Logic, automata and languages – Master 2, Univ. Bordeaux, 2016/17

Contents

- Anca Muscholl: Logic on graphs and trees.
- Marc Zeitoun: Logic and algebra on words, Presburger logic.

References:

- L. Libkin, Elements of finite model theory. Springer 2004.
- W. Thomas, Languages, automata and logic. Chapter in *Handbook of Formal Languages*, Springer 1997.

Logic in databases

- In relational databases we use the relational calculus (cf. SQL): first-order predicate calculus over relational structures.
- Query: select elements of a database with a given property.
- Query equivalence: determine if two SQL queries return the same results.

Logic in program verification

- A program is modelled by a transition system (Kripke structure), a property (specification) in some logic, e.g. temporal logic, first-order logic, etc.
- Proof systems (cf. Coq): semi-automatic way to show the validity of program properties.
- Model-checking method (cf. Spin, nu-SMV, ...): check whether all executions of the program satisfy a given specification. Fully automatic procedure.

Related topics

- Descriptive complexity theory. Deals with expressivity of logics. Logics that capture complexity classes.
- Algorithmic complexity. Deals with the complexity of reasoning tasks such as satisfiability and model-checking. Fine analysis of parameters that influence the complexity.
- Game theory. Game formulation of problems such as model-checking.
- Automata theory. Tight relation to logics in terms of expressivity.

Syntax

Vocabulary σ: constant symbols $(c_1, c_2, \ldots)$, predicates $(P_1, P_2, \ldots)$, functions $(f_1, f_2, \ldots)$. Each relational/functional symbol has an associated arity $(\rho(P), \rho(f))$.
Terms and formulas over σ are defined inductively:

- A term is either a variable, or a constant, or an expression $f(t_1, \ldots, t_n)$, where t_i are terms.
- A formula has one of the forms: $t_1 = t_2$ (t_i terms), $P(t_1, \ldots, t_n)$ (P is k-ary relation, t_i terms), $\varphi_1 \vee \varphi_2$, $\neg\varphi$, $\exists x \,.\, \phi$.

Free variables of a term t or formula φ: $\mathrm{free}(t)$, $\mathrm{free}(\varphi)$.

Semantics

- σ-structure (or model): $\mathfrak{A} = \langle A, \{c_i^{\mathfrak{A}}\}, \{P_i^{\mathfrak{A}}\}, \{f_i^{\mathfrak{A}}\}\rangle$ with universe A, $c_i^{\mathfrak{A}} \in A$, $P_i^{\mathfrak{A}} \subseteq A^{\rho(P_i)}$, $f_i^{\mathfrak{A}} : A^{\rho(f_i)} \to A$.
- Each term $t(x_1, \ldots, x_n)$ defines, for each tuple of values $\vec{a} = (a_1, \ldots, a_n) \in A^n$, a value $t^{\mathfrak{A}}(a_1, \ldots, a_n)$.
- For every formula $\varphi(x_1, \ldots, x_n)$, for every tuple of values $\vec{a} = (a_1, \ldots, a_n) \in A^n$, define the satisfaction relation $\mathfrak{A} \models \varphi(a_1, \ldots, a_n)$.

Examples

- Vocabulary σ: $(P_a)_{a\in\Sigma}, \mathsf{succ}, <$. Each P_a is a unary predicate, and $\mathsf{succ}, <$ are binary predicates.

  Finite words as σ-structures: $w = \langle [n] := \{1, \ldots, n\}, (P_a)_{a\in\Sigma}, \mathsf{succ}, < \rangle$ with $P_a = \{k \in [n] \mid \text{ the } k\text{-th letter is } a\}$.
  Infinite words $w = \langle \mathbb{N}, (P_a)_{a\in\Sigma}, \mathsf{succ}, < \rangle$.

- Vocabulary σ: $(P_a)_{a\in\Sigma}, \mathsf{left}, \mathsf{right}, <$. Predicates $\mathsf{left}, \mathsf{right}, <$ are binary.

  Finite labelled binary trees as σ-structures: $t = \langle D, (P_a)_{a\in\Sigma}, \mathsf{left}, \mathsf{right}, < \rangle$.
  Domain D is a (finite) prefix-closed subset of $\{0,1\}^*$, $\mathsf{left}, \mathsf{right} : D \to D$ with $\mathsf{left}(x) = x0$, $\mathsf{right}(x) = x1$ and $x < y$ if x is prefix of y.
  Infinite labelled binary trees: $D = \{0,1\}^*$.

- Vocabulary σ: $(P_a)_{a\in\Sigma}$, E binary predicate.

  Labelled graphs as σ-structures: P_a node labelings, E edge relation.

Model-checking problem

Given a model $\mathfrak{A}$ and a formula φ: does $\mathfrak{A} \models \varphi$ hold?

Satisfiability problem

Given a formula φ, is there some model $\mathfrak{A}$ with $\mathfrak{A} \models \varphi$?

Expressivity

Given a logic $\mathcal{L}$ and a formula φ, is there some equivalent formula φ' from $\mathcal{L}$?

Remark

We are interested in finite models ("finite model theory").

Plan

First-order logic (FO)

Variables are interpreted over elements of the structure.

Expressivity

As we will see, first-order logic is rather weak. So it is interesting to extend it, e.g. to second-order logic, or by adding fixpoints, etc

Second-order logic (SO), monadic second-order logic (MSO)

- SO: add second-order variables, interpreted as relations over the universe A.
- MSO logic: restriction of SO where second-order variables are subsets of A (relations of arity 1).

Examples

- Words containing factor ab (FO formula):

$$\exists x\,(a(x) \wedge b(x+1))$$

  Here: $a(x)$ instead of $P_a(x)$, and $x+1$ shorthand for succ:

$$\exists x\,\exists y\,(a(x) \wedge b(y) \wedge \mathsf{succ}(x,y))$$

- Words of even length (SO formula):

$$\exists X_0\,\exists X_1 \qquad \big(\mathsf{Partition}(X_0,X_1) \wedge \mathsf{min} \in X_0 \wedge \mathsf{max} \in X_1 \wedge \forall x\,(x \in X_0 \Leftrightarrow x+1 \in X_1)\big)$$

- Graphs of minimal out-degree 2 (FO formula):

$$\forall x\,\exists y\,\exists z\,(y \neq z \wedge (x,y) \in E \wedge (x,z) \in E)$$

- (Dis)connected graphs (SO formula):

$$\exists X \exists Y \qquad \big(X \neq \emptyset \wedge Y \neq \emptyset \wedge \mathsf{Partition}(X,Y) \wedge \forall x \forall y\,((x,y) \in E \rightarrow ((x \in X \wedge y \in X) \vee (x \in Y \wedge y \in Y)))\big)$$

Model-checking problem

- Given structure $\mathfrak{A}$ and (closed) formula φ, check if $\mathfrak{A} \models \varphi$.
- Query evaluation problem: given structure $\mathfrak{A}$ and formula $\varphi(\vec{x})$ with k free variables, compute $\{\vec{a} \in A^k \mid \mathfrak{A} \models \varphi(\vec{a})\}$.
- Measuring complexity: combined complexity if $\mathfrak{A}, \varphi$ both given as input, and data complexity if φ is fixed.

Model-checking game

- Two players: Verifier (or player P_0) and Falsifier (player P_1). Verifier tries to establish that $\mathfrak{A} \models \varphi(\vec{a})$, Falsifier has the opposite goal.
- Formula $\varphi(\vec{x})$ is in negation normal form (negations only at atoms).
- Game positions: (ψ, ρ) with ψ subformula of φ and $\rho : \text{free}(\psi) \to A$. Write $\psi(\vec{b})$ instead of (ψ, ρ), where ρ maps free(ψ) to $\vec{b}$.
- Initial game position: $\varphi(\vec{a})$.

Model-checking game

- Verifier moves at positions $\psi(\vec{b})$ where either $\psi = \psi_1 \vee \psi_2$ or $\psi = \exists x\,.\chi$. From $\psi = \psi_1 \vee \psi_2$ she goes either to ψ_1 or to ψ_2. From $\exists x\,.\chi(\vec{b})$ she chooses $a \in A$ and goes to $\chi(\vec{b}, a)$.
- Falsifier moves at positions $\psi(\vec{b})$ where either $\psi = \psi_1 \wedge \psi_2$ or $\psi = \forall x\,.\chi$, making the dual moves.
- Positions corresponding to (negated) atoms $\psi(\vec{b})$ are final: they are winning for Verifier iff $\mathfrak{A} \models \psi(\vec{b})$.

Thm.

Verifier has a winning strategy iff $\mathfrak{A} \models \varphi(\vec{a})$.

Algorithm

The model-checking game can be solved in time linear in the size of the game arena, thus in exponential time (combined complexity: polynomial space, data complexity: polynomial time).

Method for exploring limits of expressivity of various logics.

EF game on finite words over $\Sigma = \{c\}$ for FO($<$).

FO($<$): built from the atomic formulas $x < y$, $x = y$, $\min(x)$, $\max(x)$, using $\vee, \wedge, \neg, \exists, \forall$.

- Given finite words w_0, w_1. Let 0 be the first position of w_i and N_i the last position.
- Players: Spoiler and Duplicator.
- Play k rounds. In each round, Spoiler picks a position either in w_0 or in w_1; Duplicator answers by choosing a position in the other word.
- Let $a_1, \ldots, a_k$ (resp. $b_1, \ldots, b_k$) be the positions chosen in w_0 (resp. w_1).
- Duplicator wins if the mapping $\{a_i \to b_i \mid 1 \leq i \leq k\}$ is a partial isomorphism: for all $1 \leq i, j \leq k$,
  1. $a_i = 0$ iff $b_i = 0$, and $a_i = N_0$ iff $b_i = N_1$
  2. $a_i = a_j$ iff $b_i = b_j$
  3. $a_i < a_j$ iff $b_i < b_j$

Prop.

If $\min(|w_0|, |w_1|) > 2^k$ then Duplicator wins the k-round EF game on w_0, w_1.

Proof

Duplicator plays in such a way that after i rounds the following holds for all $0 \leq j, l \leq i+1$ (with $a_0 = b_0 = 0$, $a_{i+1} = N_0$, $b_{i+1} = N_1$):

- $a_j < a_l$ iff $b_j < b_l$,
- $|a_j - a_l| \geq 2^{k-i}$ iff $|b_j - b_l| \geq 2^{k-i}$,
- if $|a_j - a_l| < 2^{k-i}$ then $|a_j - a_l| = |b_j - b_l|$.

Example

Words $w_0 = cccc$, $w_1 = ccc$: Spoiler wins the 2-round game. Spoiler starts on the left word;

$$c\ c\ c\ c \qquad\qquad c\ c\ c$$

Proof

Duplicator plays in such a way that after i rounds the following holds for all $0 \leq j, l \leq i+1$ (with $a_0 = b_0 = 0$, $a_{i+1} = N_0$, $b_{i+1} = N_1$):

- $a_j < a_l$ iff $b_j < b_l$,
- $|a_j - a_l| \geq 2^{k-i}$ iff $|b_j - b_l| \geq 2^{k-i}$,
- if $|a_j - a_l| < 2^{k-i}$ then $|a_j - a_l| = |b_j - b_l|$.

Example

Words $w_0 = cccc$, $w_1 = ccc$: Spoiler wins the 2-round game. Spoiler starts on the left word;

$$c\ c\ c\ c \qquad\qquad c\ c\ c$$

Proof

Duplicator plays in such a way that after i rounds the following holds for all $0 \leq j, l \leq i+1$ (with $a_0 = b_0 = 0$, $a_{i+1} = N_0$, $b_{i+1} = N_1$):

- $a_j < a_l$ iff $b_j < b_l$,
- $|a_j - a_l| \geq 2^{k-i}$ iff $|b_j - b_l| \geq 2^{k-i}$,
- if $|a_j - a_l| < 2^{k-i}$ then $|a_j - a_l| = |b_j - b_l|$.

Example

Words $w_0 = cccc$, $w_1 = ccc$: Spoiler wins the 2-round game. Spoiler starts on the left word; Duplicator responds on the right one.

$$c\ c\ c\ c \qquad\qquad c\ c\ c$$

Proof

Duplicator plays in such a way that after i rounds the following holds for all $0 \leq j, l \leq i+1$ (with $a_0 = b_0 = 0$, $a_{i+1} = N_0$, $b_{i+1} = N_1$):

- $a_j < a_l$ iff $b_j < b_l$,
- $|a_j - a_l| \geq 2^{k-i}$ iff $|b_j - b_l| \geq 2^{k-i}$,
- if $|a_j - a_l| < 2^{k-i}$ then $|a_j - a_l| = |b_j - b_l|$.

Example

Words $w_0 = cccc$, $w_1 = ccc$: Spoiler wins the 2-round game. Spoiler starts on the left word; Duplicator responds on the right one.

$$c\ c\ c\ c \qquad\qquad c\ c\ c$$

Spoiler replays $a_2 = 1$ on the left word.

Proof

Duplicator plays in such a way that after i rounds the following holds for all $0 \leq j, l \leq i+1$ (with $a_0 = b_0 = 0$, $a_{i+1} = N_0$, $b_{i+1} = N_1$):

- $a_j < a_l$ iff $b_j < b_l$,
- $|a_j - a_l| \geq 2^{k-i}$ iff $|b_j - b_l| \geq 2^{k-i}$,
- if $|a_j - a_l| < 2^{k-i}$ then $|a_j - a_l| = |b_j - b_l|$.

Example

Words $w_0 = cccc$, $w_1 = ccc$: Spoiler wins the 2-round game. Spoiler starts on the left word; Duplicator responds on the right one.

$$c\ c\ c\ c \qquad\qquad c\ c\ c$$

Spoiler replays $a_2 = 1$ on the left word. If Duplicator chooses $b_2 = 0$ she looses because $a_2 \neq 0$. If Duplicator chooses $b_2 = 1$ she looses because $a_2 \neq a_1$.

Proof

Duplicator plays in such a way that after i rounds the following holds for all $0 \leq j, l \leq i+1$ (with $a_0 = b_0 = 0$, $a_{i+1} = N_0$, $b_{i+1} = N_1$):

- $a_j < a_l$ iff $b_j < b_l$,
- $|a_j - a_l| \geq 2^{k-i}$ iff $|b_j - b_l| \geq 2^{k-i}$,
- if $|a_j - a_l| < 2^{k-i}$ then $|a_j - a_l| = |b_j - b_l|$.

Example

Words $w_0 = cccc$, $w_1 = ccc$: Spoiler wins the 2-round game. Spoiler starts on the left word; Duplicator responds on the right one.

$$c\ c\ c\ c \qquad\qquad c\ c\ c$$

Spoiler replays $a_2 = 1$ on the left word. If Duplicator chooses $b_2 = 0$ she looses because $a_2 \neq 0$. If Duplicator chooses $b_2 = 1$ she looses because $a_2 \neq a_1$.

Spoiler looses in 2 rounds on $w_0 = c^5, w_1 = c^6$.

Other variants

Each EF-game depends on the logic under consideration. If we add to FO$(<)$ e.g. the predicates P_a $(a \in \Sigma)$ and succ, then we add also (for all i, j):

4. The labels of positions a_i, b_i, are the same.
5. $|a_i - a_j| = 1$ iff $|b_i - b_j| = 1$

Examples

- Let $w_0 = aabaacaa$, $w_1 = aacaabaa$. Spoiler wins the 2-round game for FO$(<)$ by choosing the b, c positions in w_0. Note: w_0, w_1 are distinguished by the FO$(<)$ formula

$$\exists x \, \exists y \, .(b(x) \wedge c(y) \wedge x < y)$$

- Let $w_0 = aabaacaa$, $w_1 = aacaabaa$ and consider the 2-round game for FO(succ) (no linear order). In this game, Duplicator wins: if Spoiler chooses the first/last position, Duplicator does the same. And if Spoiler chooses b, c or one of their neighbors, Duplicator does the same again. But 3 rounds are won by Spoiler (assuming that we have $\min$ or $\max$ in our vocabulary). A FO(succ) formula distinguishing w_0, w_1:

$$\exists x \, \exists y \, \exists z \, .(\mathrm{succ}(x, y) \wedge \mathrm{succ}(y, z) \wedge \max(z) \wedge c(x))$$

EF game for graphs

- Given: 2 graphs G_0, G_1; $G_i = (V_i, E_i)$.
- Players: Spoiler and Duplicator.
- k rounds. In each round, Spoiler chooses a graph and a node in that graph. Duplicator answers by choosing a node in the other graph. Let $a_1, \ldots, a_k$ (resp. $b_1, \ldots, b_k$) be the nodes chosen in G_0 (resp. G_1).
- Duplicator wins if for all $1 \leq i, j \leq k$: (1) $a_i = a_j$ iff $b_i = b_j$ and (2) $(a_i, a_j) \in E_0$ iff $(b_i, b_j) \in E_1$.

EF game for graphs

- Given: 2 graphs G_0, G_1; $G_i = (V_i, E_i)$.
- Players: Spoiler and Duplicator.
- k rounds. In each round, Spoiler chooses a graph and a node in that graph. Duplicator answers by choosing a node in the other graph. Let $a_1, \ldots, a_k$ (resp. $b_1, \ldots, b_k$) be the nodes chosen in G_0 (resp. G_1).
- Duplicator wins if for all $1 \leq i, j \leq k$: (1) $a_i = a_j$ iff $b_i = b_j$ and (2) $(a_i, a_j) \in E_0$ iff $(b_i, b_j) \in E_1$.

Quantifier rank

Measures the nesting of quantifiers in a FO formula. Denote by FO$[k]$ the set of FO-formulas of quantifier rank at most k.

Examples: $A(x) \vee B(y)$ is an FO$[0]$ formula,
$\exists x \forall y (A(x) \rightarrow (B(y) \wedge (x, y) \in E))$ is an FO$[2]$ formula.

Notations

The formula φ below and the vectors of nodes $\vec{a}, \vec{b}$ are such that the number of free variables of φ equals the length of $\vec{a}$ and the length of $\vec{b}$.

- Assume that $\vec{a}$ ($\vec{b}$, resp.) is a vector of nodes in G_0 (G_1, resp.).
  Write $(G_0, \vec{a}) \equiv_k (G_1, \vec{b})$ if for all formulas $\varphi \in \mathsf{FO}[k]$:

$$G_0 \models \varphi(\vec{a}) \quad \Leftrightarrow \quad G_1 \models \varphi(\vec{b})$$

- EF-game played on $(G_0, \vec{a})$ and $(G_1, \vec{b})$: as before, with predefined values $\vec{a}, \vec{b}$.

Notations

The formula φ below and the vectors of nodes $\vec{a}, \vec{b}$ are such that the number of free variables of φ equals the length of $\vec{a}$ and the length of $\vec{b}$.

- Assume that $\vec{a}$ ($\vec{b}$, resp.) is a vector of nodes in G_0 (G_1, resp.). Write $(G_0, \vec{a}) \equiv_k (G_1, \vec{b})$ if for all formulas $\varphi \in \mathsf{FO}[k]$:

$$G_0 \vDash \varphi(\vec{a}) \quad \Leftrightarrow \quad G_1 \vDash \varphi(\vec{b})$$

- EF-game played on $(G_0, \vec{a})$ and $(G_1, \vec{b})$: as before, with predefined values $\vec{a}, \vec{b}$.

Thm. (Ehrenfeucht-Fraïssé)

The following is equivalent:

1. $G_0 \equiv_k G_1$
2. Duplicator wins the k-round EF-game on G_0, G_1.

Notations

The formula φ below and the vectors of nodes $\vec{a}, \vec{b}$ are such that the number of free variables of φ equals the length of $\vec{a}$ and the length of $\vec{b}$.

- Assume that $\vec{a}$ ($\vec{b}$, resp.) is a vector of nodes in G_0 (G_1, resp.).
  Write $(G_0, \vec{a}) \equiv_k (G_1, \vec{b})$ if for all formulas $\varphi \in \mathsf{FO}[k]$:

$$G_0 \models \varphi(\vec{a}) \quad \Leftrightarrow \quad G_1 \models \varphi(\vec{b})$$

- EF-game played on $(G_0, \vec{a})$ and $(G_1, \vec{b})$: as before, with predefined values $\vec{a}, \vec{b}$.

Thm. (Ehrenfeucht-Fraïssé)

The following is equivalent:

1. $(G_0, \vec{a}) \equiv_k (G_1, \vec{b})$
2. Duplicator wins the k-round EF-game on $(G_0, \vec{a}), (G_1, \vec{b})$.

How Duplicator wins k rounds

- Partial isomorphism from $(G_0, \vec{c}), (G_1, \vec{d})$: each c_i maps to the corresponding d_i, and the mapping respects all atomic relations (labels, edges etc).
- Duplicator wins k rounds on $(G_0, \vec{a}), (G_1, \vec{b})$ iff there exist non-empty sets $I_k, I_{k-1}, \ldots, I_0$ of partial isomorphisms s.t.:
  1. all partial isomorphisms extend $\vec{a} \mapsto \vec{b}$,
  2. (forward property) for all $F \in I_j$, for every node a in G_0 there exists a node b in G_1 s.t. $F \cup \{a \mapsto b\} \in I_{j-1}$.
  3. (backward property) for all $F \in I_j$, for every node b in G_1 there exists a node a in G_0 s.t. $F \cup \{a \mapsto b\} \in I_{j-1}$.

Proof

1. If Duplicator wins k rounds on $(G_0, \vec{a}), (G_1, \vec{b})$, then it follows by induction (on k) that $(G_0, \vec{a}) \equiv_k (G_1, \vec{b})$.
2. Fix $(G_0, \vec{a})$. We define a FO$[k]$ formula $\varphi_{G_0,\vec{a}}$ that holds for exactly those $(G_1, \vec{b})$ such that Duplicator wins k rounds on $(G_0, \vec{a}), (G_1, \vec{b})$:
   - $\varphi^0_{G_0,\vec{a}}(\vec{x})$ is the conjunction of all atoms $\psi(\vec{x})$ where $(G_0, \vec{a}) \models \psi(\vec{x})$, and all negated atoms $\neg\psi(\vec{x})$ where $(G_0, \vec{a}) \not\models \psi(\vec{x})$.
   - $\varphi^{j+1}_{G_0,\vec{a}}(\vec{x}) = \bigwedge_{c \in V(G_0)} \exists y \,.\varphi^j_{G_0,\vec{a},c}(\vec{x}, y) \wedge \forall y \,.\bigvee_{d \in V(G_0)} \varphi^j_{G_0,\vec{a},d}(\vec{x}, y)$

$\{c^n \mid n$ is even$\}$ is not FO-definable

Apply EF and recall that Duplicator wins on c^{2^k}, c^{2^k+1}.

Connectivity

Deduce that "graph connectivity" is not expressible in FO(E):

Given a word w over $\Sigma = \{c\}$ "build" a directed graph $G(w)$ as follows.

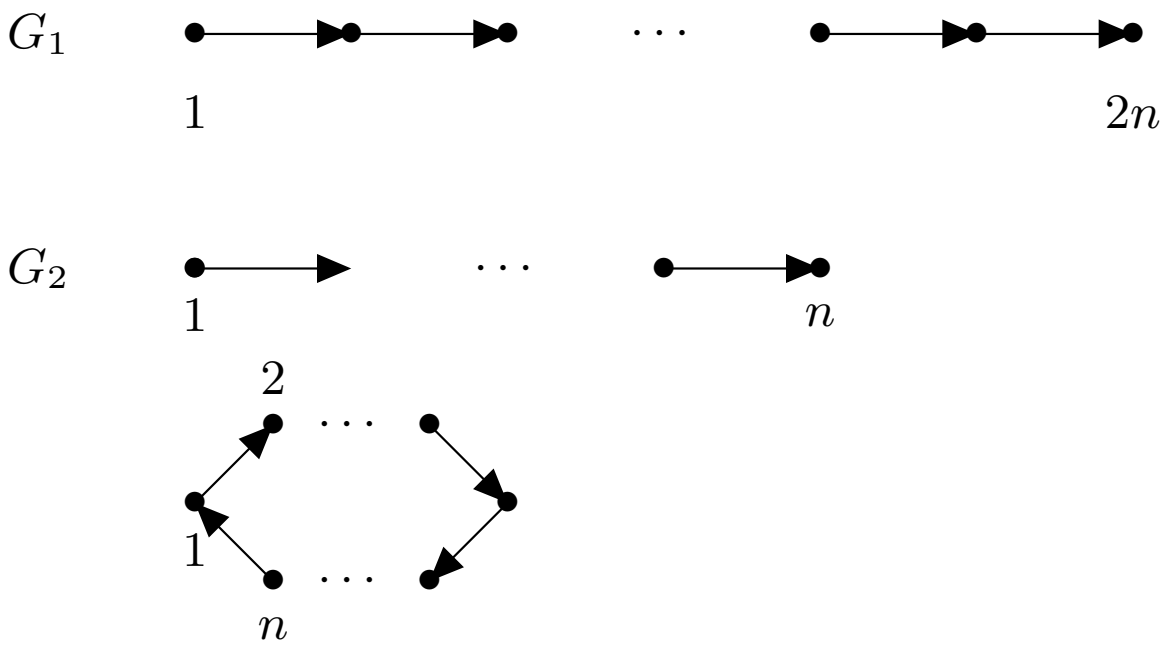
- The set of vertices is the set $\{0, \ldots, |w| - 1\}$ of positions in w.
- There is en edge from i to j if
  - either $j = i + 2$, or
  - $i = |w| - 2$, $j = 0$, or
  - $i = |w| - 1$, $j = 1$ holds.

Observe that $|w|$ is odd iff $G(w)$ is connected.
Suppose that connectivity is expressed by an FO(E) formula φ. The formula obtained from φ by replacing each atomic formula $E(x, y)$ by

$$y = x + 2 \vee (x = \max -1 \wedge y = \min) \vee (x = \max \wedge y = \min +1)$$

says that $|w|$ is odd. Thus, we get a contradiction.

Example: FO$[E]$ cannot express "graph acyclicity"



EF-game

- $a_1, \ldots, a_k$ resp. $b_1, \ldots, b_k$: nodes chosen by Spoiler and Duplicator.
- $d(c, c')$ = distance between nodes c, c':

$$d(c, c') \in \mathbb{N} \cup \{\infty\}.$$

- Invariant after i rounds:
  1. $d(a_j, a_l) \leq 2^{k-i}$ implies $d(b_j, b_l) = d(a_j, a_l)$.
  2. $d(a_j, a_l) > 2^{k-i}$ implies $d(b_j, b_l) > 2^{k-i}$.
- The invariant can be preserved if n is sufficiently large (compared to k).

Local neighborhoods

Consider only graphs of bounded degree, say d.

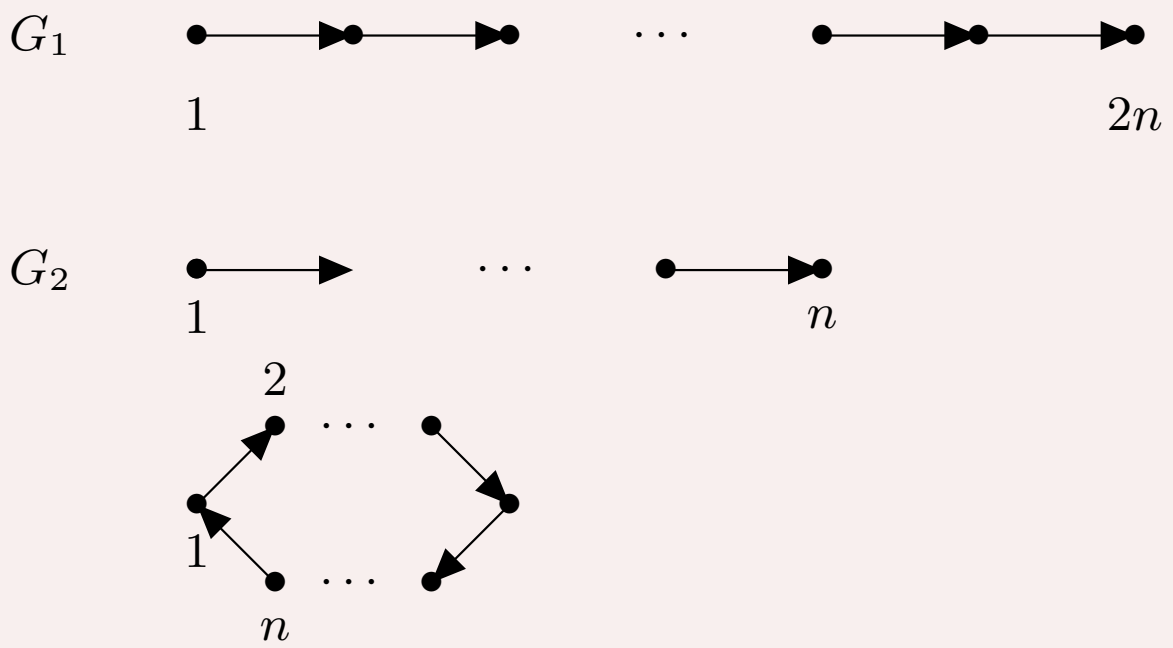
- Let $G = (V, E)$, $v \in V$, and r an integer. The neighborhood $B_r^G(v)$ around v of radius r is the subgraph induced by all w that are connected to v via a path in $E \cup E^{-1}$ of length at most r.

  $B_0^G(v) = \{v\}$, $B_1^G(v) = \{w \mid (u, v) \in E \text{ or } (v, u) \in E\} \cup \{v\}$, ...
- Since the degree is bounded, the number of different $B_r^G(v)$ is finite. We refer to each $B_r^G(v)$ up to isomorphism, as a type.
- For a type t, let $\mathrm{occ}(G, t)$ be the number of occurrences of type t neighborhoods in G.
- Fix some integers r, m and let G, G' be graphs. We define $G \sim_{r,m} G'$ if for every type t of neighborhoods of radius r, either both $\mathrm{occ}(G, t)$ and $\mathrm{occ}(G', t)$ are larger than m, or they are equal.

Hanf-locality (Hanf '65)

For every k there exist r, m such that for any graphs G, G' of degree at most d, we have: $G \sim_{r,m} G'$ implies $G \equiv_k G'$.
Equivalent formulation: any FO-definable set of graphs of bounded degree is locally threshold testable.

Connectivity



For $r = 1$: 3 types (source/target/inner nodes). Both G_1, G_2 have the same $\mathrm{occ}(G_i, t)$, for every type t. Thus, no FO$[E]$-formula of quantifier rank 1 can distinguish them. Similarly for $r > 1$ (for large enough n).

Prop.

Over words, every formula of FO[succ] is equivalent to a disjunction of conjunctions of properties of the following form:

- the word w starts with prefix p,
- the word w ends with suffix s,
- the factor f occurs in w at least (at most) m times.

Proof

We apply Hanf's theorem ($d = 2$). Let φ be a formula of FO[succ] of quantifier rank k. There exist r, m such that $L(\varphi)$ is a union of $\sim_{r,m}$-classes.
But $w \sim_{r,m} w'$ is equivalent to a conjunction of conditions as above, with words p, s of length $r + 1$, f of length $2r + 1$.
Such languages are called locally threshold testable.

Proof

Let $r = 3^k, m = k \cdot s$ with $s = d^{r+1}$ (maximal size of neighborhood of radius r). We show that $G \sim_{r,m} G'$ implies that Duplicator has a winning strategy for k rounds.

Define sets $I_k, \ldots I_0$ of partial isomorphisms. For $j = k, \ldots, 0$ let $(a_1, \ldots, a_{k-j}) \mapsto (b_1, \ldots, b_{k-j})$ belong to I_j if the graph consisting of the union of neighborhoods of radius 3^j around the a_i is isomorphic to the graph consisting of the union of neighborhoods of radius 3^j around the b_i (with $(a_1, \ldots, a_{k-j})$ mapped to $(b_1, \ldots, b_{k-j})$).

Check the "forward property": assume the a is chosen in G.

1. If for some $1 \leq i \leq (k-j)$ we have $d(a, a_i) \leq \frac{2}{3} \cdot 3^j$ then $B^G_{3^j-1}(a) \subseteq B^G_{3^j}(a_i)$ and we choose b in G' as the image of a in $B^{G'}_{3^j}(b_i)$.
2. Else, $B^G_{3^j-1}(a)$ is disjoint from all $B^G_{3^j-1}(a_i)$. It suffices to find a neighborhood in G' of same type as $B^G_{3^j-1}(a)$, disjoint from all $B^{G'}_{3^j-1}(b_i)$. This is possible if there are enough neighborhoods of each type, which is ensured by the choice of m.

Fix an integer $r > 0$.

- For arbitrary graphs G, G' define $G \sim_r G'$ if there exists a bijection $f : V \to V'$ s.t. $B_r^G(v)$ and $B_r^{G'}(f(v))$ are isomorphic, for all $v \in V$.
- If $\vec{a}$ are vertices in V and $\vec{b}$ vertices in V', then define $(G, \vec{a}) \sim_r (G', \vec{b})$ if there exists a bijection $f : V \to V'$ s.t. $B_r^G(\vec{a}, v)$ and $B_r^{G'}(\vec{b}, f(v))$ are isomorphic, for all $v \in V$.

Hanf's theorem

For every k there exists r such that $G \sim_r G'$ implies $G \equiv_k G'$.

Proof.

Induction over the quantifier rank k. Set $r = 3^k$.
For $k = 0$, $(G, \vec{a}) \sim_1 (G', \vec{b})$ means in particular that $\vec{a}$ and $\vec{b}$ satisfy the same atomic formulas, so $G, \vec{a} \equiv_0 G', \vec{b}$.
For $k > 0$ let $G, \vec{a} \sim_{3^{k+1}} G', \vec{b}$. Assume that $G, \vec{a} \models \varphi(\vec{x})$ with φ of quantifier rank $k+1$. The formula $\varphi(\vec{x})$ is a Boolean combination of formulas $\exists y.\, \psi(\vec{x}, y)$, with ψ of rank k.

$G, \vec{a} \models \exists y.\, \psi(\vec{x}, y)$ iff there exists $c \in V$ such that $G, \vec{a}, c \models \psi(\vec{x}, y)$. But $G, \vec{a} \sim_{3^{k+1}} G', \vec{b}$ implies $G, \vec{a}, c \sim_{3^k} G', \vec{b}, d$ for some suitable d (see proof for bounded degree). By induction, $G', \vec{b}, d \models \psi(\vec{x}, y)$, therefore $G', \vec{b} \models \varphi(\vec{x})$.

Local formulas

- Formula $\varphi(x_1, \ldots, x_k)$ is r-local if it uses only quantification restricted to $B_r^G(x_1, \ldots, x_k)$.

Thm. (Gaifman)

For every FO formula $\varphi(x_1, \ldots, x_k)$ there is some integer $r > 0$ such that φ is a Boolean combination of

- r-local formulas of the form $\psi(x_1, \ldots, x_k)$,
- sentences of the form

$$\exists y_1, \ldots, y_n \Big(\bigwedge_j \chi(y_j) \wedge \bigwedge_{i<j} d^{>2r}(y_i, y_j) \Big)$$

where each χ is r-local and $d^{>2r}(y_i, y_j)$ means that y_i, y_j are at least $2r + 1$ far from each other.

Plan

Second-order logic (SO)

Countable set of variables ($x, y, \ldots$ first-order, $X, Y, \ldots$ second-order). We define terms and formulas:

- Each variable, each constant is a term.
- Atomic formulas:
  - FO atomic formulas: $t = t'$, $P(t_1, \ldots, t_k)$ (t, t' terms, P predicate)
  - $X(t_1, \ldots, t_k)$ with t_i terms, X second-order variable
- if $\varphi, \varphi_1, \varphi_2$ are formulas then $\varphi_1 \vee \varphi_2$, $\neg\varphi$ are formulas.
- If φ is a formula then $\exists x\,.\varphi$, $\exists X\,.\varphi$ is a formula.

Semantics

For each formula $\varphi(\vec{x}, \vec{X})$ we define $\mathfrak{A}, \vec{b}, \vec{B} \vDash \varphi$: $\vec{b}$ is a tuple of elements of the universe A of the same length as $\vec{x}$; $\vec{B} = (B_1, \ldots, B_k)$ with $B_i \subseteq A^{n_i}$ if X_i is of arity n_i.
Example: $\mathfrak{A}, \vec{b}, B \vDash X(t_1, \ldots, t_k)$ (with free variables of t_i among $\vec{x}$) if the tuple $(t_1^{\mathfrak{A}}(\vec{b}), \ldots, t_k^{\mathfrak{A}}(\vec{b}))$ belongs to B.

Monadic second-order logic (MSO)

Restriction of SO where all second-order variables have arity 1 (ie., sets).
Notation: $X(x)$, or equivalently $x \in X$.

Existential/universal SO

- Every SO formula can be rewritten into prenex normal form:

$$Q_1 X_1 \ldots Q_m X_m \; Q'_1 x_1 \ldots Q'_n x_n \; \varphi(X_1, \ldots, X_m, x_1, \ldots, x_n)$$

  (Q_i, Q'_j quantifiers, φ is quantifier-free)
- Existential SO logic ($\exists$SO):

$$\exists X_1 \ldots \exists X_m \; \varphi(X_1, \ldots, X_m)$$

  (φ has no further second-order quantification)
- Universal SO logic ($\forall$SO): $\exists$ replaced by $\forall$.

Examples

- REACHABILITY is $\forall$MSO-definable and $\exists$SO-definable
- 3-COL is $\exists$MSO-definable
- CLIQUE and HAMILTONICITY are $\exists$SO-definable

MSO-games

- Given: 2 graphs G_0, G_1; $G_i = (V_i, E_i)$.
- Players: Spoiler and Duplicator.
- k rounds. In each round, two possible kinds of moves:
  1. Spoiler chooses a graph and a node in that graph. Duplicator answers by choosing a node in the other graph.
  2. Spoiler chooses a graph and a subset of nodes in that graph. Duplicator answers by choosing a subset of nodes in the other graph.

  Let $a_1, \ldots, a_m$ (resp. $b_1, \ldots, b_m$) be the nodes chosen in G_0 (resp. G_1), and $A_1, \ldots, A_n$ (resp. $B_1, \ldots, B_n$) be the sets of nodes chosen in G_0 (resp. G_1).
- Duplicator wins if for all $1 \leq i, j \leq m$, $1 \leq p \leq n$:
  1. $a_i = a_j$ iff $b_i = b_j$,
  2. $(a_i, a_j) \in E_0$ iff $(b_i, b_j) \in E_1$,
  3. $a_i \in A_p$ iff $b_i \in B_p$.

Notations

- MSO$[k]$: MSO (monadic SO) formulas of quantifier rank k.
- Let G_0, G_1 be graphs. For MSO$[k]$ formulas with free variables $\vec{x}, \vec{X}$: let $\vec{a}$ ($\vec{b}$ resp.) be a vector of nodes in G_0 (G_1, resp.) and $\vec{A}$ ($\vec{B}$, resp.) be a vector of subsets of nodes in G_0 (G_1, resp.).
  Write $(G_0, \vec{a}, \vec{A}) \equiv_k (G_1, \vec{b}, \vec{B})$ if for all $\varphi \in$ MSO$[k]$:

$$G_0 \vDash \varphi(\vec{a}, \vec{A}) \quad \Leftrightarrow \quad G_1 \vDash \varphi(\vec{b}, \vec{B})$$

Thm.

The following are equivalent:

- $(G_0, \vec{a}, \vec{A}) \equiv_k (G_1, \vec{b}, \vec{B})$
- Duplicator has a winning strategy in the k round MSO-game.

MSO over words and trees

- Data complexity of model-checking MSO over words or trees is solvable in linear time (Büchi's theorem). Combined complexity is solvable in polynomial space.
- Satisfiability of MSO over words or trees is solvable in non-elementary time ($\geq \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2^x}}}}}}_{n}$ for every n).

"Tree-like" graphs: bounded tree-width

An undirected graph $G = (V, E)$ has tree-width k if there is some tree T with nodes labeled by subsets of V of size at most $k + 1$, such that:

- For each edge uv of G, there is some node in T with label containing both u and v.
- For each node v of G, the set of nodes of T with labels containing v, is connected.

Courcelle's theorem

Data complexity of (model-checking) MSO over graphs of fixed tree-width is solvable in linear time. Combined complexity is solvable in PSPACE.

Fagin's theorem

$\exists$SO = NP.

Proof

1. Upper bound: given a graph G and an $\exists$SO formula $\exists S.\, \varphi$, with φ first-order model-checking $G \vDash \varphi$ is in NP: guess S and recall that data complexity of FO model-checking is in P.
2. Lower bound: start with any NP-complete graph problem.
   Express in FO that a binary relation L is linear order on the set of vertices $\{0, \ldots, n-1\}$:
   - $\forall x, y, z : L(x, x) \wedge (L(x, y) \wedge L(y, z) \rightarrow L(x, z))$ (reflexive, transitive)
   - $\forall x, y : L(x, y) \vee L(y, x)$ (total)
   - $\forall x, y : L(x, y) \wedge L(y, x) \rightarrow x = y$ (anti-symmetric)

   Use L for describing the successor relation on $\{0, \ldots, n-1\}^k$ (integers in basis n, ranging from 0 to $n^k - 1$). Encode then the proof of Cook's theorem.