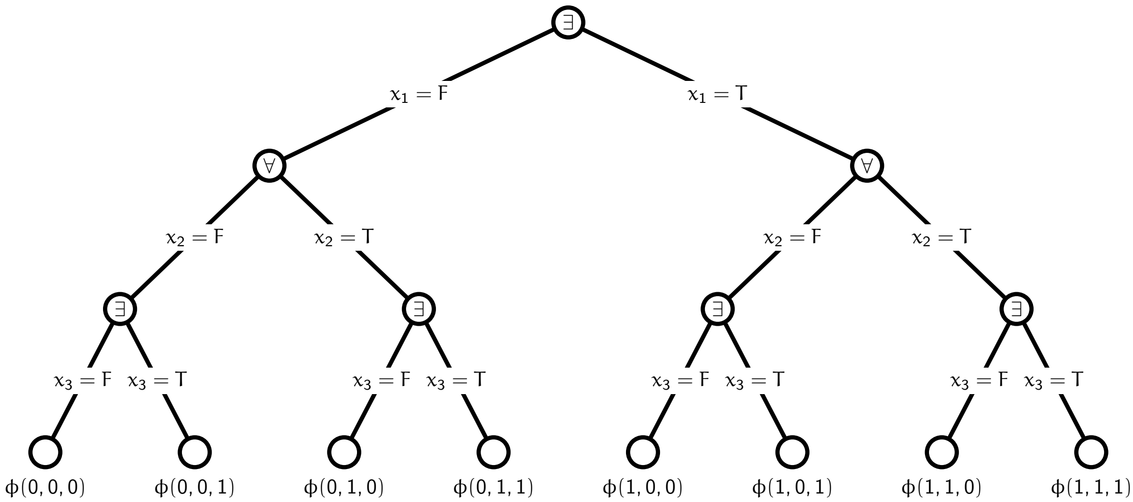
Complexity of the EF-Problem

- It is easy to prove that the problem is in PSPACE
- The difficult part is proving hardness for PSPACE
- The problem is in fact PSPACE-complete
- It is proved by reducing QBF (Quantified Boolean Formula) to the problem of determining whether **II** has a winning strategy
- QBF formulas have the form

$$\exists x_1 \forall x_2 \exists x_3 \cdots Qx_k \, (C_1 \wedge \cdots \wedge C_n)$$

  where each C_j is a disjunction of literals

QBF is in PSPACE



- Exhaustive search of the evaluation tree
- For each node, only one bit of information (true/false)
- $\forall$-nodes are true iff both children are true
- $\exists$-nodes are true iff at least one child is true
- Space proportional to the tree height (recursion depth)

QBF is in PSPACE

On input ϕ:

- If ϕ has no quantifiers, then evaluate ϕ and accept iff it is true.
- If $\phi = \exists x\, \phi$, then recursively evaluate $\phi'[x = 0]$ and $\phi'[x = 1]$ and accept iff either computation accepts.
- If $\phi = \forall x\, \phi'$, then recursively evaluate $\phi'[x = 0]$ and $\phi'[x = 1]$ and accept iff both computations accept.
- Recursion depth = number of variables of ϕ and each level stores values of formula for one variable, so total space used for recursion is linear. Evaluating ϕ at each level also requires linear space, but this can be shared between calls.

The EF-problem is PSPACE-complete

Theorem (Pezzoli)

The EF-problem for finite structures over any fixed signature that contains at least one binary and one ternary relation is PSPACE-complete.

- The proof for hardness goes by reducing QBF to the EF-problem
- Given a QBF formula ϕ of the form

$$\exists x_1 \forall x_2 \cdots \exists x_{2r-1} \forall x_{2r} \, (C_1 \wedge \cdots \wedge C_n),$$

  we build two structures $\mathcal{A}$ and $\mathcal{B}$ over $\Sigma = \{E, H\}$, where E is binary and H is ternary, such that **I** wins $\mathcal{G}_{2r+1}(\mathcal{A}, \mathcal{B})$ iff ϕ is satisfiable

Sketch of the proof

- **I**'s moves correspond to existential quantifiers
- **II**'s moves correspond to universal quantifiers
- Structures $\mathcal{A}$ and $\mathcal{B}$ consist of r blocks
- Each block is made of a certain number of subgraphs, called "gadgets", which are of three types: J, L, and I
- Some elements of the domains are labelled by truth values or pairs of truth values
- Some elements in the last block (block r) are labelled by clauses of ϕ
- A pair of consecutive rounds i, $i+1$ is played within block $\lceil i/2 \rceil$ and corresponds to instantiate a pair of consecutive variables $\exists x_i \forall x_{i+1}$

Sketch of the proof (cont.)

- At round i, **I** assigns the truth value T (resp., F) to variable x_i by choosing an element in block $\lceil i/2 \rceil$ of one of the structures (say, $\mathcal{A}$) "labelled" by T (resp., F)
- **II** is forced to reply by choosing an element "labelled" by a pair of truth values TT or TF (resp., FT or FF) in $\mathcal{B}$, which corresponds to recording **I**'s assignment (the first truth value) and to assign a truth value to variable x_{i+1} (the second truth value)
- At round $i + 1$, **I** chooses an "unlabelled" element in $\mathcal{B}$
- **II** is forced to reply by recording the truth value of x_{i+1} in $\mathcal{A}$ by choosing an element "labelled" the same as the second truth value chosen at round i

Sketch of the proof (cont.)

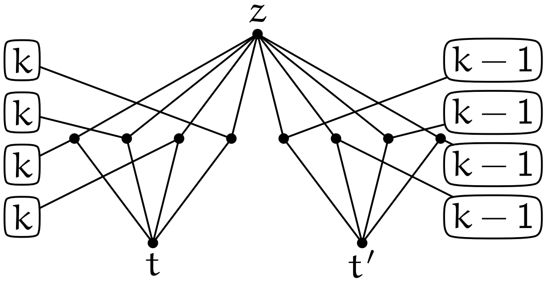
E.g., the pair of rounds may go like this:

round i	round $i+1$	
$s : \mathsf{T}(x_i)$	$d : \mathsf{F}(x_{i+1})$	$\mathcal{A}$
$d : \mathsf{TF}(x_i x_{i+1})$	$s : r$	$\mathcal{B}$

- The "labelling" is encoded by a ternary relation H such that $H(u, v, w)$ holds iff
  - u and v are adjacent in the same block
  - w is in the last block and is labelled by clause C_j
  - Clause C_j is made true by the truth values that label u and/or v

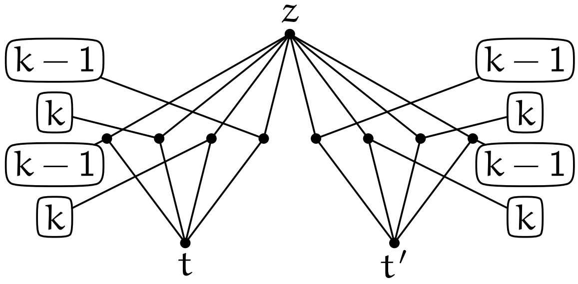
Gadgets J_k, L_k

Circled node are special neighbours



Gadget J_k

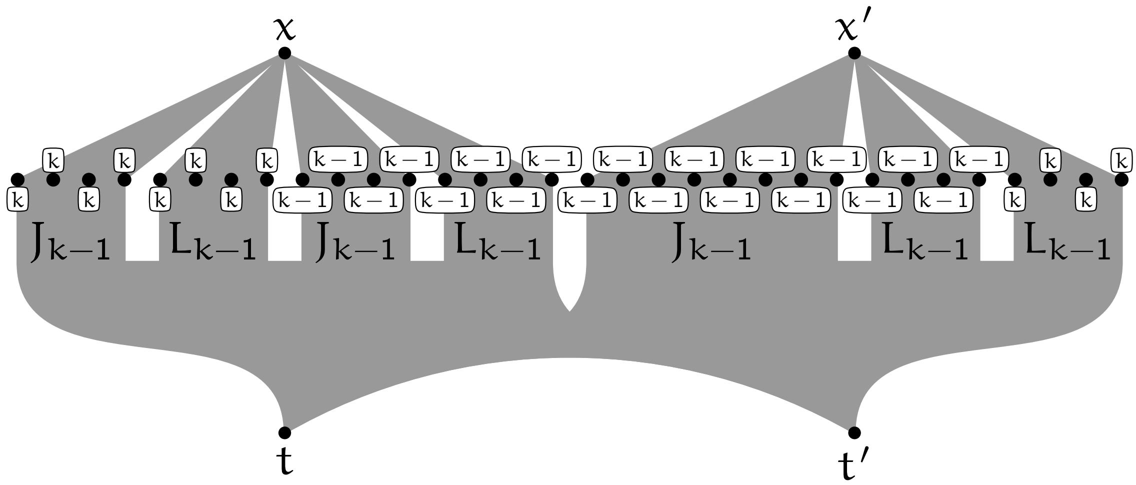
- four nodes in the middle have k special neighbours and target t
- four nodes in the middle have $k-1$ special neighbours and target t'



Gadget L_k

- four nodes in the middle have k special neighbours (two with target t and two with target t')
- four nodes in the middle have $k-1$ special neighbours (two with target t and two with target t')

Gadget I_k



- x and x' are linked to 16 nodes each (*nodes in the middle*)
- Each node in the middle is the source of a gadget J_{k-1} or L_{k-1}
- All gadgets share the same two targets t and t'
- Each node in the middle has either k or $k-1$ special neighbours
- I_k is symmetric if I_k's special neighbours are removed

Forcing pairs

Lemma (Forcing lemma)

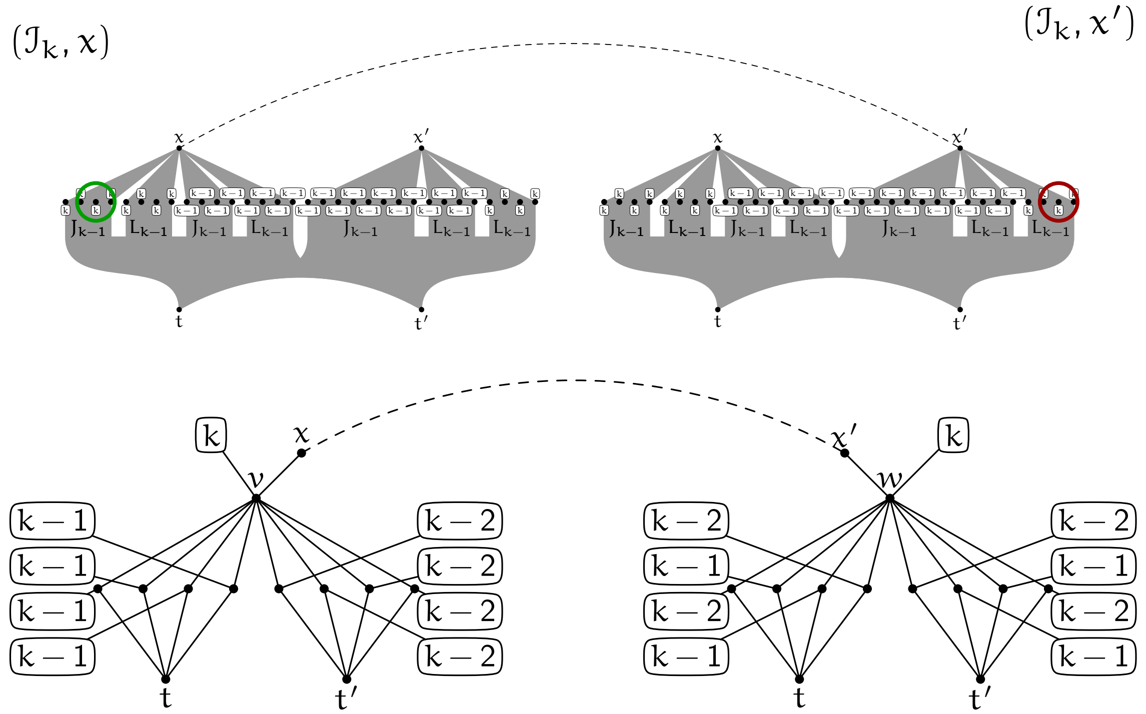
*In the $(k+1)$-moves EF-game on (I_k, x, I_k, x'), **I** can force the pair (t, t'), but **II** has a winning strategy in the k-moves EF-game that allows him to reply t to t and t' to t'.*

- **Notation:** let kxG denote a node adjacent to x, with k special neighbours, and which is the source of a gadget of type G_{k-1}, with $G \in \{J, L\}$
- In the $(k+1)$-moves game **I** starts by playing $v = kxJ$
- **II** must answer with $w = kx'L$
  - otherwise, **I** wins by moving into the special neighbours
- **I** chooses $w(k-1)t'$ in L_{k-1}
- **II** must answer $v(k-1)t$ in J_{k-1}

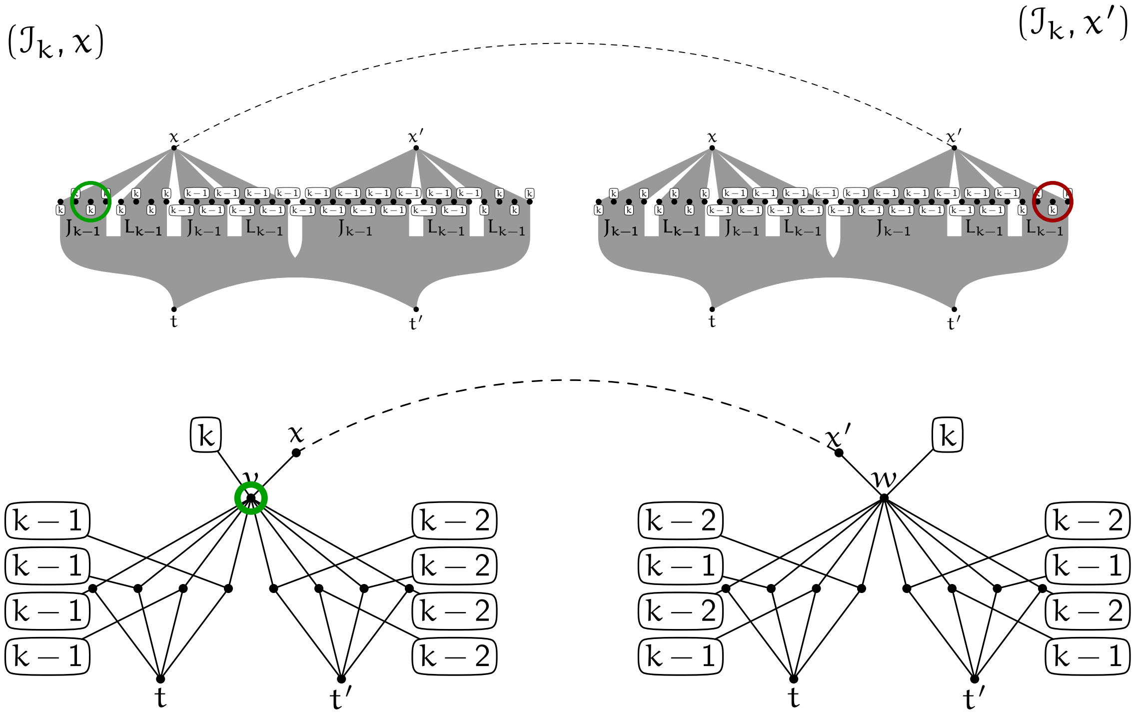
Remark

The above lemma says nothing about who has a winning strategy.

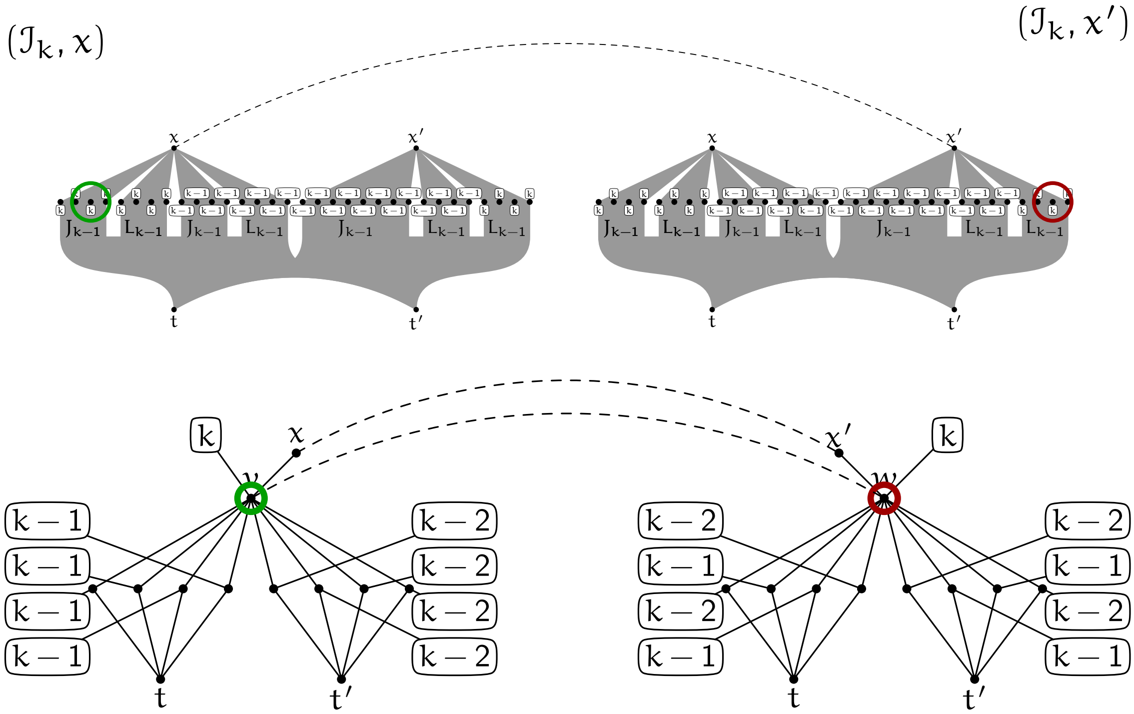
Forcing pairs (cont.)



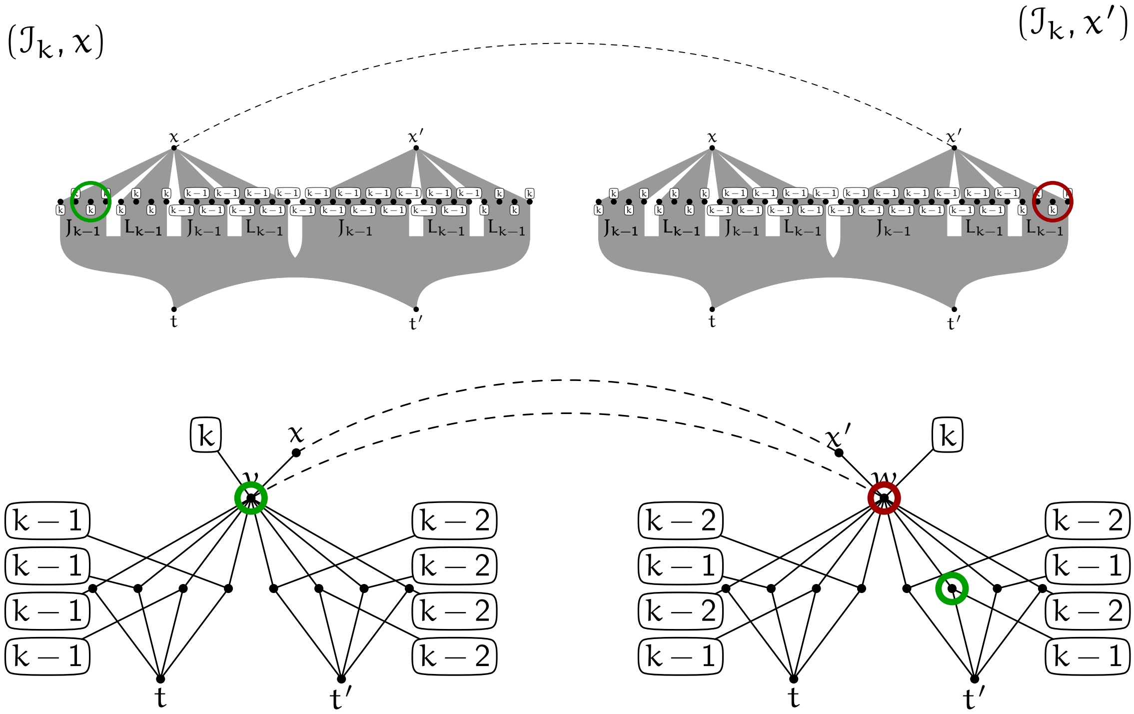
Forcing pairs (cont.)



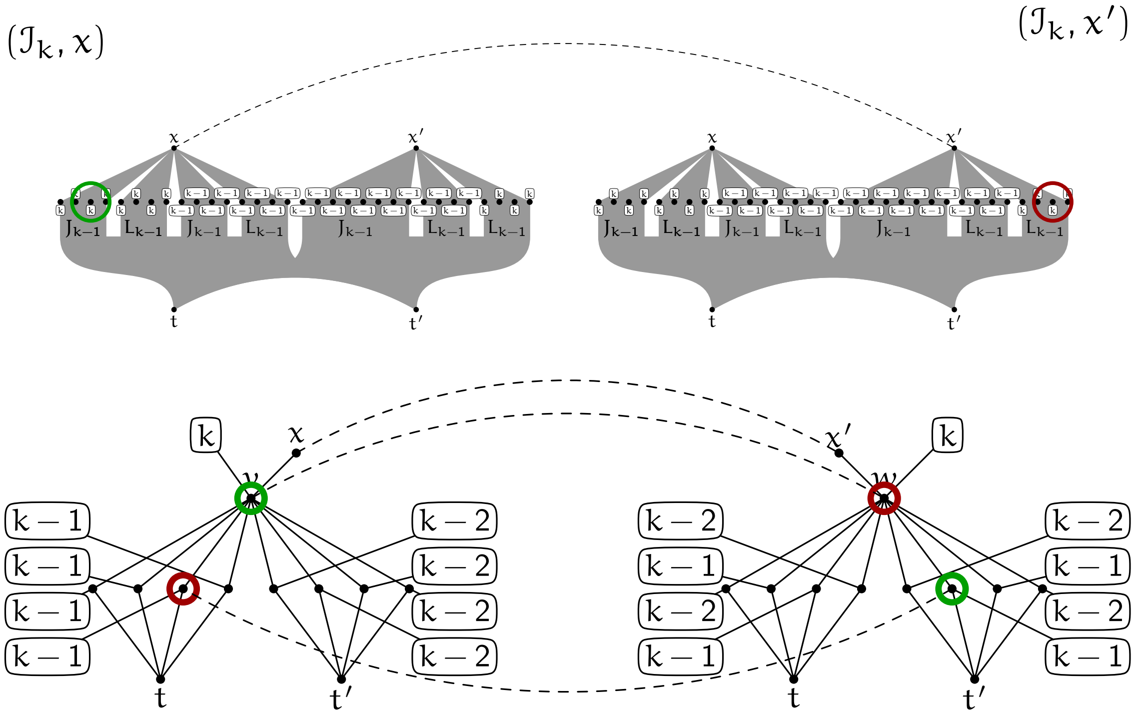
Forcing pairs (cont.)



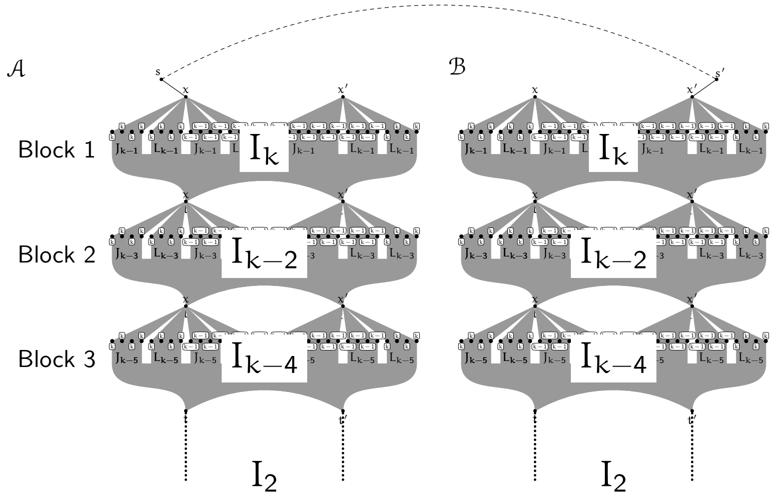
Forcing pairs (cont.)



Forcing pairs (cont.)

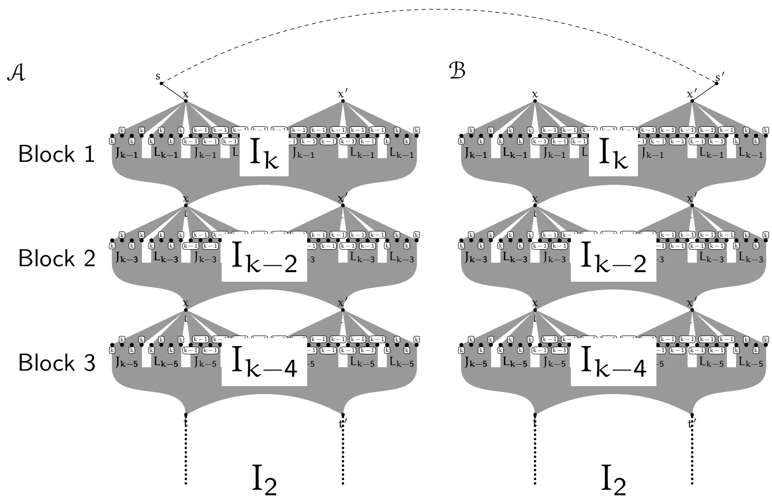


The whole structure



- The game will proceed by choosing two vertices in each block, from top to bottom, according to the strategy of the forcing lemma
- The two vertices in each block are the source of a gadget J_h or L_h and a vertex in the middle in the same gadget
- The pairs of vertices connecting two blocks are never chosen by the players

The whole structure (cont.)

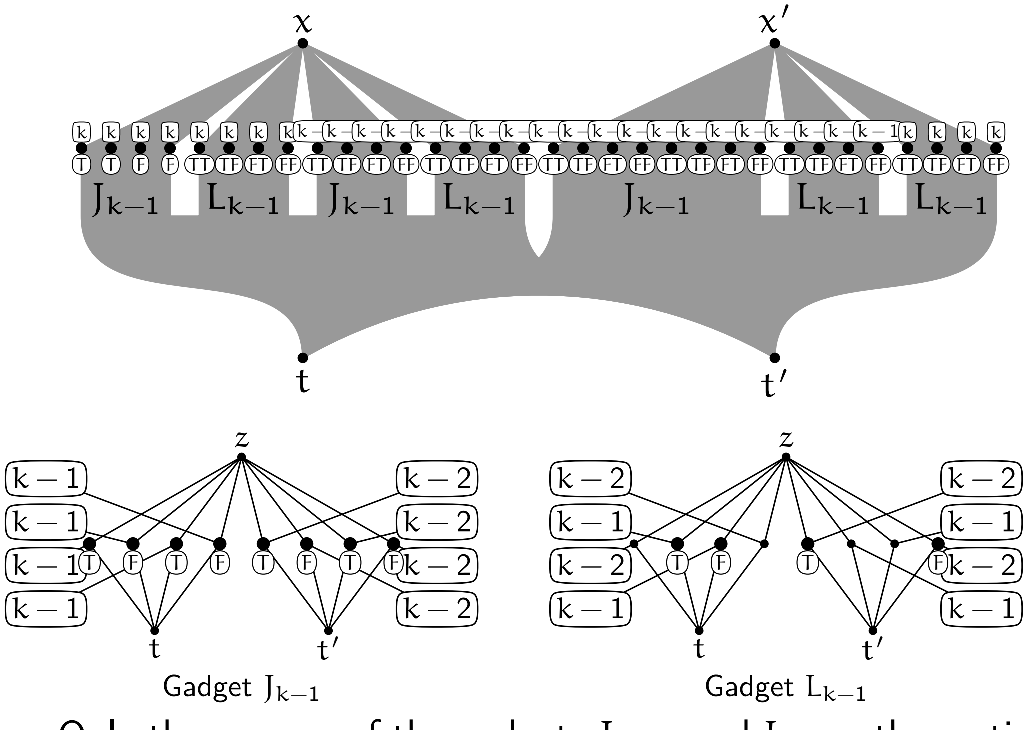


- Up to now, $\mathcal{A}$ and $\mathcal{B}$ are $(2r+1)$-equivalent
- A "meta-labelling" of the vertices is introduced
- The last block of each structure is slightly changed
- The "meta-labelling" induces a ternary relation H
- H relates a winning strategy for **I** with the satisfiability of a formula ϕ

The truth-value labelling

- Same labelling no matter what ϕ is
- Just a convenience for defining H
- There are no unary predicates in the vocabulary
- Of the four vertices kxJ, two are labelled T and the other two F
- For each group of four vertices kxL, $(k-1)xJ$, $(k-1)xL$, $(k-1)x'J$ (two groups of four vertices),$(k-1)x'L$ and $kx'L$, one is labelled TT, one TF, one FT, one FF
- Of the four vertices in the middle of any gadget J_{k-1} with $k-1$ special neighbours, or $k-2$ special neighbours, two are labelled T and two F
- In gadget L_{k-1} the two vertices $(k-1)zt'$ and the two vertices $(k-2)zt$ are not labelled
- Of the two remaining vertices $(k-1)zt$ and the two $(k-2)zt'$, one is labelled T and the other F

The truth-value labelling (cont.)



- Only the sources of the gadgets J_{k-1} and L_{k-1}, the vertices in the middle of J_{k-1} and half of the vertices in the middle of L_{k-1} are labelled by (pairs of) truth values

Labelling vertices by clauses

- The last block is labelled in a way that depends on ϕ
- In the last block, t and t' are replaced by two sets of elements labelled by clauses of ϕ
- t' is replaced by $2r + 1$ vertices labelled C_1, $2r + 1$ vertices labelled C_2, ..., $2r + 1$ vertices labelled C_n
- t is replaced by $2r + 1$ vertices labelled C_1, ..., $2r + 1$ vertices labelled C_n, *plus an unlabelled vertex*
- The new vertices are not mutually adjacent, but they are adjacent to all the vertices previously connected to t' or t, respectively
- The labelling of vertices with (pairs of) truth values and clauses is used to define the ternary relation H

The ternary relation H

Definition (Ternary relation H)

$H(u, v, w)$ holds if, and only iff, u and v are consecutive in the same block $\lceil i/2 \rceil$, w is in the last block, w is labelled by a clause C and one of the following holds:

- u is labelled $a \in \{T, F\}$, v is labelled $b \in \{T, F\}$, or
- u is labelled ab, with $a, b \in \{T, F\}$, v is not labelled, or
- u is labelled ac, v is labelled b, with $a, b, c \in \{T, F\}$,

and assigning a to x_i and b to x_{i+1} makes C true

Lawful strategies

- **I** starts playing in $\mathcal{A}$
- Then, **I** will play in $\mathcal{A}$ at every odd round and in $\mathcal{B}$ at every even round
- Besides, **I** plays on the "left" of $\mathcal{A}$ in odd rounds and on the "right" of $\mathcal{B}$ in even rounds
- At each odd round, **II** is forced to record **I**'s choice in $\mathcal{B}$, i.e., if **I** picks an element labelled T in $\mathcal{A}$ then **II** must reply with TT or TF, but not with FF or FT (otherwise, she is bound to lose in less than $2r + 1$ rounds)
- Similarly, **II** is forced to record its choice in $\mathcal{A}$ at the next round, i.e., if she has chosen TF in $\mathcal{B}$ then she will pick an element labelled by F in $\mathcal{A}$
- If **II** fails to play like that, at some following round **I** may pick an element labelled by a clause C that appears in some triple of H, but **II** would not be able to do so in the opposite structure

What if **II** does not record **I**'s choices?

Example

$$\phi \stackrel{\text{def}}{=} \exists x_1 \forall x_2 \exists x_3 \forall x_4 \left((\bar{x}_3 \vee x_2) \wedge \bar{x}_1 \wedge (x_1 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee x_4) \right)$$

Suppose that during a game the following labelling is determined:

round 1	round 2	round 3	round 4	
$s : F(x_1)$	$d : F(x_2)$	$s : F(x_3)$	$d : F(x_4)$	$\mathcal{A}$
$d : FF(x_1 x_2)$	$s : r$	$d : TF(x_3 x_4)$	$s : r'$	$\mathcal{B}$

- **II** does not record the move made by **I** at round 3
- At round 5, **I** jumps to an element labelled by clause $\bar{x}_3 \vee x_4$ in $\mathcal{A}$, which determines a triple in H
- **II**, however, cannot find a corresponding element in $\mathcal{B}$ (no clause is satisfied when x_4 is false, but x_3 is true)

What if **II** does not record **I**'s choices?

Example

$$\phi \overset{\text{def}}{=} \exists x_1 \forall x_2 \exists x_3 \forall x_4 \left((\bar{x}_3 \vee x_2) \wedge \bar{x}_1 \wedge (x_2 \vee x_3) \wedge (x_4 \vee \bar{x}_4) \right)$$

Suppose that during a game the following labelling is determined:

round 1	round 2	
$s : F(x_1)$	$d : F(x_2)$	$\mathcal{A}$
$d : TF(x_1 x_2)$	$s : r$	$\mathcal{B}$

- **II** does not record the move made by **I** at round 1
- At round 3, **I** may choose an element labelled by $\bar{x}_1$ in $\mathcal{A}$, which determines a triple in H
- **II**, however, cannot find a corresponding element in $\mathcal{B}$ (no clause is satisfied when x_1 is true and x_2 is false)

How **I** wins if ϕ is satisfiable

- Suppose that ϕ is satisfiable
- Assuming that **I** follows a lawful strategy and **II** correctly records the truth values, the choices of the players will determine the same truth assignment for the variables of ϕ, both in $\mathcal{A}$ and $\mathcal{B}$
- At the last round, **I** chooses the only vertex w not labelled by any clause at the bottom of $\mathcal{A}$
- But, by the forcing lemma, **II** is bound to choose a vertex w' at the bottom of $\mathcal{B}$ labelled by some clause C, or to choose a vertex not adjacent to the choice **I** has made in $\mathcal{B}$ in the previous round
- In the latter case, **II** loses immediately
- In the former case, since **I** has played in such a way to build a satisfying assignment and **II** has recorded such assignment in $\mathcal{B}$, the last choice by **II** will determine a triple (u', v', w') of $H^{\mathcal{B}}$, for some previously chosen vertices u' and v'
- But $(u, v, w) \notin H^{\mathcal{A}}$ for corresponding u, v in $\mathcal{A}$

Complexity results for pebble games

- Pebble games are a variant of EF-games in which each player has a limited number of pebbles and re-uses them
- They correspond to formulas with a bounded number of variables

Theorem

Given a positive integer k *and structures* $\mathcal{A}$ *and* $\mathcal{B}$ *the problem of determining whether* ***II*** *has a winning strategy in the existential* k*-pebble game on* $\mathcal{A}$ *and* $\mathcal{B}$ *is EXPTIME-complete.*

Corollary

All algorithms for determining whether k*-strong consistency can be established are inherently exponential.*

P. G. Kolaitis, J. Panttaja
On the Complexity of Existential Pebble Games
CSL 2003

The proof of EXPTIME-completeness is not that easy. . .

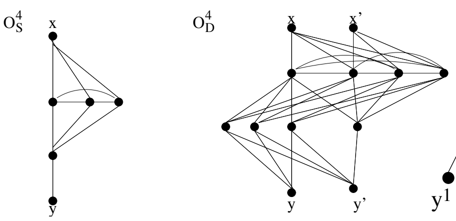
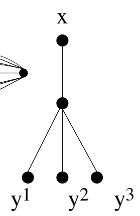


Fig. 3. Single Input One-Way Switch O^4



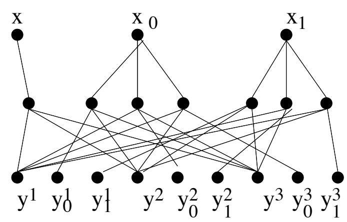
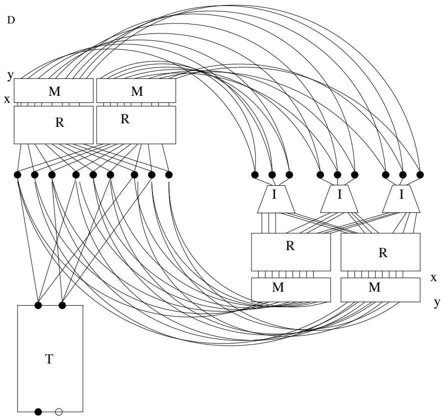
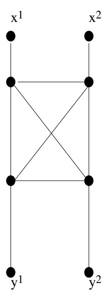


Fig. 6. I^3 Gadget





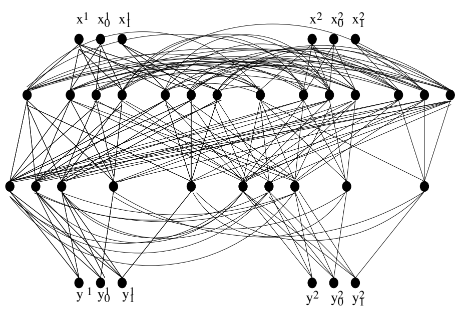


Fig. 7. A subgraph of M^4

Fig. 10. This is component decomposition of the Duplicator's graph for the reduction

Conclusions

- EF-games not explored much algorithmically
  - What is the complexity of the EF-problem for (labelled) arbitrary trees?
  - What is complexity of the EF-problem for signature containing only a binary relations E (i.e., graphs)?
  - The question for the complexity of first-order equivalence for finite structures, that is, isomorphism, is open (strictly related to the graph isomorphism problem)
- Simpler proofs?
- May notions from Combinatorial Game Theory help?
  - Berlekamp's et al. *Winning Ways*