CSPs and Datalog

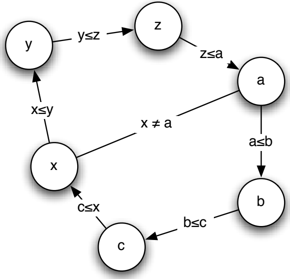
Manuel Bodirsky

CNRS/LIX, École Polytechnique

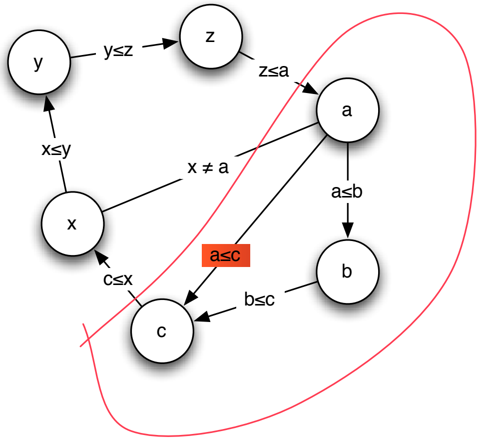
December 2013

Example

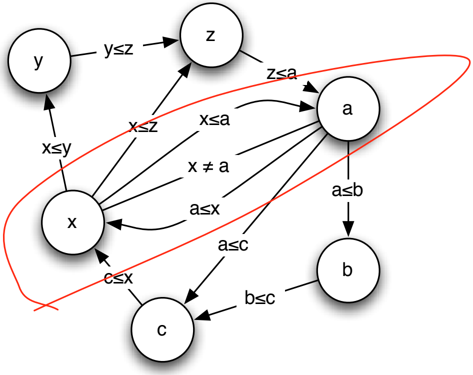
An instance of CSP$(\mathbb{Q}; \leq, \neq)$:



Example



Example



Datalog

Datalog can be seen as

- Prolog without function symbols
- conjunctive queries + recursion

Datalog

Datalog can be seen as

- Prolog without function symbols
- conjunctive queries + recursion

Introduced in the context of constraint satisfaction by Feder, Vardi, and Kolaitis

Datalog

Datalog can be seen as

- Prolog without function symbols
- conjunctive queries + recursion

Introduced in the context of constraint satisfaction by Feder, Vardi, and Kolaitis

Can be used to formulate local consistency methods studied in Artificial Intelligence (AI) since the late 70s

Datalog

Datalog can be seen as

- Prolog without function symbols
- conjunctive queries + recursion

Introduced in the context of constraint satisfaction by Feder, Vardi, and Kolaitis

Can be used to formulate local consistency methods studied in Artificial Intelligence (AI) since the late 70s

Main algorithmic technique studied in more applied AI literature

(l, k)-Datalog

Example of a Datalog program

$$
\begin{aligned}
tc(x, y) &:- \quad x < y \\
tc(x, y) &:- \quad tc(x, z), z < y \\
false() &:- \quad tc(x, x)
\end{aligned}
$$

Relation symbols that appear in rule heads (*tc*, *false*): IDBs

(l, k)-Datalog

Example of a Datalog program

$$\begin{aligned} tc(x,y) &:- \quad x < y \\ tc(x,y) &:- \quad tc(x,z), z < y \\ false() &:- \quad tc(x,x) \end{aligned}$$

Relation symbols that appear in rule heads (*tc*, *false*): IDBs
All other relation symbols ($<$): EDBs

(l, k)-Datalog

Example of a Datalog program

$$
\begin{aligned}
tc(x, y) &:- \quad x < y \\
tc(x, y) &:- \quad tc(x, z), z < y \\
false() &:- \quad tc(x, x)
\end{aligned}
$$

Relation symbols that appear in rule heads (*tc*, *false*): IDBs
All other relation symbols ($<$): EDBs
Is a $(2, 3)$-Datalog program (maximal IDB arity is 2, number of variables is 3)

(l, k)-Datalog

Example of a Datalog program

$$\begin{aligned} tc(x, y) &:- \quad x < y \\ tc(x, y) &:- \quad tc(x, z), z < y \\ false() &:- \quad tc(x, x) \end{aligned}$$

Relation symbols that appear in rule heads (*tc*, *false*): IDBs
All other relation symbols ($<$): EDBs
Is a $(2, 3)$-Datalog program (maximal IDB arity is 2, number of variables is 3)
false is special 0-ary IDB

(l, k)-Datalog

Example of a Datalog program

$$\begin{aligned} tc(x,y) &:- \quad x < y \\ tc(x,y) &:- \quad tc(x,z), z < y \\ false() &:- \quad tc(x,x) \end{aligned}$$

Relation symbols that appear in rule heads (*tc*, *false*): IDBs
All other relation symbols ($<$): EDBs
Is a $(2,3)$-Datalog program (maximal IDB arity is 2, number of variables is 3)
false is special 0-ary IDB
Program Π solves CSP(Γ):

Π derives *false* on an instance A if and only if A is unsatisfiable.

(l, k)-Datalog

Example of a Datalog program

$$\begin{aligned} tc(x,y) &:- \quad x < y \\ tc(x,y) &:- \quad tc(x,z), z < y \\ false() &:- \quad tc(x,x) \end{aligned}$$

Relation symbols that appear in rule heads (*tc*, *false*): IDBs
All other relation symbols ($<$): EDBs
Is a $(2,3)$-Datalog program (maximal IDB arity is 2, number of variables is 3)
false is special 0-ary IDB
Program Π solves CSP(Γ):

Π derives *false* on an instance A if and only if A is unsatisfiable.

Example: CSP$(\mathbb{Q}; <)$ can be solved by $(2,3)$-Datalog program

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

- EDB: Single binary relation symbol E

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

- EDB: Single binary relation symbol E
- IDBs: one symbol for each unary relation with a primitive positive definition in H

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

- EDB: Single binary relation symbol E
- IDBs: one symbol for each unary relation with a primitive positive definition in H
- *false* corresponds to the empty unary relation

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

- EDB: Single binary relation symbol E
- IDBs: one symbol for each unary relation with a primitive positive definition in H
- *false* corresponds to the empty unary relation
- Have rule $R(x) \mathrel{:-} E(x,y) \wedge S(y)$
  iff $H \models \forall x, y \; (E(x,y) \wedge S(y)) \Rightarrow R(x)$.

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

- EDB: Single binary relation symbol E
- IDBs: one symbol for each unary relation with a primitive positive definition in H
- *false* corresponds to the empty unary relation
- Have rule $R(x) :- E(x, y) \wedge S(y)$
  iff $H \models \forall x, y \ (E(x, y) \wedge S(y)) \Rightarrow R(x)$.
- Have rule $R(y) :- S(x) \wedge E(x, y)$
  iff $H \models \forall x, y \ (S(x) \wedge E(x, y)) \Rightarrow R(y)$.

Arc-Consistency as Datalog Program

The Arc-Consistency procedure for CSP(H) can be formulated in Datalog:

- EDB: Single binary relation symbol E
- IDBs: one symbol for each unary relation with a primitive positive definition in H
- *false* corresponds to the empty unary relation
- Have rule $R(x)$:– $E(x, y) \wedge S(y)$
  iff $H \models \forall x, y \ (E(x, y) \wedge S(y)) \Rightarrow R(x)$.
- Have rule $R(y)$:– $S(x) \wedge E(x, y)$
  iff $H \models \forall x, y \ (S(x) \wedge E(x, y)) \Rightarrow R(y)$.

Advantage of this perspective: can also be applied to infinite Γ
However, need: Γ has only finitely many binary primitive positive definable unary relations.

Completeness

Every Datalog program can be evaluated on an input structure A in polynomial time in the size of A.

Definition

Say that Datalog solves $\text{CSP}(\Gamma)$ if there exists a Datalog program Π such that for all finite A:
Π derives *false* list if and only if there is no homomorphism from A to Γ.

Facts:

- Datalog solves $\text{CSP}(P_k)$, $\text{CSP}(\vec{P}_k)$, $\text{CSP}(T_k)$

Completeness

Every Datalog program can be evaluated on an input structure A in polynomial time in the size of A.

Definition

Say that Datalog solves $\mathrm{CSP}(\Gamma)$ if there exists a Datalog program Π such that for all finite A:
Π derives *false* list if and only if there is no homomorphism from A to Γ.

Facts:

- Datalog solves $\mathrm{CSP}(P_k)$, $\mathrm{CSP}(\vec{P}_k)$, $\mathrm{CSP}(T_k)$
- Datalog solves $\mathrm{CSP}(\vec{C}_k)$, $\mathrm{CSP}(\mathbb{Q}; <)$

Completeness

Every Datalog program can be evaluated on an input structure A in polynomial time in the size of A.

Definition

Say that Datalog solves $\mathrm{CSP}(\Gamma)$ if there exists a Datalog program Π such that for all finite A:
Π derives *false* list if and only if there is no homomorphism from A to Γ.

Facts:

- Datalog solves $\mathrm{CSP}(P_k)$, $\mathrm{CSP}(\vec{P}_k)$, $\mathrm{CSP}(T_k)$
- Datalog solves $\mathrm{CSP}(\vec{C}_k)$, $\mathrm{CSP}(\mathbb{Q};<)$
- Datalog does not solve $\mathrm{CSP}(K_3)$

Completeness

Every Datalog program can be evaluated on an input structure A in polynomial time in the size of A.

Definition

Say that Datalog solves $\mathrm{CSP}(\Gamma)$ if there exists a Datalog program Π such that for all finite A:
Π derives *false* list if and only if there is no homomorphism from A to Γ.

Facts:

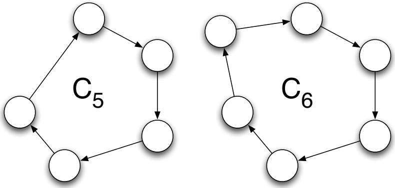
- Datalog solves $\mathrm{CSP}(P_k)$, $\mathrm{CSP}(\vec{P}_k)$, $\mathrm{CSP}(T_k)$
- Datalog solves $\mathrm{CSP}(\vec{C}_k)$, $\mathrm{CSP}(\mathbb{Q};<)$
- Datalog does not solve $\mathrm{CSP}(K_3)$

Question: Which CSPs can be solved by Datalog?

Exercise

Write a Datalog program that solves graph 2-colorability, $\mathrm{CSP}(K_2)$.

The existential pebble game

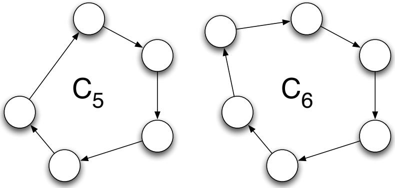


Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator

The existential pebble game



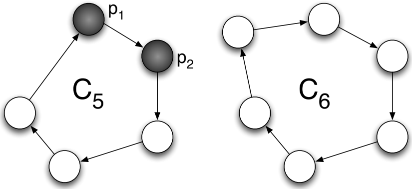
Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator

Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively

The existential pebble game



Let A, B be relational structures.

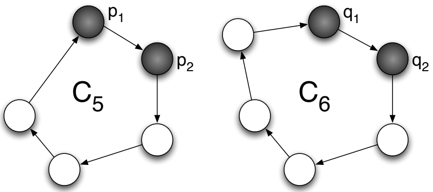
The existential (l, k)-pebble game

Two players: Spoiler and Duplicator

Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively

Spoiler places his pebbles on A,

The existential pebble game

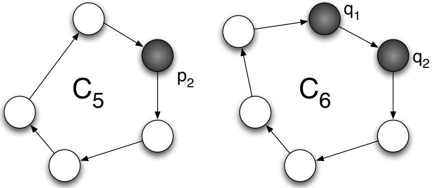


Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator
Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively
Spoiler places his pebbles on A, Duplicator replies on B

The existential pebble game

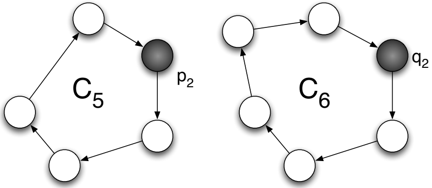


Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator
Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively
Spoiler places his pebbles on A, Duplicator replies on B
Spoiler removes all but at most l pebbles

The existential pebble game

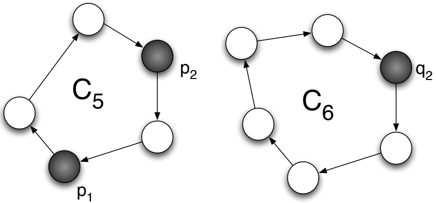


Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator
Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively
Spoiler places his pebbles on A, Duplicator replies on B
Spoiler removes all but at most l pebbles
Duplicator removes corresponding pebbles

The existential pebble game

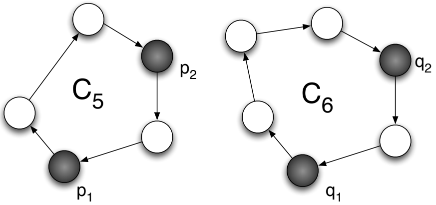


Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator
Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively
Spoiler places his pebbles on A, Duplicator replies on B
Spoiler removes all but at most l pebbles
Duplicator removes corresponding pebbles
Repeat

The existential pebble game

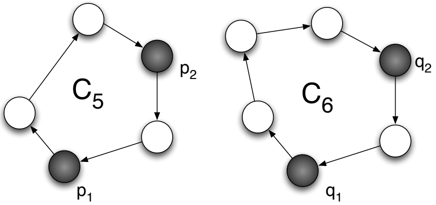


Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator
Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively
Spoiler places his pebbles on A, Duplicator replies on B
Spoiler removes all but at most l pebbles
Duplicator removes corresponding pebbles
Repeat

The existential pebble game



Let A, B be relational structures.

The existential (l, k)-pebble game

Two players: Spoiler and Duplicator
Each has k pebbles $p_1, \ldots, p_k$ and $q_1, \ldots, q_k$, respectively
Spoiler places his pebbles on A, Duplicator replies on B
Spoiler removes all but at most l pebbles
Duplicator removes corresponding pebbles
Repeat

Spoiler wins if eventually $p_i \mapsto q_i$ is not a partial homomorphism from A to B, otherwise Duplicator wins.

The existential pebble game: observations

Let Γ be an arbitrary relational structure.

- Let A be a satisfiable instance of $\mathrm{CSP}(\Gamma)$. Then Duplicator wins the existential k-pebble game on A, Γ.

The existential pebble game: observations

Let Γ be an arbitrary relational structure.

- Let A be a satisfiable instance of $\mathrm{CSP}(\Gamma)$. Then Duplicator wins the existential k-pebble game on A, Γ.
- Suppose that Duplicator has a winning strategy: that is, no matter how Spoiler plays, Duplicator can always play such that she wins.

The existential pebble game: observations

Let Γ be an arbitrary relational structure.

- Let A be a satisfiable instance of CSP(Γ). Then Duplicator wins the existential k-pebble game on A, Γ.
- Suppose that Duplicator has a winning strategy: that is, no matter how Spoiler plays, Duplicator can always play such that she wins.
  Then Duplicator also has a memoryless winning strategy: she can win in such a way that her moves only depend on the current positions of the pebbles, and not on the history of the game.

The Existential Pebble Game and Datalog

Let Γ be a relational structure with finitely many inequivalent primitive positive definable relations of arity k, for all $k \geq 1$.

The Existential Pebble Game and Datalog

Let Γ be a relational structure with finitely many inequivalent primitive positive definable relations of arity k, for all $k \geq 1$.

Theorem 1 (Feder, Vardi'93).

CSP(Γ) cannot be solved by (l, k)-Datalog if and only if there exists an unsatisfiable instance A of CSP(Γ) such that Duplicator wins the existential (l, k)-pebble game on A, Γ.

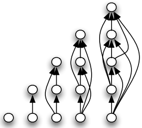
The Existential Pebble Game and Datalog

Let Γ be a relational structure with finitely many inequivalent primitive positive definable relations of arity k, for all $k \geq 1$.

Theorem 1 (Feder,Vardi'93).

CSP(Γ) cannot be solved by (l,k)-Datalog if and only if there exists an unsatisfiable instance A of CSP(Γ) such that Duplicator wins the existential (l,k)-pebble game on A,Γ.

Theorem fails for general infinite Γ:



Let T_∞ be $\dot{\bigcup}_{n\geq 1}^{\infty} T_n$.
(CSP(T_∞) is the same as CSP$(\mathbb{Q};<)$)

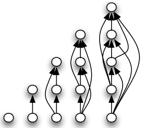
The Existential Pebble Game and Datalog

Let Γ be a relational structure with finitely many inequivalent primitive positive definable relations of arity k, for all $k \geq 1$.

Theorem 1 (Feder, Vardi'93).

CSP(Γ) cannot be solved by (l, k)-Datalog if and only if there exists an unsatisfiable instance A of CSP(Γ) such that Duplicator wins the existential (l, k)-pebble game on A, Γ.

Theorem fails for general infinite Γ:



Let T_∞ be $\dot{\bigcup}_{n\geq 1}^{\infty} T_n$.
(CSP(T_∞) is the same as CSP$(\mathbb{Q}; <)$)

Thus: CSP(T_∞) cannot be solved by $(1, 2)$-Datalog program

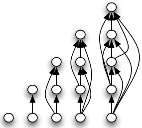
The Existential Pebble Game and Datalog

Let Γ be a relational structure with finitely many inequivalent primitive positive definable relations of arity k, for all $k \geq 1$.

Theorem 1 (Feder,Vardi'93).

CSP(Γ) cannot be solved by (l, k)-Datalog if and only if there exists an unsatisfiable instance A of CSP(Γ) such that Duplicator wins the existential (l, k)-pebble game on A, Γ.

Theorem fails for general infinite Γ:



Let T_∞ be $\dot{\bigcup}_{n \geq 1}^{\infty} T_n$.
(CSP(T_∞) is the same as CSP$(\mathbb{Q}; <)$)

Thus: CSP(T_∞) cannot be solved by $(1, 2)$-Datalog program
But: Duplicator looses the $(1, 2)$-pebble game on (G, T_∞) for all graphs G with a directed cycle.

Application: Datalog Inexpressibility

Application: Datalog Inexpressibility

Theorem 2.

CSP$(\mathbb{Q}; x > y \vee x > z)$ cannot be solved by Datalog.

Application: Datalog Inexpressibility

Theorem 2.

CSP$(\mathbb{Q}; x > y \vee x > z)$ cannot be solved by Datalog.

Remarks

- problem can be solved in linear time (later during lecture)

Application: Datalog Inexpressibility

Theorem 2.

CSP$(\mathbb{Q}; x > y \vee x > z)$ cannot be solved by Datalog.

Remarks

- problem can be solved in linear time (later during lecture)
- relation $x > y \vee x > z$ $(\equiv x > \min(y,z))$ is preserved by min

Application: Datalog Inexpressibility

Theorem 2.

CSP$(\mathbb{Q}; x > y \vee x > z)$ cannot be solved by Datalog.

Remarks

- problem can be solved in linear time (later during lecture)
- relation $x > y \vee x > z$ $(\equiv x > \min(y,z))$ is preserved by min
- Suppose every variable x in an instance Φ of this CSP appears in a constraint $x > y \vee x > z$, for some variables y, z. Then Φ is unsatisfiable.

Application: Datalog Inexpressibility

Theorem 2.

CSP$(\mathbb{Q}; x > y \vee x > z)$ cannot be solved by Datalog.

Remarks

- problem can be solved in linear time (later during lecture)
- relation $x > y \vee x > z$ ($\equiv x > \min(y, z)$) is preserved by min
- Suppose every variable x in an instance Φ of this CSP appears in a constraint $x > y \vee x > z$, for some variables y, z. Then Φ is unsatisfiable.

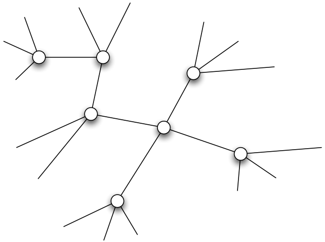
Claim (proof comes later): $(\mathbb{Q}; <)$ and $(\mathbb{Q}; \{(x, y, z) \mid x > y \vee x > z\})$ have only finitely many inequivalent primitive positive definable relations, for all n.

Unsatisfiable Instances of High-Girth

Let k, l be arbitrary.

Unsatisfiable Instances of High-Girth

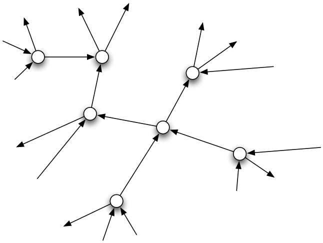
Let k, l be arbitrary.



Take a 4-regular graph G of girth $\geq 2k$

Unsatisfiable Instances of High-Girth

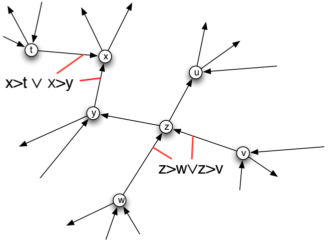
Let k, l be arbitrary.
Take a 4-regular graph G of girth $\geq 2k$



Orient along an Euler tour

Unsatisfiable Instances of High-Girth

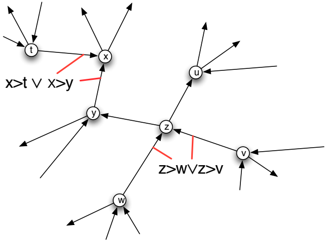
Let k, l be arbitrary.
Take a 4-regular graph G of girth $\geq 2k$
Orient along an Euler tour



Transform into an instance of the CSP

Unsatisfiable Instances of High-Girth

Let k, l be arbitrary.
Take a 4-regular graph G of girth $\geq 2k$
Orient along an Euler tour



Transform into an instance of the CSP
Facts:

- the resulting instance A of CSP$(\mathbb{Q}; x > y \vee x > z)$ is unsatisfiable
- Duplicator wins the existential pebble game on A, $(\mathbb{Q}; x > y \vee x > z)$

How Duplicator should play

A connected subgraph G' of G is called dominated if

How Duplicator should play

A connected subgraph G' of G is called dominated if

- all vertices have either indegree 2, or indegree 0 (leaves);

How Duplicator should play

A connected subgraph G' of G is called dominated if

- all vertices have either indegree 2, or indegree 0 (leaves);
- all vertices have outdegree 1, or outdegree 0 (root);

How Duplicator should play

A connected subgraph G' of G is called dominated if

- all vertices have either indegree 2, or indegree 0 (leaves);
- all vertices have outdegree 1, or outdegree 0 (root);
- all leaves must be pebbled.

How Duplicator should play

A connected subgraph G' of G is called dominated if

- all vertices have either indegree 2, or indegree 0 (leaves);
- all vertices have outdegree 1, or outdegree 0 (root);
- all leaves must be pebbled.

Observe: G' has size at most $2k$.

How Duplicator should play

A connected subgraph G' of G is called dominated if

- all vertices have either indegree 2, or indegree 0 (leaves);
- all vertices have outdegree 1, or outdegree 0 (root);
- all leaves must be pebbled.

Observe: G' has size at most $2k$.

Goal of Duplicator: if the root r in a dominated tree is pebbled, it's value is strictly larger than the minimum of the values assigned to the leaves.

How Duplicator should play

A connected subgraph G' of G is called dominated if

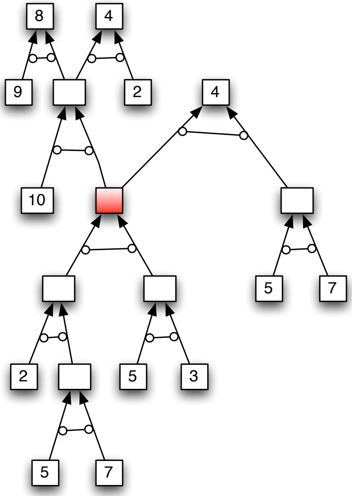
- all vertices have either indegree 2, or indegree 0 (leaves);
- all vertices have outdegree 1, or outdegree 0 (root);
- all leaves must be pebbled.

Observe: G' has size at most $2k$.

Goal of Duplicator: if the root r in a dominated tree is pebbled, it's value is strictly larger than the minimum of the values assigned to the leaves.

Clearly, this is satisfied at the beginning of the game.

How Duplicator should play



How Duplicator should play