
Computability and Definability, Part 2

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, 2021/2

<http://www.cs.technion.ac.il/~janos/COURSES/236331-21>

- Location: Taub 8
- Time: Thursday 9:30-12:30 (including Tirgul)
- Office: Taub 628
- Office hours: Thursday 14:00 or by appointment

ZOOM

- makowsky@technion.ac.il is inviting you to a scheduled Zoom meeting.
- Topic: CS 236331
- Time: This is a recurring meeting: Thursday 9:30-12:30
- Join Zoom Meeting at:

<https://technion.zoom.us/j/91510301938>

Course requirements

- Prerequisites are NOT enforced.
Recommended prerequisites: Some familiarity with the concepts of Logic and sets, Automata and formal languages or Computability
- Homework
- Final project

Overview

Part 2: Regular languages and modelchecking

- Regular languages are MSOL-definable
- The role of order
- HornESOL
- Beyond SOL
- Model checking
Resolution, Quantifier rank, FOL, SOL, HornESOL,

Session 3 (November 11, 2021)

Regular languages and MSOL

[Back to overview](#)

Regular languages, I

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

We assume you are familiar with *automata theory*!

Theorem:(S. Kleene (1953); M. Rabin and D. Scott (1959))

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*

Words as τ -structures (repetition)

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 satisfying these conditions are **words in Σ^*** .

Languages of Σ^* A language L is a subset of Σ^* represented as a set of τ_{word} -structure.

Regular languages, II

In this course we shall prove:

Theorem:(R. Büchi, C. Elgot, B. Trakhtenbrot (1960-62))

The following are equivalent:

- (i) L is **regular**, i.e., describable by a **regular expression**
- (ii) The set of τ_{word} -structures $\mathfrak{A}_w$ with $w \in L$ is definable in $\text{MSOL}(\tau_{word})$.

In this session we prove that (i) implies (ii).

Languages definable in SOL

Assume our alphabet is $\{a, b, c\}$.

The following are definable in SOL

- The set of words which have the same number of a 's and b 's.
- The set of words which have an even number of a 's and of b 's.
- The set of words which are palindroms, i.e., words which are the same when read backwards.

Languages definable in MSOL

Assume our alphabet is $\{a, b, c\}$.

The following are definable in MSOL

- The set of words which have an even number of a 's and of b 's.
- The set of words which consist of an arbitrary repetition of a fixed word.

The following are definable in FOL

- Every finite language L .
- The set of words which consist of an arbitrary repetition of fixed letter a followed by an arbitrary repetition of fixed letter b .

The following are **not** definable in MSOL

- The set of words which have the same number of a 's and b 's.
- The set of words which are palindroms, i.e., words which are the same when read backwards.

Regular expressions and MSOL

The collection of regular languages over an alphabet Σ is defined recursively as follows:

- The empty language $\emptyset$ is a regular language.
- For each $a \in \Sigma$ the singleton language $\{a\}$ is a regular language.
- If A is a regular language, A^* (Kleene star) is a regular language.
Due to this, the empty string language $\{\epsilon\}$ is also regular.
- If A and B are regular languages, then $A \cup B$ (union) and $A \circ B$ (concatenation) are regular languages.
- No other languages over Σ are regular.

Exercise: Prove by induction that every regular language of word-structures is MSOL-definable.

$$A \cup B$$

- Assume $\phi_A \in \text{MSOL}$ defines A and Assume $\phi_B \in \text{MSOL}$ defines B .
- A word w is in $A \cup B$ iff w is in A or in B .
- Therefore $\phi_A \vee \phi_B$ defines $A \cup B$.

$A \circ B$

- Assume $\phi_A \in \text{MSOL}$ defines A and Assume $\phi_B \in \text{MSOL}$ defines B .
- A word w is in $A \circ B$ iff $w = uv$ with $u \in A$ and $v \in B$.
- Let w be of length n and the universe of the structure w be $[n]$.
- So there is a position $i \in [n]$ in the word w such that the word w restricted to the first $i - 1$ positions gives a word $u \in A$, and the word w restricted all the positions $j \geq i$ gives a word $v \in B$.
- We write: $\exists i \left(\phi_A^{x < i} \wedge \phi_B^{x \geq i} \right)$
where x is a free individual variable.

We still have to explain what we mean by $\phi_A^{x < i}$ and $\phi_B^{x \geq i}$.

$$A^*$$

- Assume $\phi_A \in \text{MSOL}$ defines A .
- A word w is in A^* iff there is $i \in \mathbb{N}$ with $w = \underbrace{u \dots u}_i$ and $u \in A$.
- Let w be of length n and the universe of the structure w be $[n]$.
- So there is a set of positions $X \subseteq [n]$ in the word w such that the word w restricted to the interval defined by $j \in X$ and its successor $\text{succ}_X(j)$ in X is a word $u \in A$.
- We write: $\exists X \forall j \left(j \in X \rightarrow \left(\phi_A^{(j \leq x < \text{succ}_X(j))} \right) \right)$
where x is a free individual variable.

We still have to explain what we mean by $\phi_A^{(j \leq x < \text{succ}_X(j))}$.

Relativization of a formula, I

Let τ be a vocabulary and $\mathfrak{A}$ be a τ -structure.

$\phi \in \text{SOL}(\tau)$ a sentence, i.e. a formula without free variables.

Let $\theta(x)$ be a formula of $\text{SOL}(\tau)$ with one free individual variable x .

We denote by $\theta(\mathfrak{A})$ the set

$$\theta(\mathfrak{A}) = \{a \in A : (\mathfrak{A}, a) \models \theta(a)\}$$

i.e., the set of elements a of the universe of $\mathfrak{A}$ which satisfy $\theta(a)$ in $\mathfrak{A}$.

We define the τ -structure $\mathfrak{A}^{\theta(x)}$ as follows:

- The universe of $\mathfrak{A}^{\theta(x)}$ is the set $\theta(\mathfrak{A})$.
- The relations (functions) of $\mathfrak{A}^{\theta(x)}$ are the restrictions of the relations (functions) of $\mathfrak{A}$ to the set $\theta(\mathfrak{A})$.

We want to define the formula $\phi^{\theta(x)}$ such that

$$\mathfrak{A} \models \phi^{\theta(x)} \text{ iff } \mathfrak{A}^{\theta(x)} \models \phi$$

Relativization of a formula, II

$\phi^{\theta(x)}$ is called the **relativization of ϕ to $\theta(x)$** .

$\phi^{\theta(x)}$ is defined inductively over ϕ :

- ϕ **atomic** with free variables $x_1, \dots, x_m$: $\phi^{\theta(x)} = \bigwedge_{i \in [m]} \theta(x_i) \wedge \phi$.
- $\phi = (\phi_1 \wedge \phi_2)$: $(\phi_1 \wedge \phi_2)^{\theta(x)} = (\phi_1^{\theta(x)} \wedge \phi_2^{\theta(x)})$
- $\phi = \neg \phi$: $(\neg \phi)^{\theta(x)} = \neg(\phi^{\theta(x)})$
- $\phi = \exists y \phi$: $(\exists y \phi)^{\theta(x)} = \exists y(\theta(y) \wedge \phi^{\theta(x)})$
- $\phi = \exists R \phi$ with free variables $x_1, \dots, x_m$ as arguments of R :
 $(\exists R \phi)^{\theta(x)} = \exists R \left((\forall x_1, \dots, x_m R(x_1, \dots, x_m) \rightarrow \bigwedge_{i \in [m]} \theta(x_i)) \wedge \phi^{\theta(x)} \right)$

Compulsory exercise: Relativization

Prove by induction on ϕ :

$$\mathfrak{A} \models \phi^{\theta(x)} \text{ iff } \mathfrak{A}^{\theta(x)} \models \phi$$

Compulsory exercise: Regular languages

Prove (by induction):

Every regular language is definable in MSOL

Write the formula of MSOL for

$(c(ab)^*c)^*$

Example: *cabababababccabababccababc*

Back to overview

The role of order

- Word-structures are linearly ordered.
- What can we say about finite linear orders?

Back to overview

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 be interpreted as a linear order.

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

We shall prove this later.

- The class of structures of even cardinality EVEN is definable with order in $\text{MSOL}(\tau_<)$ 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 $(a, b) \in 2NEXT^{\mathfrak{A}}$ iff $a < b$ and there is exactly one element strictly between a and b .
- The first element is in $Odd^{\mathfrak{A}}$.
If $a \in Odd^{\mathfrak{A}}$ and $(a, b) \in 2NEXT^{\mathfrak{A}}$ then $b \in Odd^{\mathfrak{A}}$.
- 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 $Odd^{\mathfrak{A}}$.

Q.E.D.

Spectrum of a sentence ϕ

Let τ be a vocabulary and $\phi \in \text{SOL}(\tau)$.

The **spectrum of ϕ** , $S(\phi)$, is the set of natural numbers $n \in \mathbb{N}$ such that there is a finite structure $\mathfrak{A}$ with n elements which satisfies ϕ .

Example: The even numbers are the spectrum of ϕ_{EVEN} .

Challenge:

- For which $A \subseteq \mathbb{N}$ is there an FOL-definable class of finite sets (empty vocabulary) with spectrum A ?
- For which $A \subseteq \mathbb{N}$ is there an FOL-definable class of finite linear orders with spectrum A ?
- Is there a FOL-definable class of finite linear orders with spectrum the primes?
- Replace FOL by MSOL or SOL in the above questions.

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

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

Back to overview

The fragment HornESOL(τ), I

- 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.

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

where α_i, β are atomic.

- Let $\tau_0 \subseteq \tau$.
A quantifier-free τ -formula is a **Horn clause over τ_0** if it is a **disjunction** of atomic or negated atomic formulas where **among all the occurrence of a relation symbol from τ_0 at most one** occurrence is **not negated**.
- A quantifier-free τ -formula is a **Horn formula over τ_0** if it is a **conjunction** of Horn clauses.

Examples of Horn clauses, I

Let $\tau = \{P_1, P_2, Suc, First, Last, <\}$ with $P_1, P_2, First, Last$ unary and $Suc, <$ binary. and $\tau_0 = \{P_1, P_2, First, Last, Suc\}$. We write $x \leq y$ for $(x < y \vee x = y)$. The following are Horn clauses over τ_0 :

- $\phi_1(Suc, x, y) = (Suc(x, y) \rightarrow x < y)$ is equivalent to $(\neg Suc(x, y) \vee x < y)$
 $\phi_2(Suc, x, y, z) = ((x \leq z \wedge z \leq y \wedge Suc(x, y)) \rightarrow (x = z \vee y = z))$
 is equivalent to
 $(\neg(x < z) \vee \neg(z < y) \vee \neg Suc(x, y) \vee x = z \vee y = z).$
Suc is negated, but $x < y$ can occur *negated and non-negated* in τ_0 -Horn formulas.

$\forall x, y, z(\phi_1(Suc, x, y) \wedge \phi_2(Suc, x, y, z))$ says Suc is a successor relation of the linear order.

- $P_1(x) \leftrightarrow \neg P_2(x)$ is equivalent to
 $(\neg P_1(x) \vee \neg P_2(x)) \wedge (P_2(x) \vee P_1(x))$ is *not* a conjunction of Horn clauses.
Both P_1, P_2 are non-negated.
 $\forall x(\neg P_1(x) \vee \neg P_2(x)) \wedge (P_2(x) \vee P_1(x))$ says P_1, P_2 is a partition.

Examples of Horn clauses, II

We want to say that a linear order has even cardinality.

$$\begin{aligned}
 & \exists Suc, P_1, P_2, First, Last \\
 & \forall x_1, x_2, y_1, y_2, x, y, z \\
 & ((First(x_1) \rightarrow \neg(y_1 < x_1)) \\
 & \quad \wedge (First(x_1) \rightarrow P_1(x_1)) \\
 & \quad \wedge (Last(x_2) \rightarrow \neg(x_2 < y_2)) \\
 & \quad \wedge (Last(x_2) \rightarrow P_2(x_2)) \\
 & \quad \wedge (\neg P_1(x) \vee \neg P_2(x)) \\
 & \quad \wedge ((P_1(x) \wedge Suc(x, y)) \rightarrow P_2(y)) \\
 & \quad \wedge ((P_2(x) \wedge Suc(x, y)) \rightarrow P_1(y)) \\
 & \quad \wedge (Suc(x, y) \rightarrow x < y) \\
 & \quad \wedge ((x < z \wedge z < y \wedge Suc(x, y)) \rightarrow (x = z \vee y = z)))
 \end{aligned} \tag{1}$$

The fragment HornESOL(τ), I

- A formula $\phi \in \text{SOL}(\tau)$ is in $\text{HornESOL}(\tau)$ if it is of the form

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

where H is a Horn formula over $U_{1,r_1}, U_{2,r_2}, \dots, U_{k,r_k}$
and v_i are first order variables.

Some classes of graphs order invariantly (o.i.) definable in $\text{HornESOL}(\tau_{\text{graph}})$

The following are definable in $\text{HornESOL}(\tau)$.

We know from a later theorem, the Immerman-Vardi-Grädel Theorem, that this is true, but it may be difficult and not at all obvious, to write down the formula in $\text{HornESOL}(\tau)$.

- Graphs 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.

You can view this as a very instructive homework!

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 $\phi \in \text{HornESOL}(\tau)$ 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 $\phi \in \text{HornESOL}(\tau)$ such that for all τ -structures $\mathfrak{A}$ and linear orderings $R^A \subset \mathfrak{A}(V)^2$ $\mathfrak{A} \in \mathcal{C}$ iff $\langle \mathfrak{A}, R^A \rangle \models \phi$.

Logics capturing complexity classes

an outlook

Without requiring the presence of order we have:

- (R. Fagin; C. Christen; N. Jones and A. Selman, all around 1973)
A class $\mathcal{C}$ of finite structures is in **NP** iff $\mathcal{C}$ is definable in existential second order logic ESOL.
- A class $\mathcal{C}$ of finite structures is in **PH**, the polynomial hierarchy, iff $\mathcal{C}$ is definable in SOL.
- There is a similar theorem for **PSpace**.

By requiring the presence of an order relation we have

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

Conclusion: The logical equivalent to $P = NP$

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

The following are equivalent:

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

Back to overvoew

Is every graph property SOL-definable?

How could we prove that **no**?

Let $\phi \in \text{SOL}$ define a graph property $\mathcal{P}$.

We crudely estimate the time complexity of

Modelchecking $\phi \in \text{SOL}$

Input: A graph G of order n

Question: Does G satisfy ϕ .

We show that there is a constant $k(\phi)$ such that

Modelchecking $\phi \in \text{SOL}$ can be solved in time $O(2^{n^{k(\phi)}})$.

A graph property not definable in SOL

Let $A \subseteq \mathbb{N}$ be non-computable.

- Such a set exists. There are uncountable many subsets of $\mathbb{N}$, but only countably many Turing machines recognizing subsets of $\mathbb{N}$.
- Let $G \in \mathcal{P}_A$ iff the order of G is in A .
- $\mathcal{P}_A$ is not SOL-definable.

Actually, in a precise sense, almost all graph properties are not SOL-definable.

Can we find a concrete example?

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.

What is a winning strategy?

We discuss the case of HEX.

A **winning strategy for HEX** is a table $WS(HEX)$ with three columns.

Position	Move of Player II	Move of Player I
$P_0 = (G, s, t)$	pass	a_1
$P_1^1 = (G, s, t, a_1)$	b_1^1	$a_2(P_1, b_1^1)$
P_1^1	b_2^1	$a_2(P_1, b_2^1)$
P_1^1	b_3^1	$a_2(P_1, b_3^1)$
	$\vdots$	
	$\vdots$	
P_1^1	b_k^1	$a_2(P_1, b_k^1)$
P	b	$a(P, b)$
	$\vdots$	
	$\vdots$	
$P(end)$		

where $P(end)$ is a winning position for Player I.

Is the existence of $WS(HEX)$ expressible in SOL?

- $WS(HEX) \not\subseteq V(G)^m$ for any m , i.e., it is not a relation.
- If $G = K_m$ is a complete graph, $m \geq 3$ and s and t have distance at least 2, then Player I has a winning strategy with the first move.
- The same holds for wheels W_m .
- Find graph properties $\mathcal{P}$ for which HEX is SOL-definable.

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 I can win in k moves.

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

S-HEX and S-GEO are FOL-definable for fixed k .
(and therefore solvable in **P**).

Back to overview

[File:w-hex.tex](#)

The complexity of $\mathfrak{A} \models \phi$

Propositional Logic

$$\phi \in FOL$$

$$\phi \in SOL$$

$$\phi \in \exists SOL$$

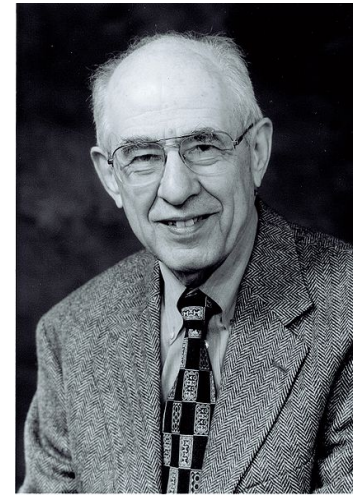
$$\phi \in HornSOL$$

We first discuss **upper** bounds

Resolution and the Davis-Putnam Procedure



Martin Davis, * 1928



Hilary Putnam, * 1926

- Davis, Martin; Putnam, Hillary (1960).
[A Computing Procedure for Quantification Theory.](#)
Journal of the ACM 7 (3): 201-215
- Davis, Martin; Logemann, George, and Loveland, Donald (1962).
[A Machine Program for Theorem Proving](#)
Communications of the ACM 5 (7)

Resolution, the propositional case, I

Here we recall a syntactic method of checking whether a set of formulas Σ of clauses is satisfiable called **resolution**.

Resolution is a **machine friendly** method which involves some preprocessing transforming a given set of propositional formulas into a set $clause(\Sigma)$ of **clauses**.

Clauses, I

- Var is the set of propositional variables $\{p_i : i \in \mathbb{N}\}$.
- A **literal** is either a propositional variable, p_i , or the negation of a propositional variable, $\neg p_i$. The constant $\perp$ is also a literal. We denote literals by l_j .
- A **clause** is a finite set of literals $\{l_1, l_2, \dots, l_k\}$. We denote clauses by C_j . We denote the empty clause (the empty set of literals) by $\square$.

Clauses, II

- Let z be a propositional assignment, i.e., a function

$$z : Var \rightarrow \{0, 1\}$$

The meaning function M_{clause} for clauses is defined inductively as follows:

Basis: $M_{clause}(\{p_j\}, z) = z(p_j)$,

$M_{clause}(\{\neg p_j\}, z) = 1 - z(p_j)$,

$M_{clause}(\square, z) = 0$.

Closure: $M_{clause}(\{l_1, l_2, \dots, l_k\}, z) = \max\{M_{clause}(\{l_j\}, z) : j \leq k\}$.

If S is a set of clauses $M_{clause}(S, z) = \min\{M_{clause}(C, z) : C \in S\}$.

- A set S of clauses is **satisfiable** if there is an assignment z such that $M_{clause}(S, z) = 1$.

Resolution trees

- (i) Let $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}$ be two clauses. We say that the clause $C_1 \cup C_2$ is obtained from $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}$ by **one resolution step**, and we write

$$\frac{C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}}{C_1 \cup C_2}$$

- (ii) A **resolution tree** T is a binary (directed) labeled tree $T = \langle V, E \rangle$ such that the labels of V are clauses. Furthermore, if C_1, C_2 are labels of the two sons of a vertex labeled with C then $C_1 = D_1 \cup \{p_j\}, C_2 = D_2 \cup \{\neg p_j\}$ and $C = D_1 \cup D_2$. In other words, the label of the father is the result of performing a resolution step on the labels of its two sons. Note that several vertices may carry the same label.
- (iii) Let S be a set of clauses and C be a clause. We say that **S proves C by resolution**, if there is a resolution tree with the root labeled C and all the leaves labeled with clauses from S . We write $S \vdash_{res} C$ for S proves C by resolution.

Examples

(i) Draw a resolution tree for $S \vdash_{res} \Box$ for

$$S = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}.$$

(ii) Draw a resolution tree for $S \vdash_{res} \Box$ for

$$S = \{\{p_1, p_2, p_3\}, \{\neg p_1, p_2\}, \{\neg p_3, p_2\}, \{\neg p_2\}\}.$$

(iii) Invent your own example and draw the tree!

Soundness of resolution

The next proposition establishes that resolution steps preserve the meaning of the clauses on which they are based. We call this the **soundness** of resolution steps. More precisely:

- (i) Let S be a set of clauses and $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\} \in S$.
Let z be a propositional assignment such that $M_{clause}(S, z) = 1$.
Then $M_{clause}(S \cup \{C_1 \cup C_2\}, z) = 1$.
- (ii) Let T be a resolution tree and S_0 be the set of clauses which are the labels of its leaves and S the set of all its labels.
Let z be a propositional assignment such that $M_{clause}(S_0, z) = 1$.
Then $M_{clause}(S, z) = 1$.
- (iii) If S is a set of clauses such that $S \vdash_{res} \square$ then S is not satisfiable.

Proof of soundness, I

We prove only (i).

To prove (ii), we can proceed by induction on the depth of the tree applying (i) as the induction step.

(iii) is a direct consequence of (i).

- So let $C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\} \in S$ be two clauses and z be a propositional assignment such that

$$M_{clause}(S \cup \{C_1 \cup \{p_j\}, C_2 \cup \{\neg p_j\}\}, z) = 1.$$

- We have to show that

$$M_{clause}(S \cup \{C_1 \cup C_2\}, z) = 1.$$

- It suffices to prove the case where $z(p_j) = 1$, as the case $z(p_j) = 0$ is similar.

Proof of soundness, II

- $z(p_j) = 1$ implies that $M_{clause}(C_2, z) = 1$ and therefore $M_{clause}(C_1 \cup C_2, z) = 1$.
- As $M_{clause}(S, z) = 1$ we also have $M_{clause}(S \cup \{C_1 \cup C_2\}, z) = 1$.
- this completes the proof of (i).

Q.E.D.

Completeness of resolution, I

We would like to state a completeness theorem for resolution.

The obvious formulation would be:

- (*) Let S be a set of clauses, and C be a clause.
Then $S \vdash_{res} C$ iff C is a logical consequence of S .

The following example shows that this is not true:

Counterexample: for (*)

Let S be $\{\{p_0\}\}$ and C be $\{p_0, p_1\}$.

Clearly C is a logical consequence of S .

However, there is no resolution step applicable given S only.

Completeness of resolution, II

The best we can hope for is the following:

Theorem: Completeness of Resolution for Satisfiability

(**) Let S be a set of clauses.

S is not satisfiable iff $S \vdash_{res} \square$.

This will follow from the **Compactness Theorem**

and the completeness of the **Davis–Putnam Procedure**.

End of session 3 (November 11, 2021)

Begin of session 4 (November 18, 2021)

We repeat and **expand** the material
about the **Davis-Putnam procedure**.

Then we continue with the complexity analysis
of model checking.

Reduced sets of clauses

- A clause C is **valid** if it contains at least one propositional variable p together with $\neg p$.
- A set of clauses S is **reduced** if it does not contain any valid clause.

Observation: Let S be a set of clauses, and let

$$\text{red}(S) = S - \{C \in S : C \text{ is valid} \}$$

S and $\text{red}(S)$ are logically equivalent.

Proof: Exercise.

The Davis–Putnam Procedure, I

- We choose a variable p occurring in a set of clauses S . As S is reduced, p and $\neg p$ do not occur together in a clause of S .
- We partition S into clauses $S(pos)$, $S(neg)$, $S(-)$ where p occurs positively, negatively, or not at all.
- We eliminate p from S by defining
$$S(no - p) = \{C \cup D : C \cup \{p\} \in S(pos), D \cup \{\neg p\} \in S(neg)\} \cup S(-)$$
- We prove that S is satisfiable iff $S(no - p)$.
- $S(no - p)$ may grow **quadratically**.

The Davis–Putnam Procedure, II

- Repeating this we reach $S(\text{no variables})$, which is either empty or has the empty clause as its only element.
- This can be achieved in n iterations, where n is the number of variables in S .
- It may need exponentially many resolution steps.

The Davis–Putnam Procedure, III

Let S be a finite set of **reduced** clauses.

Without loss of generality let $p_0, \dots, p_n$ be all the variables occurring in S .

We define inductively sets of clauses

$$S_j, S_j(pos), S_j(neg), S_j(-)$$

for $j = 0, \dots, n$ as follows:

- $S_0 = S$,
- $S_j(pos) = \{C \in S_j : p_j \in C\}$,
- $S_j(neg) = \{C \in S_j : \neg p_j \in C\}$,
- $S_j(-) = \{C \in S_j : p_j \notin C \text{ and } \neg p_j \notin C\}$,
- $S_{j+1} = S_j(-) \cup \{C \cup D : C \cup \{p_j\} \in S_j(pos), D \cup \{\neg p_j\} \in S_j(neg)\}$.

Preserving satisfiability

For every $j = 0, \dots, n$ S_j is satisfiable iff S_{j+1} is satisfiable.

Proof:

- If S_j is satisfiable then S_{j+1} is satisfiable, by the soundness of resolution.
- So let us assume that S_{j+1} is satisfiable by a truth assignment z_0 .
- Let z_1 be the truth assignment obtained from z_0 by putting $z_1(p_j) = 1 - z_0(p_j)$.
- If neither z_0 nor z_1 satisfy S_j then there are clauses C, D with

$$C \cup D \in \{C \cup D : C \cup \{p_j\} \in S_j(pos) \\ D \cup \{\neg p_j\} \in S_j(neg)\}$$

and $M_{clause}(C \cup D, z_0) = 0$.

- Which is a contradiction.

Termination

S_{n+1} is either empty or contains only the empty clause. Furthermore, S_{n+1} is empty iff S is satisfiable.

Proof: Exercise

Completeness of the Davis–Putnam Procedure

A finite set of clauses S is satisfiable iff the Davis–Putnam Procedure returns the **empty set** (not the empty clause).

Proof: Use the lemmas.

Complexity of Resolution

- The Davis–Putnam Procedure seems rather crude and may need an exponential number of resolution steps. Its performance is also sensitive to the numbering of the variables.
- It was shown in a sequence of papers (Tseitin, Galil, Haken, Urquhart, Szemeredi) that there are many sets of n clauses S which are unsatisfiable and which need an exponential number of resolution steps to discover the unsatisfiability.
- In the case of average complexity the situation is more complex as it depends on the input distribution for clauses. There are quite natural distributions for which resolution is polynomial on the average, and others, equally natural, for which resolution is exponential on the average.

Weakening and Unit Resolution

Variations on resolution

- The **weakening rule**: Let C, D be clauses with $C \subseteq D$.

$$\frac{C}{D}$$

Theorem A: Let S be a set of clauses, and C be a clause. Then C can be derived from S with resolution and weakening iff $S \models C$.

- **Horn clauses**: Clauses with at most one non-negated literal.
- **Unit resolution**: Let C be a clause and ℓ be a literal.

$$\frac{C \cup \{\ell\}, \neg\ell}{C}$$

- **Theorem B**: Let H be a set of Horn clauses. Then H is satisfiable iff $\square$ cannot be derived from S using unit resolution.

Homework: (compulsory, unless marked with *)

H 1.1 A Horn clause with exactly one positive (negative) literal is called a **positive (negative) fact**.

Show; A set of Horn clause with no positive facts is always satisfiable.

H 1.2 Prove Theorem A from the previous slide.

H 1.3 Prove Theorem B from the previous slide.

H 1.4 **HornSAT** is the problem of deciding whether a finite set of Horn clauses is satisfiable.

Show: **HornSAT** can be solved in **quadratic time**.

[Back to overview](#)

Why is **HornSAT** fast?

- When we apply one resolution step of unit resolution to a clause C and a literal ℓ

$$\frac{C \cup \{\ell\}, \neg\ell}{C}$$

the resulting clause C is smaller than the clause $C \cup \{\ell\}$.

- In the general worst case assume that S contains n clauses and every clause has an occurrence of p .
Assume further that half of the clauses of S contain p and half contain a negated p . So $S(n) - p$ contains $\frac{n^2}{2}$ clauses.
- This shows that in the general case the Davis-Putnam Procedure may use an exponential number of resolution steps, whereas in the case of Horn clauses, a quadratic upper bound is assured.

Homework: (compulsory, unless marked with *)

H 1.5* Show: **HornSAT** can be solved in **linear time**.

W. Dowling, and J. Gallier, *Linear-time algorithms for testing the satisfiability of propositional Horn formulae*. Journal of Logic Programming, 3 (1984) 267–284

A. Itai, J. A. Makowsky, *Unification as a complexity measure for logic programming*. The Journal of Logic Programming, 4.2, (1987) 105–117
(Technion Technical Report **1982**)

[Back to overview](#)

2SAT aka Krom-formulas

Another case where the clauses obtained in the Davis-Putnam Procedure do not grow is 2SAT, where the clauses in S contain at most **two** variables.

$$\frac{\{\ell_1, q\}, \{\neg q, \ell_2\}}{\{\ell_1, \ell_2\}}$$

- The Davis-Putnam Procedure solves 2SAT in deterministic polynomial time. $2SAT \in \mathbf{P}$.
- In fact it can be solved in linear time
Even, S.; Itai, A.; Shamir, A.
On the complexity of time table and multi-commodity flow problems,
SIAM Journal on Computing, 5(4): 691-703, (1976)
- It can also be solved in non-deterministic logarithmic space, $2SAT \in \mathbf{NL}$.
This is shown in the book
C. Papadimitriou, Computational Complexity, 1994, Theorems 9.1, 16.2, 16.3.

2SAT Algorithm, I

Instead of the Davis-Putnam Procedure we can use the following to get a more efficient algorithm for 2SAT.

Let S be a set of clauses over the variables $p_1, \dots, p_n$.
We define a directed graph G_S of order $2n$ as follows:

- The vertices $V(G_S)$ are the literals $p_i, \neg p_i$ for $i \in [n]$.
An arc (directed edge) (ℓ_1, ℓ_2) is in $E(G_S)$ if $(\ell_1, \neg \ell_2) \in S$.
- Note that $(\neg p \vee q)$ can be written as $(\neg(\neg\neg p) \vee q)$, $(\neg(\neg\neg p) \vee \neg\neg q)$, or $(\neg p \vee \neg(\neg q))$.
- If $(\neg p, q) \in S$ this gives **two** arcs: from p to q and from $\neg q$ to $\neg p$.
- Similarly, $(p \vee q)$ can be written as $(\neg(\neg p) \vee q)$, $(\neg(\neg p) \vee \neg\neg q)$, or $(p \vee \neg\neg q)$.
- If $(p, q) \in S$ this gives **two** arcs: from $\neg p$ to q and from $\neg q$ to p .

2SAT Algorithm, II

Now we can state Theorem 9.1 from Papadimitriou's book:

Theorem: Let S a set of clauses of size at most 2.

S is unsatisfiable if and only if there is a variable p such that there are paths from p to $\neg p$ and from $\neg p$ to p in G_S .

To check whether 2SAT is false, we **guess** a variable p and use *REACHABILITY* for both $u = p$ and $v = \neg p$ and also for $u = \neg p$ $v = p$.

Problem: *REACHABILITY*

Input: A directed graph G of order n and two vertices $u, v \in V(G)$.

Question: Is there a directed path from u to v ?

2SAT Algorithm, III

Theorem:(Savitch, 1970)

REACHABILITY can be solved in L , i.e., deterministic logarithmic space $Space(O(\log^2 n))$.

Hence, falsifying *2SAT* is in non-deterministic logarithmic space NL , and *2SAT* is in $Co - NL$.

Theorem:(Immerman, 1988, Szelepcsényi 1987) $NL = Co - NL$.

We conclude that $2SAT \in NL$.

Alfred Horn and Melven R. Krom



Alfred Horn (1918-2001)
UCLA



Melven R. Krom (1932(?)-2018)
UC Davis

- Alfred Horn:
On sentences which are true of direct unions of algebras.
J. Symb. Log. 16: 14-21 (1951)
- Melven R. Krom:
The Decision Problem for a Class of First-Order Formulas in Which all Disjunctions are Binary,
Z. Math. Logik, 13(12): 15-20, (1967)
- Melven R. Krom:
The Decision Problem for Formulas in Prenex Conjunctive Normal Form with Binary Disjunctions.
J. Symb. Log. 35(2): 210-216 (1970)

Homework: (continued)

H 1.6 A set of clauses S is **renamable Horn** if there is a set of propositional variables V_0 such that replacing each occurrence of $p_i \in V_0$ in S by $\neg p_i$ transform S into a set of Horn formulas. Show: Checking whether S is renamable Horn can be done in **polynomial time**.

H 1.7* Show: Checking whether S is renamable Horn can be done in **linear time**.

V. Chandru, C.R. Coullard, P.L. Hammer, M. Montaez, and X. Sun, *On renamable Horn and generalized Horn functions*. Annals of Mathematics and Artificial Intelligence 1:14 (2005) 33-47.

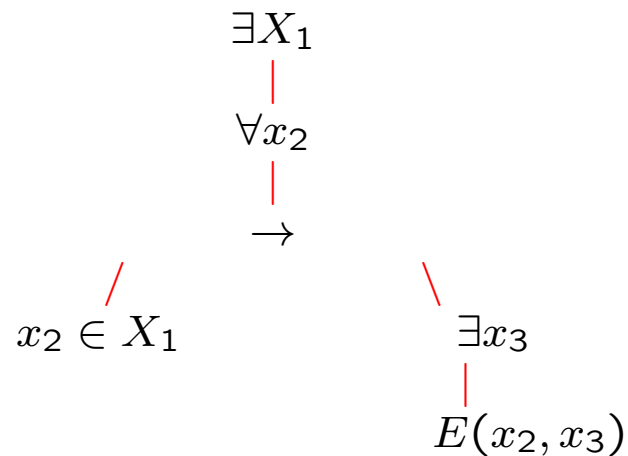
The complexity of Model checking

Upper bounds

Quantifier rank of a formula, I

We write a formula ϕ as a tree:

$$\exists X_1 \forall x_2 (x_2 \in X_1 \rightarrow \exists x_3 E(x_2, x_3))$$



The quantifier rank is biggest number of quantifiers one can find along a path in this tree.

Here it is 3.

Quantifier rank of a formula, II

- For formulas in **prenex normal form** the quantifier rank **equals** the **number of quantifiers**.
- If we **reuse variables**, the quantifier rank can be **smaller** than the number of quantifiers used in prenex normal form.

$$\forall x_1 (\exists x_2 E(x_1, x_2) \wedge \exists x_2 \neg E(x_1, x_2))$$

Quantifier rank **2**

$$\forall x_1 \exists x_2 \exists x_3 (E(x_1, x_2) \wedge \neg E(x_1, x_3))$$

Quantifier rank **3**

[Back to overview](#)

Computing the meaning function $\mathcal{M}(\mathfrak{A}, z, \phi)$

Given

- $\mathfrak{A}$, a τ -structure with $|A| = m$
- $\phi \in \text{SOL}(\tau)$ of length n and quantifier depth q
- z an assignment $z : \text{Var}_{\text{FOL}} \cup \text{Var}_{\text{SOL}} \rightarrow A \cup \bigcup_i \wp(A_i)$.

We want to compute **inductively** the meaning function

$$\mathcal{M}(\mathfrak{A}, z, \phi)$$

and estimate its computational complexity with respect to time and space denoted by $\text{TIME}(\mathfrak{A}, z, \phi)$, $\text{SPACE}(\mathfrak{A}, z, \phi)$.

If ϕ has no free variables we also write $\mathfrak{A} \models \phi$ if $\mathcal{M}(\mathfrak{A}, z, \phi) = 1$.

Computing $\mathcal{M}(\mathfrak{A}, z, \phi)$ for FOL, I

Recall that τ is purely relational and terms t are either constants or variables.

Atomic formulas:

$R(t_1, t_2, \dots, t_r)$ with $R \in \tau$ and $t_1 \approx t_2$.

Takes one step in a random access look-up table.

Takes m^r , resp. m^2 steps for searching the table.

One bit space for the result.

Boolean operations:

$\phi = (\phi_1 \wedge \phi_2)$, $\phi = (\phi_1 \vee \phi_2)$, $\phi = \neg\phi_1$

$TIME(\mathfrak{A}, z, \phi) \leq TIME(\mathfrak{A}, z, \phi_1) + TIME(\mathfrak{A}, z, \phi_2) + 1$

$SPACE(\mathfrak{A}, z, \phi) \leq \max(SPACE(\mathfrak{A}, z, \phi_1), SPACE(\mathfrak{A}, z, \phi_2))$

Computing $\mathcal{M}(\mathfrak{A}, z, \phi)$ for FOL, II

Quantifiers:

$$\phi = \exists x \phi_1(x), \quad \phi = \forall x \phi_1(x)$$

We search the structure for an element, hence

$$TIME(\mathfrak{A}, z, \phi) \leq m \cdot TIME(\mathfrak{A}, z, \phi_1)$$

We can denote location of search in binary, hence

$$SPACE(\mathfrak{A}, z, \phi) \leq \log m \cdot SPACE(\mathfrak{A}, z, \phi_1)$$

Conclusion:

$$TIME(\mathfrak{A}, z, \phi) = O(n \cdot m^q)$$

$$SPACE(\mathfrak{A}, z, \phi) = O(q \cdot \log m)$$

Computing $\mathcal{M}(\mathfrak{A}, z, \phi)$ for FOL, III

We have considered two problems for *FOL*:

- (i) The **combined complexity** of $\mathcal{M}(\mathfrak{A}, z, \phi)$ where both $\mathfrak{A}$ and ϕ are the input.
This is in *PSPACE*.
- (ii) The complexity of $\mathcal{M}(\mathfrak{A}, z, \phi)$ for **fixed** ϕ , where only $\mathfrak{A}$ is the input.
This is in **P** and even in *LOGSPACE* $\subseteq$ **P**.

Back to overview

Computing $\mathcal{M}(\mathfrak{A}, z, \phi)$ for SOL, I

Now we consider *SOL*.

The only change comes from the **second order quantifiers**:

Now search is over **all subsets of A^r** .

This takes time 2^{m^r} .

The characteristic function of these sets has size m^r .

Conclusion:

$$TIME(\mathfrak{A}, z, \phi) = O(n \cdot 2^{q \cdot m^r})$$

$$SPACE(\mathfrak{A}, z, \phi) = O(q \cdot \log m^r) = O(q \cdot r \cdot \log(m))$$

Computing $\mathcal{M}(\mathfrak{A}, z, \phi)$ for SOL, II

We consider two problems for *SOL*:

- (i) The **combined complexity** of $\mathcal{M}(\mathfrak{A}, z, \phi)$ where both $\mathfrak{A}$ and ϕ are the input.

This is in *PSPACE*.

- (ii) The complexity of $\mathcal{M}(\mathfrak{A}, z, \phi)$ for **fixed** ϕ , where only $\mathfrak{A}$ is the input.

This is also in *PSPACE*.

Computing $\mathcal{M}(\mathfrak{A}, z, \phi)$ for SOL, II

We want to use now **non-deterministic machines**.

We denote by $\exists SOL(\tau)$ the set of $SOL(\tau)$ -formulas ψ of the form

$$\psi = \exists X_1 \exists X_2 \dots \exists X_k \phi(X_1, X_2, \dots, X_k)$$

with $\phi \in FOL(\tau \cup \{X_1, X_2, \dots, X_k\})$

Fact:

For fixed $\psi \in \exists SOL$ we have that $\mathcal{M}(\mathfrak{A}, z, \psi)$ is in **NP**, and

for fixed $\psi \in SOL$ we have that $\mathcal{M}(\mathfrak{A}, z, \psi)$ is in **PH**,

the **Polynomial Hierarchy**.

The Polynomial Hierarchy, I

We look at **Oracle Turing Machines** OTM . Let X be a problem and C be a class of problems.

We define

$$\mathbf{P}^X = \{Y : \exists M \text{ accepts } Y \text{ using } X \text{ as oracle} \}$$

$$\mathbf{P}^C = \{Y : \exists M \text{ accepts } Y \text{ using } X \in C \text{ as oracle} \}$$

Here M is a deterministic polynomial time OTM .

Similarly,

$$\mathbf{NP}^X = \{Y : \exists M \text{ accepts } Y \text{ using } X \text{ as oracle} \}$$

$$\mathbf{NP}^C = \{Y : \exists M \text{ accepts } Y \text{ using } X \in C \text{ as oracle} \}$$

Here M is a non-deterministic polynomial time OTM .

The Polynomial Hierarchy, II

We define inductively:

$$\Delta_0\mathbf{P} = \Sigma_0\mathbf{P} = \Pi_0\mathbf{P} = \mathbf{P}$$

and

$$\Delta_{i+1}\mathbf{P} = \mathbf{P}^{\Sigma_i\mathbf{P}}$$

$$\Sigma_{i+1}\mathbf{P} = \mathbf{NP}^{\Sigma_i\mathbf{P}}$$

$$\Pi_{i+1}\mathbf{P} = \mathbf{CoNP}^{\Sigma_i\mathbf{P}}$$

Finally,

$$\mathbf{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i\mathbf{P}$$

Note that $\mathbf{PH} \subseteq PSPACE$ and
 $\mathbf{P} = \mathbf{NP}$ iff $\mathbf{P} = \mathbf{PH}$.

Back to overview

Horn formulas, I

A **propositional Horn clause** is a formula of the form

$$\neg p_1 \vee \neg p_2 \vee \dots \vee \neg p_m \vee q$$

with atmost one variable unnegated.

Equivalently, we can write

$$(p_1 \wedge p_2 \wedge \dots \wedge p_m \rightarrow q)$$

$m = 0$ gives $\text{true} \rightarrow q$ and the absence of q gives

$$(p_1 \wedge p_2 \wedge \dots \wedge p_m \rightarrow \text{false})$$

or

$$\neg(p_1 \wedge p_2 \wedge \dots \wedge p_m)$$

A *FOL* Horn clause is obtained by replacing variables by atomic formulas.

Horn formulas, II

The **size** $s(C)$ **of a clause** C is the number of variables occurring in Σ .

The **size** $s(\Sigma)$ **of set of clauses** Σ **is defined as** $\sum_{C \in \Sigma} s(C)$

SAT is the problem of deciding whether a set Σ of clauses with n variables of size m is satisfiable.

Theorem:(S. Cook and L. Levin)

SAT can be solved in $TIME(2^n \cdot m)$ and is **NP**-complete.

$HORNSAT$ is like SAT but with Σ a set of Horn clauses.

$2SAT$ is like SAT with all clauses of size at most 2.

Theorem: $HORNSAT$ is in **P**.

Theorem: $2SAT$ is in **NL**.

A special case of $\exists\text{SOL}$, I

We want to test whether a formula $\Psi \in \exists\text{SOL}(\tau)$ is satisfied in a τ -structure $\mathfrak{A}$ where

$$\Psi = \exists X_1 \exists X_2 \dots \exists X_j \forall x_1 \forall x_2 \dots \forall x_k \bigwedge_{\ell=1}^n \Phi_\ell(\bar{x}, \bar{y})$$

and $\Phi(\bar{x}, \bar{y})$ is a quantifier free formula.

- We reduce the problem to a formula $\bar{\Psi}$ of propositional logic where $\bar{\Psi}$ has size polynomial in the size of Φ .
- We show that $\bar{\Psi}$ is satisfiable iff there is an assignment $z : Var \rightarrow A$ such that $\mathcal{M}(\mathfrak{A}, z, \Psi) = 1$
- Now the complexity upper bound of checking whether $\mathcal{M}(\mathfrak{A}, z, \psi) = 1$ only depends on the complexity of checking whether $\bar{\Psi}$ is satisfiable.

A special case of $\exists\text{SOL}$, II

For simplicity let

$$\Psi = \exists X \forall x_1 \forall x_2 \dots \forall x_k \bigwedge_{\ell=1}^n \Phi_\ell$$

a τ_{graphs} -formula with X r -ary.

So each Φ_ℓ consists of atomic or negated atomic formulas $x_i \approx x_j$, $E(x_i, x_j)$ or $X(x_{i_1}, \dots, x_{i_r})$.

Let $\mathfrak{A}$ be a structure with elements $a_1, \dots, a_n$.

There are n^k many assignments for the bound variables x_i .

Let h be the length of $\Phi = \bigwedge_{\ell} \Phi_\ell$.

A special case of $\exists$ SOL, III

Let $z_0 = z_0(\bar{y})$ a fixed assignment of elements of A to the free variables of Ψ , and let $z = z(\bar{x})$ vary over all assignment to the bound variables of Ψ .

We now form the formula

$$\bigwedge_z \text{subst}(\Phi, z, z_0)$$

This formula has exactly $h \cdot n^k$ many literals, which is a polynomial in n .

In $\mathfrak{A}$ each atomic formula $E(a_i, a_j)$ or $a_i \approx a_j$ is true or false, so we can replace them by **true** or **false** respectively.

We replace each $X(a_{i_1}, \dots, a_{i_r})$ by a propositional variable $p_{a_{i_1}, \dots, a_{i_r}}$.

We obtain in this way a propositional formula $\bar{\Psi}$.

We can also turn it into **reduced** form in linear time.

A special case of $\exists$ SOL, IV

Claim 1:

If $\mathfrak{A} \models \Psi$ then $\bar{\Psi}$ is satisfiable.

Proof: Assume $\mathfrak{A} \models \Psi$.

Then there is $U \subset A^r$ such that

$$\mathfrak{A}, U \models \forall x_1 \forall x_2 \dots \forall x_k \bigwedge_{\ell=1}^n \Phi_\ell$$

.

We now define an assignment

$$z(p_{\bar{a}}) = \begin{cases} 1 & \text{if } \bar{a} \in U \\ 0 & \text{if } \bar{a} \notin U \end{cases}$$

Exercise: Show that this z makes $\bar{\Psi}$ true.

A special case of $\exists$ SOL, $\forall$

Claim 2:

If $\bar{\Psi}$ is satisfiable then $\mathfrak{A} \models \Psi$.

Proof: Assume z is an assignment which makes $\bar{\Psi}$ true.

We define an interpretation U for X by

$$\bar{a} \in U \text{ iff } z(p_{\bar{a}}) = 1.$$

Exercise: Show that

$$\mathfrak{A}, U \models \forall x_1 \forall x_2 \dots \forall x_k \bigwedge_{\ell=1}^n \Phi_{\ell}.$$

A special case of $\exists$ SOL, $\forall$

We have shown:

Theorem:(Grädel, 1991)

Given a τ -structure $\mathfrak{A}$ and a formula ψ of the form

$$\psi = \exists X_1 \exists X_2 \dots \exists X_j \forall x_1 \forall x_2 \dots \forall x_k \bigwedge_{\ell=1}^n \Phi_\ell(\bar{x}, \bar{y})$$

checking whether

$$\mathfrak{A} \models \psi$$

is in

- **NP** in general;
- **P** if all the $\Phi_\ell(\bar{x}, \bar{y})$ are Horn clauses;
- **NL** if all the $\Phi_\ell(\bar{x}, \bar{y})$ are Krom clauses.

A special case of $\exists$ SOL, VI

Proof of Theorem:

- The construction of $\bar{\Psi}$ from Ψ is done in polynomial time.
- The size of $\bar{\Psi}$ is polynomial in the size of Ψ .
- We check the satisfiability $\mathfrak{A} \models \Psi$. using
 SAT $\in$ **NP**;
 HORN SAT $\in$ **P**;
 and *2SAT* $\in$ **NL**.

[Back to overview](#)

Definability and Complexity, I

Let K be a class of finite τ -structures.

Let $\mathcal{L}(\tau) \subseteq SOL(\tau)$.

Typically $\mathcal{L}(\tau)$ is one of $Krom\exists SOL(\tau)$, $Horn\exists SOL(\tau)$, $\exists SOL(\tau)$, $SOL(\tau)$, $MSOL(\tau)$,

K is **definable in** $\mathcal{L}(\tau)$ if there exists $\Psi \in \mathcal{L}(\tau)$ such that

$$\mathfrak{A} \in K \text{ iff } \mathfrak{A} \models \Psi$$

Let $\mathbf{C}$ be a complexity class.

typically $LOGSPACE$, NL , P , NP , PH , $PSPACE$

K is in $\mathbf{C}$ iff the problem $\mathfrak{A} \in K$
can be decided with the resources allowed in $\mathbf{C}$.

Definability and Complexity, II

We have shown:

- If K is definable in FOL
then $K \in LOGSPACE$.
- If K is definable in $Krom\exists SOL$
then $K \in \mathbf{NL}$.
- If K is definable in $Horn\exists SOL$
then $K \in \mathbf{P}$.
- If K is definable in $\exists SOL$
then $K \in \mathbf{NP}$.
- If K is definable in SOL
then $K \in \mathbf{PH}$.

Definability and Complexity, III

We will show in the sequel for **ordered structures**

- (Grädel) If $K \in \mathbf{NL}$, then K is definable in $Krom\exists SOL$.
- (Grädel) If $K \in \mathbf{P}$, then K is definable in $Horn\exists SOL$.

For **arbitrary structures** we have

- (Fagin, Christen) If $K \in \mathbf{NP}$, then K is definable in $\exists SOL$.
- (Meyer and Stockmeyer) If $K \in \mathbf{PH}$, then K is definable in SOL .

LOGSPACE

What about *FOL*-definability and *LOGSPACE*?

Exercise:

Show that the set of words of even size is not *FOL*-definable.

Exercise:

Show that the set of words of even size is in *LOGSPACE*.

Exercise:

Conclude that *FOL*-definability is weaker than decidability in *LOGSPACE*.

Questions:

Which logic corresponds to *LOGSPACE*?

FOL+deterministic transitive closure

Which complexity class corresponds to *FOL*?

The circuit complexity class $\mathbf{AC}_0$.

Session 4 continues with Part 3

We introduced the Ehrenfecht-Fraïssé games
on the white board and played it a bit.

This will be repeated in Session 5.

End of Session 4
