Computability and Definability

J.A. Makowsky

Department of Computer Science
Technion - Israel Institute of Technology
Haifa, Israel

janos@cs.technion.ac.il
http://cs.technion.ac.il/~janos

Course given the Technion, Haifa, Israel

Winter Semester, 2015/6

Lecture 1: Prelude

- Complexity
- Definability
- Descriptive complexity
- References

Computing devices, I

Device: **Input** $\rightarrow$ $\boxed{\textbf{Device D}}$ $\rightarrow$ **Output**

Machines: Finite Automaton, Turing Machine (with resource bounds), Register Machine (with resource bounds),

Circuits: Boolean and Algebraic Circuits

Formulas: Formulas of First Order Logic FOL, Second Order Logic SOL, Monadic Second Order Logic MSOL, Fixed Point Logic, Temporal logic, etc

Computing devices, II

Transducer:

$$\text{In-structure} \rightarrow \boxed{\text{Device T}} \rightarrow \text{Out-structure}$$

Acceptor:

$$\text{Input} \rightarrow \boxed{\text{Device A}} \rightarrow \{0, 1\}$$

Counter:

$$\text{Input} \rightarrow \boxed{\text{Device C}} \rightarrow \mathbb{N}$$

Combinatorial problems, I

Acceptors: Deciding properties of a graph

Connected, cycle-free, hamiltonian, 3-colorable

$$\text{Graph} \to \boxed{\text{Device A}} \to \{0,1\}$$

Transducers: Finding configurations in a graph

Connected component, (hamiltonian) cycle, 3-coloring

$$\text{Graph} \to \boxed{\text{Device T}} \to \text{Graph}$$

Counters: Counting configurations in a graph

Connected components, (hamiltonian) cycles,

$$\text{Graph} \to \boxed{\text{Device C}} \to \mathbb{N}$$

Input for Devices

- For Finite Automata and Turing Machines the inputs are **coded** as (finite) words over some alphabet Σ.
- For Boolean circuits the inputs are **coded** as Boolean vectors in $\bigcup_n \{0,1\}^n$.
- For Algebraic circuitS over a field or ring $\mathcal{R}$, the inputs are **coded** as vectors over $\bigcup_n \mathcal{R}^n$.
- For Register Machines we may have specialized registers for specific data types, including words, natural numbers, real numbers, finite relations, etc.....

Complexity theory, I

Each machine type uses resources:

- Computing time
- Number of gates
- Space on tape
- Number of auxiliary registers
- Content size of registers

Complexity theory, II

Computability: There is a machine which solves the problem.

Complexity: There is a machine which solves the problem
with prescribed resources.

Machine classes: Determinsitic Finite Automata,
Non-determinsitic Finite Automata,
Pushdown Automata, Weighted Automata

Determinsitic Complexity classes: Time($f(n)$), Space($f(n)$),
PTime $= \mathbf{P}$, LogSpace $= \mathbf{L}$, $\mathbf{PSpace}$.

Non-determinsitic Complexity classes: NTime($f(n)$), NSpace($f(n)$),
NPTime $= \mathbf{NP}$, NLogSpace $= \mathbf{NL}$,
NPSpace.

$$\mathbf{L} \subseteq \mathbf{NL} = \mathbf{CoNL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PH} \subseteq \mathbf{PSpace} \subseteq \mathbf{ExpTime}$$

Complexity theory, III

Upper bounds: Problem $\mathcal{P}$ **can** be solved in the prescibed resource bounds.

Lower bounds: Problem $\mathcal{P}$ **cannot** be solved in the prescibed resource bounds.

Relative bounds: Problem $\mathcal{P}$ needs **at least/most** the amount of resources as problem $\mathcal{P}'$.

Definability, I

We specify a **problem** (a set of instances) in a **formal language**.

Formal languages can be

- Regular expressions for sets of words.

  The words over $\{a, b\}$ where all the a's come before the b's.

- First order logic FOL for sets of graphs.

  The regular graphs of degree 5.

- Second order logic SOL for sets of graphs.

  The connected graphs.

- Temporal logic for behaviour of programs.

  Inputs on which the program terminates.

Definability, II

A problem $\mathcal{P}$ is definable in a formal language $\mathcal{L}$
if there is an expression (a formula) of $\mathcal{L}$ which characterizes exactly the instances of $\mathcal{P}$.

Definable in $\mathcal{L}$: Connectivity of graphs is definable in Monadic Second Order Logic MSOL.

Non-definable in $\mathcal{L}$: Connectivity of graphs is not definable in First Second Order Logic FOL.

Relative-definable in $\mathcal{L}$: A graph is Eulerian in any logic $\mathcal{L}$ where being of even cardinality is definable.

Definability, III

How do we prove **definability** in a given logic $\mathcal{L}$?

- We translate the set theoretic concept directly into the logic.

  A graph has no edges.

- We first translate the set theoretic concept $\mathcal{C}$ into another concept $\mathcal{C}'$ and prove their equivalence.

  A graph is Eulerian iff it is connected and each vertex has even degree.

  This may be a (difficult) theorem of mathematics. .

  Then we translate $\mathcal{C}'$ into $\mathcal{L}$.

How do we prove **non-definability** in a given logic $\mathcal{L}$?

- **We have to develop special tools!**

Descriptive Complexity

We are looking for theorems of the form:

A class of objects $\mathcal{O}$ is

computable with specific resource bounds

iff

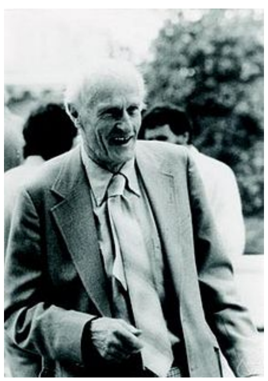
it is definable in a specific logic $\mathcal{L}$.

The first theorem of this firm was discovered during World War II independently in the USA (by S. Kleene) and in Poland (by A. Mostowski).

The Kleene-Mostowski Theorem (1943, 1947)

A set $A \subset \mathbb{N}$ of natural numbers is **recursively enumerable**
(or equivalently **semi-computable** by a Turing machine) iff
A is definable in the arithmetic structure of the natural numbers
$\langle \mathbb{N}, +, \times, <, 0, 1 \rangle$ by a Σ^0_1 formula.

Σ^0_0 formulas are FOL formulas with only bounded quantifiers $\exists x < t, \forall x < t$. Σ^0_1 formulas are FOL formulas of the form $\exists x \phi(x)$ where $\phi \in \Sigma^0_0$.



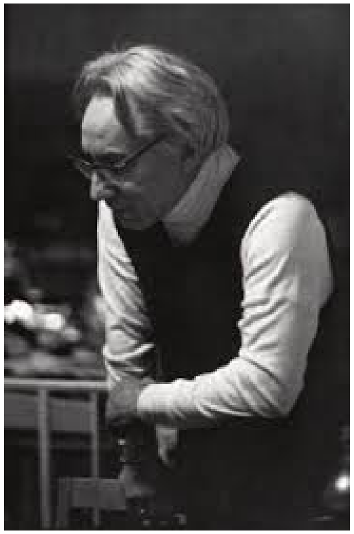
S. Kleene

A. Mostowski

The Büchi-Elgot-Trakhtenbrot Theorem (1958, 1960)

A language (set of words) is recognizable by a Finite Automaton iff it is definable in (existential) Monadic Second Order Logic.

. R. Büchi

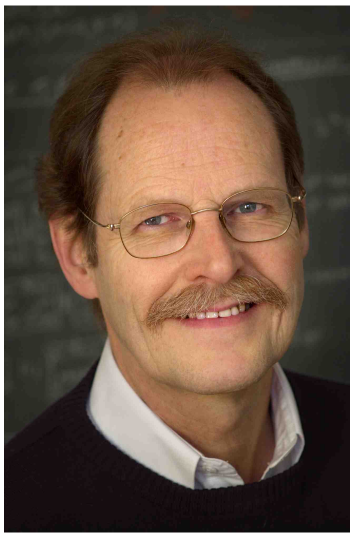


C. Elgot

B. Trakhtenbrot

The Jones-Selman-Fagin Theorem (1974)

A language (set of words) is recognizable by a non-deterministic Turing machine in polynomial time iff it is definable in existential Second Order Logic.



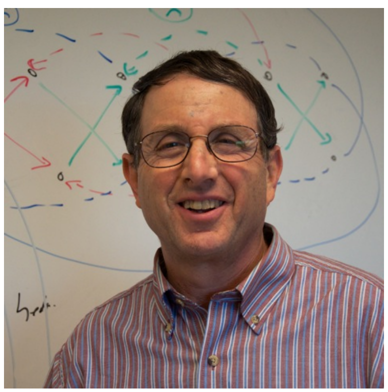
. N. Jones

A. Selman

R. Fagin

The Immerman-Vardi-Grädel Theorem (1980, 1991)

A language (set of words) is recognizable by a deterministic Turing machine in polynomial time iff it is definable in existential Second Order Logic with Horn formulas.



N. Immerman

M. Vardi

E. Grädel

References: Textbooks

[Pa94] C. Papadimitriou,
Computational Complexity,
Addison-Wesley 1994

[EF95] H.-D. Ebbinghaus and Jörg Flum,
Finite Model Theory,
Perspectives in Mathematical Logic,
Springer 1995, ISBN 3-540-60149-X

[Li04] L. Libkin,
Elements of Finite Model Theory, Springer, 2004

[Bo99] B. Bollobas,
Modern Graph Theory,
Graduate Texts in Mathematics, Springer 1999.

[Di97] R. Diestel,
Graph Theory,
Graduate Texts in Mathematics, Springer 1997.

Lecture 1:

Second Order Logic SOL and its fragments.

In this course we look at (labeled) graphs and other relational structures.

- The basic definitions.

Logics, a reminder

We define logics.

- Vocabularies: The basic relations
- Structures: Interpretations of vocabularies
- Variables: Indivicual variables, relation variables, function variables
- Atomic formulas
- Boolean closures
- Quantifications

Vocabularies

A vocabulary is a (finite) set of basic symbols.

We deal with (possibly many-sorted) relational vocabularies.
The basic symbols are sorts symbols and relation symbols.

Sort symbols: $U_\alpha : \alpha \in \mathbb{N}$

Relation symbols: $R_{i,\alpha} : i \in \mathrm{Ar}, \alpha \in \mathbb{N}$ where Ar is a set of arities, i.e. of finite sequences of sort symbols.

Constant symbols: $c_{\alpha,\beta}$ for $\alpha, \beta \in \mathbb{N}$, where α indicates the sort number.

In the case of one-sorted vocabularies, the arity is just of the form $\underbrace{\langle U, U, \ldots\ldots, U \rangle}_{n}$ which will denoted by n.

Vocabularies are denoted by greek letters $\tau, \sigma, \tau_i, \sigma_i$ with $i \in \mathbb{N}$.

τ-structures, I

τ-structures are interpretations of vocabularies.

More precisely, a τ-structure is a function assigning subsets of cartesian products of a fixed set A to each symbol.

$$\mathfrak{A} : \tau \to A \cup \bigcup_{n=1}^{\infty} \wp(A^n)$$

with the following restrictions:

- $\mathfrak{A}(U_\alpha) = A_\alpha \subseteq A$
- $\mathfrak{A}(U_\alpha) \cap \mathfrak{A}(U_\beta) = \emptyset$ for $\alpha \neq \beta$
- If $i = \langle U_{\alpha_1}, \ldots, U_{\alpha_k} \rangle$ is the arity of $R_{i,\alpha}$ then $\mathfrak{A}(R_{i,\alpha}) \subseteq A_{\alpha_1}, \times \ldots, \times A_{\alpha_k}$
- $\mathfrak{A}(c_{\alpha,\beta}) \in A_\alpha$.

τ-structures, II: Graphs and hypergraphs

Graphs and digraphs: $\tau_{graph} = \{U_1, R_{2,1}\}$.

The elements of the set $\mathfrak{A}(U_1) = V$ are called vertices. The subset $\mathfrak{A}(R_{2,1}) = E \subseteq V^2$ is called the (directed) edge relation.
If E is symmetric, the τ-structures is an undirected graph, otherwise it is a directed graph (aka digraph).
If $(u, u) \in E$ the veretx u has a loop.

Hypergraphs: $\tau_{hgraph} = \{U_1, U_2, R_{\langle 1,2 \rangle,1}\}$

The elements of the set $\mathfrak{A}(U_1) = V$ are called vertices.
The elements of the set $\mathfrak{A}(U_2) = E$ are called edges.
The subset $\mathfrak{A}(R_{\langle 1,2 \rangle,1}) \subseteq V \times E$ is called the undirected incidence relation.

Directed hypergraphs: $\tau_{hgraph} = \{U_1, U_2, R_{\langle 1,2,1 \rangle,1}\}$

The elements of the set $\mathfrak{A}(U_1) = V$ are called vertices.
The elements of the set $\mathfrak{A}(U_2) = E$ are called edges.
The subset $\mathfrak{A}(R_{\langle 1,2,1 \rangle,1}) \subseteq V \times E \times V$ is called the directed incidence relation.

τ-structures, III: Labeled graphs and words

Vertex labeled Graphs: Graphs with ℓ-many vertex labels, $\ell \in \mathbb{N}$:
$\tau_{lgraph} = \{U_1, R_{2,1}, P_1, \ldots, P_\ell\}$,
like graphs but with unary predicates P_i for vertex labels.

Edge labeled Graphs: Graphs with ℓ-many edge labels, $\ell \in \mathbb{N}$:
$\tau_{lgraph} = \{U_1, R_{2,i}\}$ with $i = 1, \ldots, \ell$,
like graphs but with ℓ-many edge relations for edge labels.

Words in Σ^*: Let Σ be a finite alphabet (set).
$\tau_{word} = \{U_1, R_{2,1}, R_{1,a}\}$, $a \in \Sigma$, where
$\mathfrak{A}(R_{2,1})$ is a linear order, and
$\mathfrak{A}(R_{1,a}) \cap \mathfrak{A}(R_{1,b}) = \emptyset$ for $a, b \in \Sigma, a \neq b$, and $\bigcup_{a \in \Sigma} \mathfrak{A}(R_{1,a}) = \mathfrak{A}(U_1)$.
τ_{word}-structures satsifying these conditions are words in Σ^*.

Empty structures

In logic and universal algebra a τ-structure $\mathfrak{A}$ is non-empty, i.e., for at least one sort symbol $U_\alpha \in \tau$ the set $\mathfrak{A}(U_\alpha) \neq \emptyset$.

We allow empty structures!

The reason for not allowing empty structures is the axiomatization of First Order Logic FOL. The axiom

$$\forall x P(x) \Rightarrow \exists x P(x)$$

only holds in non-empty one-sorted τ-structures.

Making structures one-sorted

We can always make τ-structures into one-sorted τ'-structures:

- We replace the sorts $U_\alpha \in \tau$ by one sort $V \in \tau'$.
- We add for each sort $U_\alpha \in \tau$ a unary relation symbol $P_\alpha \in \tau'$.
- We replace each $R_{(\alpha_1,\ldots,\alpha_m),i} \in \tau$ by $R_{m,i} \in \tau'$. Constant symbols remain the same.

We then make a τ-structure $\mathfrak{A}$ into a τ'-structure $\mathfrak{A}'$ by setting

- $\mathfrak{A}'(V) = \bigcup_{U_\alpha \in \tau} \mathfrak{A}(U_\alpha)$, and
- $\mathfrak{A}'(P_\alpha) = \mathfrak{A}(U_\alpha)$
- $\mathfrak{A}(R_{(\alpha_1,\ldots,\alpha_m),i}) = \mathfrak{A}'(R_{m,i})$

Isomorphisms and homomorphisms of τ-structures

Let $\mathfrak{A}$ and $\mathfrak{B}$ be two τ-strucures on sets
$A = \bigcup_{\alpha, U_\alpha \in \tau} \mathfrak{A}(U_\alpha$ and $B = \bigcup_{\alpha, U_\alpha \in \tau} \mathfrak{B}(U_\alpha$ respectively.
Let $f : A \to B$ a function. f is a τ-homomorphism if

- For all $U_\alpha \in \tau$ we have:
  $a \in \mathfrak{A}(U_\alpha)$ iff $f(a) \in \mathfrak{B}(U_\alpha)$.
- For all $R_{(\alpha_1,\ldots,\alpha_m),i} \in \tau$ we have:
  $(a_1, \ldots, a_m) \in \mathfrak{A}(R_{(\alpha_1,\ldots,\alpha_m),i})$ iff $(f(a_1), \ldots, f(a_m)) \in \mathfrak{B}(R_{(\alpha_1,\ldots,\alpha_m),i})$.
- For all $c_\alpha \in \tau$ we have:
  $f(\mathfrak{A}(c_\alpha)) = \mathfrak{B}(c_\alpha)$.

f is a τ-isomorphism if additionally f is one-one and onto.

$\mathfrak{A}$ and $\mathfrak{B}$ are τ-isomorphic if there is a τ-isomorphism $f : A \to B$.

τ-substructures

Let $\mathfrak{A}$ and $\mathfrak{B}$ be two τ-strucures on sets
$A = \bigcup_{\alpha, U_\alpha \in \tau} \mathfrak{A}(U_\alpha$ and $B = \bigcup_{\alpha, U_\alpha \in \tau} \mathfrak{B}(U_\alpha$ respectively.

$\mathfrak{A}$ is isomorphic to a substructure of $\mathfrak{B}$ if there is a function $f : A \to B$ such that:

- f is one-one.
- For all $U_\alpha \in \tau$ we have:
  If $a \in \mathfrak{A}(U_\alpha)$ then $f(a) \in \mathfrak{B}(U_\alpha)$.
- For all $R_{(\alpha_1,\ldots,\alpha_m),i} \in \tau$ we have:
  If $(a_1, \ldots, a_m) \in A^m$ then
  $(a_1, \ldots, a_m) \in \mathfrak{A}(R_{(\alpha_1,\ldots,\alpha_m),i})$ iff $(f(a_1), \ldots, f(a_m)) \in \mathfrak{B}(R_{(\alpha_1,\ldots,\alpha_m),i})$.
- For all $c_\alpha \in \tau$ we have:
  $f(\mathfrak{A}(c_\alpha)) = \mathfrak{B}(c_\alpha)$.

If f is the identity, we say $\mathfrak{A}$ is a substructure of $\mathfrak{B}$.

Subgraphs and induced subgraphs

In graph theory an undirected graph G without multiple edges is given by two sets $V(G)$ and $E(G)$ with $E(G) \subseteq V(G)^{(2)}$.

Let G, H be two graphs.

Subgraph: H is a subgraph of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq V(H)^2 \cap E(G)$.

This corresponds to the notion of substructure for graphs viewed as hypergraphs. i.e., τ-structures for $\tau = \tau_{hgraph}$

Induced subgraph: H is an induced subgraph of G if $V(H) \subseteq V(G)$ and $E(H) = V(H)^{(2)} \cap E(G)$.

This corresponds to the notion of substructure for graphs viewed as graphs, i.e., τ-structures for $\tau = \tau_{graph}$

Isomorphisms: H and G are isomorphic as τ_{graph}-structures iff they are isomorphic as τ_{hgraph}-structures.

Properties of a τ-structure

A property of τ-structures is a class $\mathcal{P}$ of τ-structures closed under τ-isomorphisms.

Examples:

- All *finite* τ-structures.
- All $\{R_{2,0}\}$-structures where $R_{2,0}$ is interpreted as a linear order.
- Al finite 3-dimensional matchings $3DM$, i.e. all $\{R_{3,0}\}$-structures with universe A where the interpretation of $R_{3,0}$ contains a subset $M \subseteq A^3$ such that no two triples of M agree in any coordinate.
- All binary words which are palindroms.

We say a τ-structure $\mathcal{A}$ has property $\mathcal{P}$ iff $\mathcal{A} \in \mathcal{P}$.

First Order Logic FOL

We now assume our vocabularies are one-sorted with sort symbol V.

We define the set of formulas FOL(τ):

Variables: $u, v, w, \ldots$ ranging over elements of the interpretation of V.

Terms: Variables and constant symbols in τ are τ-terms.

Atomic formulas: For each $R_{m,j} \in \tau$ and τ-terms $t_1, \ldots, t_m$ the expressions $R_{m,j}(t_1, \ldots, t_m)$, $t_1 = t_2$ are atomic formulas in FOL(τ).

Boolean conncectives: If ϕ and ψ are in FOL(τ), so are $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \Rightarrow \psi$ and $\neg\phi$.

Quantifiers: If ϕ is in FOL(τ) and v is a variable, then $\exists v \phi$ and $\forall v \phi$ are in FOL(τ).

Second Order Logic SOL

We now define SOL(τ), the set of SOL-formulas for a vocabulary τ:

FOL : FOL(τ) $\subseteq$ SOL(τ) and SOL(τ) is closed under boolean connectives and first order quantification.

Second order variables: For each $m, j \in \mathbb{N} - \{0\}$ we have second order variables $X_{m,j}$ of arity m.
For each $X_{m,j}$ a second order variable, and τ-terms $t_1, \ldots, t_m$ the expression $X_{m,j}(t_1, \ldots, t_m)$, is an atomic formulas in SOL(τ).

Second order quantification: If $\phi \in$ SOL(τ) so are $\forall X_{m,j} \phi$ and $\exists X_{m,j} \phi$.

Monadic Second Order formulas MSOL(τ) are those where for the arity m of the second order variables we have $m = 1$.

Analogously, SOL^n(τ) is obtained by restricting the arity m of the second order variables to $m \leq n$.

Lecture 1: Definability in graph theory

In this course we look at (labeled) graphs and other relational structures.

- Graph properties are classes of graphs
  closed under graph isomorphism.

- Graph parameters are functions of graphs
  invariant under graph isomorphism
  with values in some domain, usually a ring or semi-ring such as the natural numbers $\mathbb{N}$ or the integers $\mathbb{Z}$ or the reals $\mathbb{R}$, or a polynomial ring in sveral indeterminates.

Second Order Logic (SOL)

- Second Order Logic is the natural language
  to talk about graph properties.

  We shall show this informally and only after that define
  the syntax and semantic of SOL.

- We shall see we can also use SOL to define graph parameters.

Second Order Logic SOL and some of its fragments.

Atomic formulas for graphs are $E(u, v)$ and $u = v$ for individual variables u, v, and $R(u_1, \ldots, u_m)$ for m-ary relation variables R.

- First Order Logic FOL:

  Closed under boolean operations and quantification over individual variables. No relation variables.

- Second order Logic SOL:

  Closed under boolean operations and quantification over individual and relation variables of arbitrary but fixed arity.

- Monadic Second order Logic MSOL:

  Closed under boolean operations and quantification over individual and unary relation variables.

Concrete graphs (in $\mathbb{R}^3$)

A concrete graph G is given by

- a finite set of points V in $\mathbb{R}^3$, and
- a finite set E of ropes linking two points v_1, v_2.
  The ropes are continuous curves which do not intersect.

Without loss of generality we can take the points also in $\mathbb{R}^m$ for $m \geq 3$.

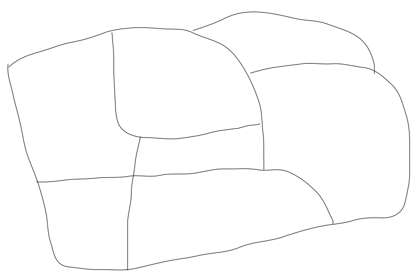
The ropes are called arcs.

Plane graphs

A plane graph G is given by

- a finite set of points V in $\mathbb{R}^2$, and
- finite set E of arcs linking two points v_1, v_2.

The arcs are continuous curves which do not intersect.



All intersection points in the drawing are points of the graph!

Abstract graphs

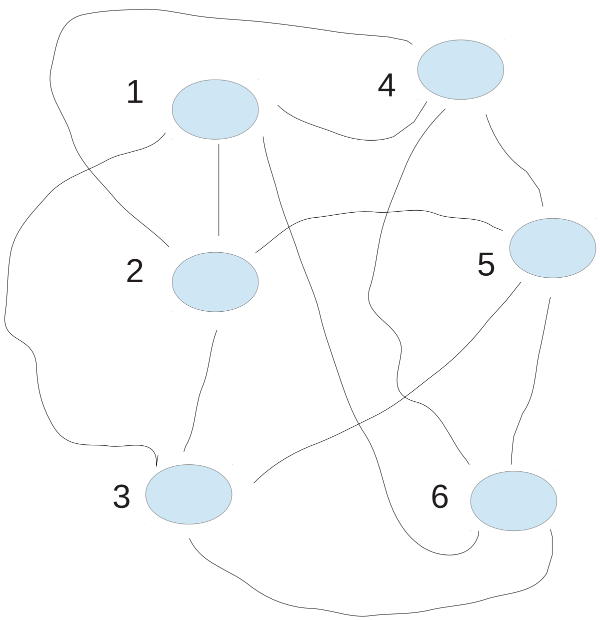
An abstract graph $G = (V(G), E(G))$ is given by

- a finite set of vertices $V = V(G)$, and
- a finite set $E = E(G)$ of edges linking two vertices v_1, v_2.

Here $E \subseteq V^{(2)}$ where $V^{(2)}$ denotes the set of unordered pairs of elements of V.

$V = \{1, \ldots, 6\}$

$$E = \begin{cases} \{(1,2),(2,3),(3,1)\} \cup \\ \{(4,5),(5,6),(6,4)\} \cup \\ \{(1,6),(6,3),(3,5),(5,2),(2,4),(4,1)\} \end{cases}$$



Graph isomorphism and subgraphs

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a function $f : V_1 \to V_2$ such that

- f is bijective (one-one and onto), and
- $(u, v) \in E_1$ iff $(f(u), f(v)) \in E_2$.

$G_1 = (V_1, E_1)$ is a subgraph of $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

$G_1 = (V_1, E_1)$ is an induced subgraph of $G_2 = (V_2, E_2)$ if $V_1 \subseteq V_2$ and for all $(u, v) \in V_1^{(2)} \cap E_2$ we also have $(u, v) \in E_1$.

$G_1 = (V_1, E_1)$ is a spanning subgraph of $G_2 = (V_2, E_2)$ if $E_1 \subseteq E_2$ and for all $u \in V_2$ $u \in V_1$ iff there is $v \in V_2$ with $(u, v) \in E_1$.

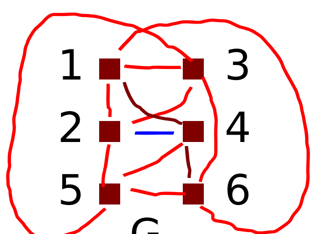
Two isomorphic graphs

$V_1 = V_2 = \{1, \ldots, 6\}$

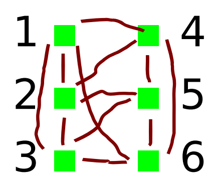
$$E_1 = \begin{cases} \{(1,2),(2,3),(3,1),(4,5),(5,6),(6,4)\} \cup \\ \{(1,6),(6,3),(3,5),(5,2),(2,4),(4,1)\} \end{cases}$$

$$E_2 = \begin{cases} \{(1,4),(4,3),(3,1),(5,2),(2,6),(6,5)\} \cup \\ \{(1,6),(6,3),(3,2),(2,4),(4,5),(5,1)\} \end{cases}$$

G_1 and G_2 are isomorphic with
$f(1) = 1, f(2) = 4, f(3) = 3, f(4) = 5, f(5) = 2, f(6) = 6.$



G

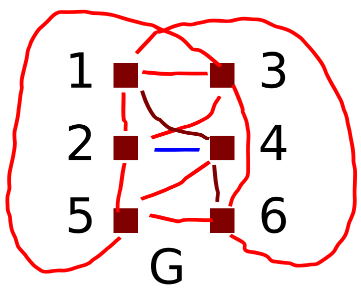


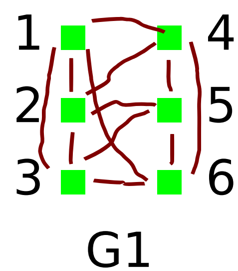
G1

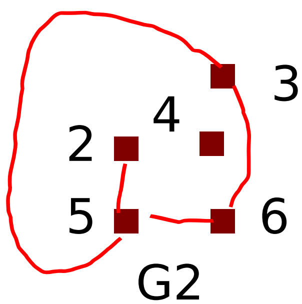
$G1$ is isomorphic to G.

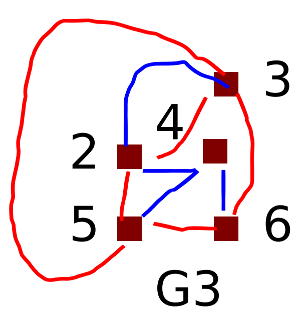
$G2$ is a subgraph of G, but not an induced subgraph.

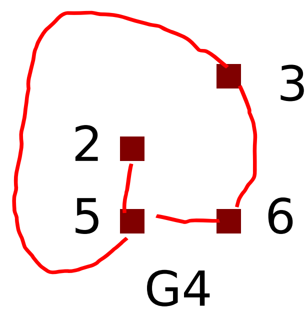
$G3$ is an induced subgraph and $G4$ is a spanning subgraph of G.











Some graph properties: Regularity

A graph G is (give definition in SOL):

- of degree bounded by $d \in \mathbb{N}$.

  Every vertex has at most d neighbors.

- k-regular ($k \in \mathbb{N}$)

  Every vertex has exactly k neighbors.

- regular

  Every vertex has exactly the same number of neighbors.

- Regular and degree bounded by d.

Definable in First Order Logic FOL

- The vertices $v_0, v_1, \ldots, v_n$ are all different:

$$\mathrm{Diff}(v_0, v_1, \ldots, v_n) : \left(\bigwedge_{i=0, j=1, i<j}^{i,j \leq n} v_i \neq v_j \right)$$

- A vertex v_0 has degree at most d:

$$\mathrm{Deg}_{\leq d}(v_0) : \forall v_1, \ldots, v_d, v_{d+1} \left(\bigwedge_{i=0}^{d+1} E(v_0, v_i) \to \bigvee_{i=0, j=0, i \neq j}^{i=d+1, j=d+1} v_i = v_j \right)$$

- A vertex v_0 has degree at least d:

$$\mathrm{Deg}_{\geq d}(v_0) : \exists v_1, \ldots, v_d \left(\mathrm{Diff}(v_1, \ldots, v_d) \wedge \bigwedge_{i=1}^{d} E(v_0, v_i) \right)$$

Regularity definable in .....

The following graph properties are definable in FOL (use previous slide):

- k-regular;
- regular and of bounded degree d;

The following are not definable in FOL (nor in Monadic Second order Logic MSOL):

- regular;
- each vertex has even degree.

To show non-definability in FOL we need the machinery of

Ehrenfeucht-Fraïssé Games or Connection matrices.

Regularity definable in .....

The following are definable in SOL:

- Two sets $A, B \subseteq V$ have the same size:

$$\text{EQS}(A,B) : \exists R\, (\text{Funct}(R,A,B) \wedge \text{Inj}(R) \wedge \text{Surj}(R))$$

  where $\text{Funct}(R,A,B), \text{Inj}(R), \text{Surj}(R)$ are FOL-formulas saying that R is a function from A to B which is one-one (injective) and onto (surjective).

- A vertex v has even degree:

  The set of neighbors of v can be partitioned into two sets of equal size

$$\text{EDeg}(v_0) : \exists A, B\, (\text{Part}(N_v, A, B) \wedge \text{EQS}(A,B))$$

- Two vertices u, v have the same degree:

  The set of neighbors N_u, N_v of u and v have the same size.

$$\text{SDeg}(u,v) : \text{EQS}(N_u, N_v)$$

Some graph properties: Closure proerties of graph classes.

A graph property is called

- hereditary if it is closed under induced subgraphs.
- monotone if it is closed under subgraphs, not necessarily induced.
- monotone decrasing if it is closed under deletion of edges, but not necessarily of vertices.
- monotone increasing if it is closed under addition of edges, but not necessarily of vertices.
- additive if it is closed under disjoint unions.

Note that **monotone** implies **hereditary** and **monotone decreasing**.

Examples for the closure properties

- d-regular graphs are only additive.
- Graphs of bounded degree d are monotone and additive.
- Cliques (complete graphs) are hereditary but not monotone.
- Connectivity is only monotone increasing.
- **Exercise:** Check the above closure properties of graph properties for your favorite graph properties.
- **Exercise:** Check the above closure properties of all the graph properties discussed in the sequel of this course.

Forbidden (induced) subgraphs

Let $\mathcal{H} = \{H_i : i \in I\}$ be a family of graphs.

- We denote by $\text{Forb}_{sub}(\mathcal{H})$ ($\text{Forb}_{ind}(\mathcal{H})$) the class of graphs G which have no (induced) subgraph isomorphic to some graph $H \in \mathcal{H}$.

- $\text{Forb}_{sub}(\mathcal{H})$ is monotone and $\text{Forb}_{ind}(\mathcal{H})$ is hereditary.

Theorem: (**Exercise**)
Let $\mathcal{P}$ be a monotone (hereditary) graph property. Then there exists a family $\mathcal{H} = \{H_i : i \in I\}$ of finite graphs such that $\mathcal{P} = \text{Forb}_{sub}(\mathcal{H})$ (respectively $\mathcal{P} = \text{Forb}_{ind}(\mathcal{H})$).

Proposition: Let $\mathcal{H} = \{H_i : i \in I\}$ be a family of graphs with I finite. Then both $\text{Forb}_{sub}(\mathcal{H})$ and $\text{Forb}_{ind}(\mathcal{H})$ are definable in FOL.

Homework 1

Characterize the following graph properties using
$\text{Forb}_{sub}(\mathcal{H})$ or $\text{Forb}_{ind}(\mathcal{H})$,
and determine their definability in FOL and SOL.

- Forests

- Cliques

- Find other examples! You may consult:

```
@BOOK(bk:BrandstaedtLeSpinrad,
AUTHOR         = {A. Brandst\"adt and V.B. Le and J. Spinrad},
TITLE          = {Graph Classes: A survey},
PUBLISHER      = {{SIAM} },
SERIES  = {{SIAM} Monographs on Discrete Mathematics and Applications},
YEAR           = {1999})
```

Some graph properties: Colorability

Let $\mathcal{P}$ be a graph property. A graph G is (*give definition in* SOL, MSOL):

- 3-colorable:

  The vertices of G can be partitioned into three disjoint sets $C_i : i = 1, 2, 3$ such that the induced graphs $G[C_i]$ consist only of isolated points.

  This can be expressed in MSOL.

- k-$\mathcal{P}$-colorable($k \in \mathbb{N}$):

  The vertices of G can be partitioned into k disjoint sets $C_i : i = 1, \ldots, k$ such that the induced graphs $G[C_i]$ are in $\mathcal{P}$.

  If $\mathcal{P}$ is definable in SOL (MSOL), this is also definable in SOL (MSOL).

- $\mathcal{P}$-colorable:

  The vertices of G can be partitioned into disjoint sets $C_i : i \in I \subset \mathbb{N}$ such that the induced graphs $G[C_i]$ are in $\mathcal{P}$.

  This is definable in SOL provided $\mathcal{P}$ is. It is not MSOL-definable.

k-colorable graphs

A subset V_1 of a graph $G = (V, E)$ is independent if it induces a graph of isolated points (without neighbors nor loops).

A graph is k-colorable if its vertices can be partitioned into k independent sets.

$$\text{Part}(X_1, X_2, X_3) :$$
$$((X_1 \cup X_2 \cup X_3 = V) \wedge ((X_1 \cap X_2) = (X_2 \cap X_3) = (X_3 \cap X_1) = \emptyset))$$

$$\text{Ind}(X) :$$
$$(\forall v_1 \in X)(\forall v_2 \in X)\neg E(v_1, v_2)$$

With this 3-colorable can be expressed as

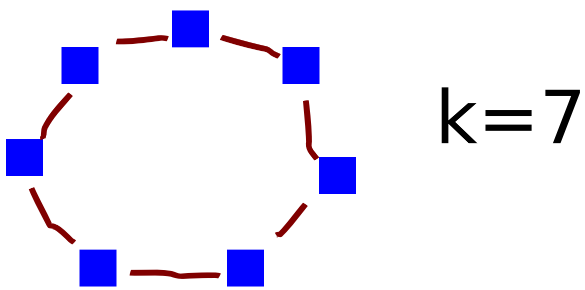
$$\exists C_1 \exists C_2 \exists C_3 \, (Part(C_1, C_2, C_3) \wedge Ind(C_1) \wedge Ind(C_2) \wedge Ind(C_3))$$

We have expressed 3-colorability by a formula in Monadic Second Order Logic.

Question: Can we express this in First Order Logic ?

Some graph properties: Chordality

A graph is a simple cycle of length k of it is of the form:



A graph is a simple cycle iff it is connected and 2-regular.

A graph G is chordal or triangulated if there is no induced subgraph of G isomorphic to a simple cycle of length ≥ 4.

Exercise: Find a MSOL-expression for chordality.

Some graph properties: Eulerian and Hamiltonian

A graph G is (*give definition in* SOL):

- Eulerian:

  We can follow each edge exactly once, pass through all the edges, and return to the point of departure.

  Theorem (Euler): A graph is Eulerian iff it is connected and each vertex has even degree.

- Hamiltonian:

  We can follow the edges visiting each vertex exactly once, and return to the point of departure.

Eulerian graphs

A graph $G = (V, E)$ is Eulerian if we can follow each edge exactly once, pass through all the edges, and return to the point of departure.

Equivalently:
Can we order all the edges of E

$$e_1, e_2, e_3, \ldots e_m$$

and choose beginning and end of th edge $e_i = (u_i, v_i)$ such that for all i, $v_i = u_{i+1}$ and $v_m = u_1$.

$$\exists R\,(\mathsf{LinOrd}(R, E) \wedge$$
$$(\forall u, v, u', v' \mathsf{First}(R, u, v) \wedge \mathsf{Last}(R, u', v') \to u = v') \wedge$$
$$(\forall u, v, u', v' \mathsf{Next}(R, u, v, u'v') \to v = u')\big)$$

whith the obvious meaning of LinOrd(R, E), First(R, u, v) and Last(u, v).

Alternatively, we can use Euler's Theorem.

As we shall see later, it cannot be expressed in MSOL.

Hamiltonian graphs

We note: A graph with n vertices is Hamiltonian if it contains a spanning subgraph which is a cycle of size n.

We define formulas:

Conn(V_1, E_1): (V_1, E_1) is connected.

Cycle(V_1, E_1): (V_1, E_1) is a cycle, i.e., regular of degree 2 and connected.

Ham(V, E) : $\exists V_1 \exists E_1 \left(Cycle(V_1, E_1) \wedge E_1 \subseteq E \wedge V_1 = V\right)$

A subtle point: Graphs vs hypergraphs, I

- Graphs are structures with universe V of vertices, and a binary edge relation E.
  There can be at most one edge between two vertices.

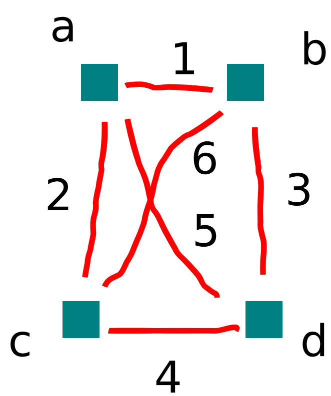
- Hypergraphs have as their universe two disjoint sets V and E and an incidence (hyperedge) relation $R(u, v, e)$.
  There can be many edges between two vertices.

- In both cases the relations are symmetric in the vertices.

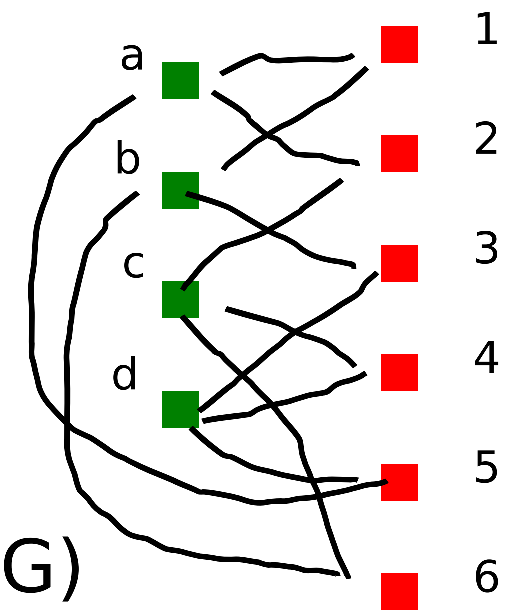
- A Graph G can be viewed as hypergraph (h-graph) $h(G)$ where there is at most one edge (up to symmetry) between two vertices.

- There is a one-one correspondence between graph and h-graphs.

G and $h(G)$



G



h(G)

A subtle point: Graphs vs hypergraphs, II

- FOL and SOL are equally expressive on graphs and h-graphs.
- MSOL is more expressive on h-graphs than on graphs.

  Hamiltonicity is not definable in MSOL on graphs, but is definable on h-graphs.

We shall discuss this in detail in a later lecture.

How to prove definablity in SOL, MSOL and FOL?

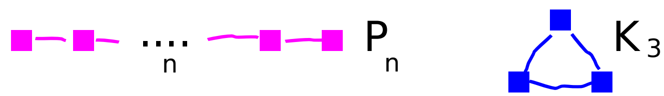
So far we have looked at properties of
abstract (directed) graphs and hypergraphs.

- Formulate the property using set theoretic language of finite sets over the set of vertices and edges and their incidence relation.

- Try to mimick this formulation in SOL.

- If you succeed, try to do it in MSOL or even FOL.

Test your fluency in SOL! (Homework)



Express the following properties in FOL, if possible.

- A graph G is a cograph if and only if there is no induced subgraph of G isomorphic to a P_4.
- A G is P_4-sparse if no set of 5 vertices induced more than one P_4 in G.
- Triangle-free graphs: There is no induced K_3.
- Existence of prescribed (induced) subgraph H.
- H-free graphs: non-existence of prescribed (induced) subgraph H.
- Let $\mathcal{P}$ be a graph property.
  $\mathcal{P}$-free graphs: non-existence of an induced subgraph $H \in P$.

Topological properties of graphs (from Wikipedia)

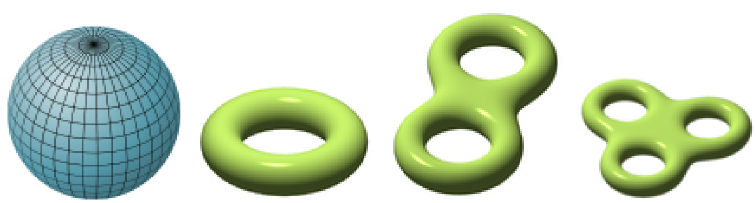
http://en.wikipedia.org/wiki/Genus_(mathematics)

So far our graph properties were formulated in the language of graphs, involving as basic concepts only vertices, edges and their incidence relations.

Topological graph theory studies the embedding of graphs in surfaces, spatial embeddings of graphs, and graphs as topological spaces.

- A graph is planar if it is isomorphic to a plane graph.
- The genus of a graph is the minimal integer n such that the graph can be drawn without crossing itself on a sphere with n handles (i.e. an oriented surface of genus n).

  Thus, a planar graph has genus 0, because it can be drawn on a sphere without self-crossing.



genus: $0, 1, 2, 3$

Planar graphs, I

A graph is planar iff it is isomorphic to a plane graph.

This definition involves the geometry of th Euclidean plane.

How can we express planarity

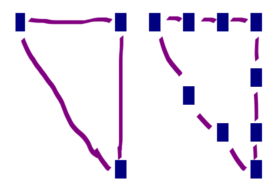
without geometry ?

Kuratowski's Theorem

Kazimierz Kuratowski (1896-1980)

http://en.wikipedia.org/wiki/Kuratowski's_theorem

A subdivision of a graph G is a graph formed by subdividing its edges into paths of one or more edges.



K_3 and a subdivision of K_3

Theorem: A finite graph G is planar if and only if it does not contain a subgraph that is isomorphic to a subdivision of K_5 or $K_{3,3}$.

Planar graphs, II

Theorem: Planarity is definable in MSOL.

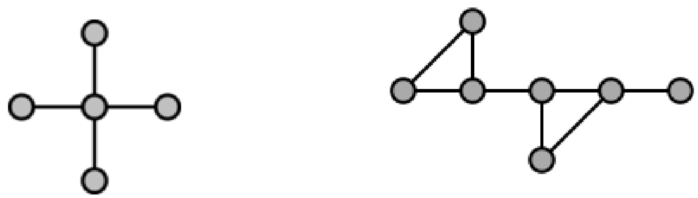
- We use Kuratowski's Theorem.
- For a fixed graph H, G is a subdivision of H, is definable in MSOL.
- For a graph property $\mathcal{P}$ definable in MSOL,
  G has a subgraph $H \in \mathcal{P}$, is definable in MSOL.

Exercise: Prove the last two statements.

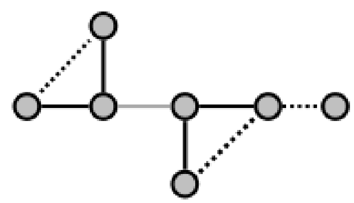
Graph minors, I

http://en.wikipedia.org/wiki/Graph_minor

An undirected graph H is called a minor of the graph G if H can be formed from G by deleting edges and vertices and by contracting edges.



H is a minor of G.



First construct a subgraph of G by deleting the dashed edges (and the resulting isolated vertex), and then contract the thin edge (merging the two vertices it connects).

Graph minors, II

Proposition: For fixed H the statement H is a minor of G is definable in MSOL.

- An edge contraction is an operation which removes an edge from a graph while simultaneously merging the two vertices it used to connect.
- An undirected graph H is a minor of another undirected graph G if a graph isomorphic to H can be obtained from G by contracting some edges, deleting some edges, and deleting some isolated vertices.
- The order in which a sequence of such contractions and deletions is performed on G does not affect the resulting graph H.
- Let $(V)H = \{v_1, \ldots, v_m\}$. We have to find $V_1, \ldots, V_m \subseteq V(G)$ which we all contract to a vertex u_i corresponding to v_i such that V_i connects to V_j iff $(v_i, v_j) \in E(H)$.
- The vertices in $V(G) - \bigcup_i^m V_i$ are discarded.

Minor closed graph classes

- H is a topological minor of G if G has a subgraph which is isomorphic to a subdivision of H.

- A graph property $\mathcal{P}$ is closed under (topological) minors, if whenever $G \in P$ and H is a (topological) minor of G the also $H \in P$.

Examples:

- Trees are not closed under minors, but forests are.

- Graphs of degree at most 2 are minor closed, but graphs of degree at most 3 are not.

- Planar graphs are both closed under minors and topological minors.

Forbidden minors, I

Let $\mathcal{H} = \{H_i : i \in I\}$ be a family of graphs.

- We denote by $\text{Forb}_{min}(\mathcal{H})$ ($\text{Forb}_{tmin}(\mathcal{H})$) the class of graphs G which have no (topoligical) minors isomorphic to some graph $H \in \mathcal{H}$.

- $\text{Forb}_{min}(\mathcal{H})$ is closed under topological minors, is monotone and hence, hereditary.

Theorem: (**Exercise**)
Let $\mathcal{P}$ be a graph property closed under (topological) minors. Then there exists a family $\mathcal{H} = \{H_i : i \in I\}$ of finite graphs such that $\mathcal{P} = \text{Forb}_{min}(\mathcal{H})$ (respectively $\mathcal{P} = \text{Forb}_{tmin}(\mathcal{H})$).

Proposition: Let $\mathcal{H} = \{H_i : i \in I\}$ be a family of graphs with I finite. Then both $\text{Forb}_{min}(\mathcal{H})$ and $\text{Forb}_{tmin}(\mathcal{H})$ are definable in MSOL.

The Graph Minor Theorem, 1983-2004

aka Robertson-Seymour Theorem

(formerly the Wagner conjecture, 1937)

Here is one of the deepest theorems in structural graph theory:

Theorem: Let $\mathcal{P}$ be a graph property closed under minors.
Then $\mathcal{P} = \text{Forb}_{min}(\mathcal{H})$ with $\mathcal{H}$ **finite**.

Corollary: Every graph property $\mathcal{P}$ property closed under minors
is definable in MSOL.

K. Wagner

N. Robertson

P. Seymour

Wagner's Theorem and Hadwiger's Conjecture

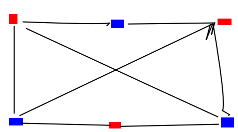
Theorem: A graph G is planar iff K_5 and $K_{3,3}$ are not minors of G.

- This gives another proof that planarity is MSOL-definable.

Conjecture: If a graph G is not k-colorable then its has the complete graph K_{k+1} as a minor.

The conjecture was proven for $k \leq 6$.

The converse is not true.
There are bipartite graphs with a K_4 minor.



Logic and Complexity: Regular languages

Let $L \subseteq \Sigma^\star$ be a *magenta language*, i.e., a set of words over the alphabet Σ.

We assume you are familiar with automata theory!

Theorem:(Kleene; Büchi, Elgot; Trakhtenbrot)

The following are equivalent:

- L is recognizable by a deterministic finite automaton.
- L is recognizable by a non-deterministic finite automaton.
- L is regular, i.e., describable by a regular expression
- The set of τ_{word}-structures $\mathfrak{A}_w$ with $w \in L$ is definable in MSOL(τ_{word}).

Complexity classes

We need to recall some complexity classes:

L: Deterministic logarithmic space.

NL: Non-deterministic logarithmic space.

P: Deterministic logarithmic space.

NP: Non-deterministic polynomial time.

PH: The polynomial hierarchy.

♯P: Counting predicates in **P** (Valiant's class)

PSpace: Deterministic polynomial space.

Complexity of SOL-properties

Fagin, Christen:
The **NP**-properties of classes of τ-structures are exactly the $\exists SOL$-definable properties.

Meyer,Stockmeyer:
The **PH**-properties (in the *polynomial hierarchy*) of classes of τ-structures are exactly the SOL-definable properties.

Makowsky, Pnueli:
For every level $\mathbf{\Sigma}_n^P$ of $\mathbf{PH}$ there are $MSOL$-definable classes which are complete for it.

Separating Complexity Classes, I

We have

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PH} \subseteq \sharp\mathbf{P} \subseteq \mathbf{PSpace}$$

- To show that $\mathbf{PH}$ does not collapse to $\mathbf{NP}$ we have to find a τ-sentence ϕSOL(τ) which is not equivalent over finite structures to an existential τ-sentence ψSOL(τ).

- Every sentence $\phi \in SOL(\tau)$ is equivalent (over finite structures) to an existential sentence $\psi \in SOL(\tau)$ iff $\mathbf{NP} = \mathbf{CoNP}$.
  Note we allow arbitrary arities of the quantified relation variables.
  Over infinite structures this is known to be false (Rabin)

- If there is a $\phi \in SOL(\tau)$ which is **not** equivalent to an existential sentence, then $\mathbf{P} \neq \mathbf{NP}$.
  And there should be such a sentence !

- To show that $\mathbf{PSpace}$ is different from $\mathbf{PH}$ it suffices to find a $\mathbf{PSpace}$-complete graph property which is not SOL-definable.

HEX and Geography, I

- The game HEX:
  Given a graph G and two vertices s, t.
  Players I and II color alternately vertices in $V - \{s, t\}$ white and black respectively.
  Player I tries to construct a white path from s to t and Player II tries to prevent this.

  HEX: The class of graphs which allow a Winning Strategy for player I.

- The game GEOGRAPHY:
  Given a **directed** graph G. Players I and II choose alternately new edges starting at the end point of the last chosen edge. The first who cannot find such an edge has lost.

  GEO: The class of graphs which allow a Winning Strategy for I.

HEX and Geography, II

Theorem (Even, Tarjan): HEX is **PSPACE**-complete.

Theorem (Schaefer): GEO is **PSPACE**-complete.

Problem: Are they SOL-definable?

This would imply that **PSPACE= PH**, and the polynomial hierarchy **collapses** to some finite level!

Short versions: Fix $k \in \mathbb{N}$.
SHORT-HEX, SHORT-GEOGRAPHY asks whether Player $\mathbf{I}$ can win in k moves.

S-HEX and S-GEO are the class of (orderd) graphs where player $\mathbf{I}$ has a winning strategy.

S-HEX and S-GEO are FOL-definable for fixed k.
(and therefore solvable in $\mathbf{P}$).

The role of order, I

Let $\tau_=$ be the one sorted vocabulary without any relation or constant symbols. We have only equality as atomic formulas.

Let $\tau_<$ be the one sorted vocabulary with one binary relation symbol $R_<$ which will e interpreted as a linear order.

- The class of structures of even cardinality EVEN is not definable in MSOL($\tau_=$).

  We shall prove this later.

- The class of structures of even cardinality EVEN is definable in MSOL($\tau_=$) by a formula ϕ_{EVEN}.

The role of order, II: Constructing ϕ_{EVEN}

We use the order to define the binary relation 2NEXT and the unary relation Odd

- For a structure $\mathfrak{A} = \langle A, < \rangle$, let $(a, b) \in \mathrm{2NEXT}^{\mathfrak{A}}$ iff $a < b$ and there is exactly one element strictly between a and b.
- The first element is in $\mathrm{Odd}^{\mathfrak{A}}$.
  If $a \in \mathrm{Odd}^{\mathfrak{A}}$ and $(a, b) \in \mathrm{2NEXT}^{\mathfrak{A}}$ then $b \in \mathrm{Odd}^{\mathfrak{A}}$.
- Let ϕ_{EVEN} be the formula which says that the last element is not in Odd.
- Now the a structure $\langle A, < \rangle$ is in EVEN iff its last element is not in $\mathrm{Odd}^{\mathfrak{A}}$.

Q.E.D.

The role of order, III: Order invariance

In the previous example EVEN the MSOL($\tau_<$)-formula ϕ_{EVEN} is order invariant in the following sense:

Let $\mathfrak{A}_1, \mathfrak{A}_2$ be two structures with universe A and different order relations $<_1$ and $<_2$.
Then $\mathfrak{A}_1 \models \phi_{EVEN}$ iff $\mathfrak{A}_2 \models \phi_{EVEN}$.

We generalise this:

Let $\mathfrak{A}_1, \mathfrak{A}_2$ be two $\tau \cup \{R_<\}$-structures with universe A and different order relations $\mathfrak{A}_1(R_<) =<_1$ and $\mathfrak{A}_2(R_<) =<_2$ but for all other symbols in $R \in \tau$ we have $\mathfrak{A}_1(R) = \mathfrak{A}_2(R)$.

A $\tau \cup \{R_<\}$-formula in SOL is order invariant if for all structures $\mathfrak{A}_1, \mathfrak{A}_2$ as above we have

$$\mathfrak{A}_1 \models \phi \text{ iff } \mathfrak{A}_2 \models \phi$$

The fragment HornESOL(τ).

- A quantifier-free τ-formula is a Horn clause if it is a disjunction of atomic or negated atomic formulas where at most one is not negated.

$$\neg\alpha_1 \vee \neg\alpha_2 \vee \ldots \vee \neg\alpha_n \vee \beta$$

  where α_i, β are atomic.

- A quantifier-free τ-formula is a Horn formula if it is a conjunction of Horn clauses.

- A formula $\phi \in$ SOL(τ) is in HornESOL(τ) of it is of the form

$$\exists U_{1,r_1}, U_{2,r_2}, \ldots, U_{k,r_k} \forall v_1, \ldots, v_m H(v_1, \ldots, v_m, U_{1,r_1}, U_{2,r_2}, \ldots, U_{k,r_k})$$

  where H is a Horn formula and v_i are first order variables.

Some classes of graphs order invariantly (o.i.) definable in HornESOL(τ_{graph})

- Graphs of even cardinality, of even degree. order is needed !
- Bipartite graphs $G = (V_1, V_2, E)$ with $|V_1| = |V_2|$.
- Regular graphs, and regular graphs of even degree.
- Connected graphs.
- Eulerian graphs.

To be discussed on the blackboard.

The Immermann-Vardi-Graedel Theorem (IVG)

Let τ be a relational vocabulary with a binary relation
for the ordering of the universe.

Theorem 1 (Immermann, Vardi, Graedel 1980-4)

Let $\mathcal{C}$ be a set of finite τ-structures. The following are equivalent:

- $\mathcal{C} \in \mathbf{P}$;
- *there is a τ-formula $\phi \in \mathrm{HornESOL}(\tau)$ such that $\mathfrak{A} \in \mathcal{C}$ iff $\mathfrak{A} \models \phi$.*

Here the presence of the ordering is crucial:

Without it the class of structures for the empty vocabulary of even cardinality is in $\mathbf{P}$, but not definable in HornESOL.

The Immermann-Vardi-Graedel Theorem (IVG):

Order invariant version

Let τ be a relational vocabulary and
$\tau_1 = \tau \cup \{R_<\}$. with a binary relation for the ordering of the universe.

Theorem 2 (Graedel 1980-4, Dawar, Makowsky)

Let $\mathcal{C}$ be a set of finite τ-structures. The following are equivalent:

- $\mathcal{C} \in \mathbf{P}$;
- *there is an order invariant τ_1-formula $\phi \in$ HornESOL(τ) such that for all τ-structures $\mathfrak{A}$ and linear orderings $R^A \subset \mathfrak{A}(V)^2$ $\mathfrak{A} \in \mathcal{C}$ iff $\langle \mathfrak{A}, R^A \rangle \models \phi$.*

Conclusion: The logical equivalent to $\mathbf{P} = \mathbf{NP}$

Let τ be a relational vocabulary which contains
a binary relation for the ordering of the universe.

The following are equivalent:

- $\mathbf{P} = \mathbf{NP}$ in the classical framework.
- Every ESOL(τ)-formula is equivalent over finite ordered τ-structures to some HornESOL(τ)-formula.
- Every o.i. ESOL(τ)-formula is equivalent over finite ordered τ-structures to some o.i. HornESOL(τ)-formula.

Logics capturing complexity classes

Without requiring the presence of order we have:

- A class $\mathcal{C}$ of finite structures is in $\mathbf{NP}$ iff $\mathcal{C}$ is definable in existential SOL.
- A class $\mathcal{C}$ of finite structures is in $\mathbf{PH}$ iff $\mathcal{C}$ is definable in SOL.

By requiring the presence of an order relation we have

- A class $\mathcal{C}$ of finite structures is in $\mathbf{P}$ iff $\mathcal{C}$ is 0.i. definable in existential HornESOL.
- There are similar theorems for $\mathbf{L}, \mathbf{NL}, \mathbf{PSpace}$.

Numeric graph invariants (graph parameters)

We denote by $G = (V(G), E(G))$ a graph,
and by $\mathcal{G}$ and $\mathcal{G}_{simple}$ the class of finite (simple) graphs, respectively.

A **numeric graph invariant** or **graph parameter** is a function

$$f : \mathcal{G} \rightarrow \mathbb{R}$$

which is invariant under graph isomorphism.

(i) Cardinalities: $|V(G)|$, $|E(G)|$

(ii) Counting configurations:

$k(G)$ the number of connected components,
$m_k(G)$ the number of k-matchings

(iii) Size of configurations:

$\omega(G)$ the clique number
$\chi(G)$ the chromatic number

(iv) Evaluations of graph polynomials:

$\chi(G, \lambda)$, the chromatic polynomial, at $\lambda = r$ for any $r \in \mathbb{R}$.
$T(G, X, Y)$, the Tutte polynomial, at $X = x$ and $Y = y$ with $(x, y) \in \mathbb{R}^2$.

Definability of numeric graph parameters, I

We first give examples where we use small, i.e., polynomial sized sums and products:

(i) The cardinality of V is FOL-definable by

$$\sum_{v \in V} 1$$

(ii) The number of connected components of a graph G, $k(G)$ is MSOL-definable by

$$\sum_{C \subseteq V : \text{component}(C)} 1$$

where component(C) says that C is a connected component.

(iii) The graph polynomial $X^{k(G)}$ is MSOL-definable by

$$\prod_{c \in V : \text{first}-\text{in}-\text{comp}(c)} X$$

if we have a linear order in the vertices and $\text{first} - \text{in} - \text{comp}(c)$ says that c is a first element in a connected component.

Definability of numeric graph parameters, II

Now we give examples with possibly large, i.e., exponential sized sums:

(iv) The number of cliques in a graph is MSOL-definable by

$$\sum_{C \subseteq V : \text{clique}(C)} 1$$

where clique(C) says that C induces a complete graph.

(v) Similarly "the number of maximal cliques" is MSOL-definable by

$$\sum_{C \subseteq V : \text{maxclique}(C)} 1$$

where maxclique(C) says that C induces a maximal complete graph.

(vi) The clique number of G, $\omega(G)$ is is SOL-definable by

$$\sum_{C \subseteq V : \text{largest} - \text{clique}(C)} 1$$

where largest − clique(C) says that C induces a maximal complete graph of largest size.

Definability of numeric graph parameters, III

Let $\mathcal{R}$ be a (polynomial) ring.

A numeric graph parameter $p : Graphs \to \mathcal{R}$ is $\mathcal{L}$-definable if it can be defined inductively:

- Monomials are of the form $\prod_{\bar{v}:\phi(\bar{v})} t$ where t is an element of the ring $\mathcal{R}$ and ϕ is a formula in $\mathcal{L}$ with first order variables $\bar{v}$.
- Polynomails are obtained by closing under small products, small sums, and large sums.

Usually, summation is allowed over second order variables, whereas products are over first order variables.

$\mathcal{L}$ is typically Second Order Logic or a suitable fragment thereof.
We are especially interested in MSOL and CMSOL, Monadic Second Order Logic, possibly augmented with modular counting quantifiers.

If $\mathcal{L}$ is SOL we denote the definable graphparameters by $\mathrm{SOLEVAL}_{\mathcal{R}}$, and similarily for MSOL and CMSOL.

Our definition of SOLEVAL is somehow reminiscent to the defintion of Skolem's definition of the Lower Elementary Functions.