Topics in Logic and Complexity
Handout 3

Anuj Dawar

MPhil Advanced Computer Science, Lent 2011

MSO is FPT on Words

There is a computable function f such that the problem of deciding, given a word w and an MSO sentence ϕ whether,

$$S_w \models \phi$$

can be decided in time $O(f(l)n)$ where l is the length of ϕ and n is the length of w.

The algorithm proceeds by constructing, from ϕ an *automaton* $\mathcal{A}_\phi$ such that the language recognized by $\mathcal{A}_\phi$ is

$$\{w \mid S_w \models \phi\}$$

then running $\mathcal{A}_\phi$ on w.

The automaton $\mathcal{A}_\phi$

The states of $\mathcal{A}_\phi$ are the equivalence classes of $\equiv_m^{\mathsf{MSO}}$ where m is the quantifier rank of ϕ.

We write $\mathsf{Type}_m^{\mathsf{MSO}}(\mathbb{A})$ for the set of all formulas ϕ with $\mathrm{qr}(\phi) \leq m$ such that $\mathbb{A} \models \phi$.

$\mathbb{A} \equiv_m^{\mathsf{MSO}} \mathbb{B}$ is equivalent to

$$\mathsf{Type}_m^{\mathsf{MSO}}(\mathbb{A}) = \mathsf{Type}_m^{\mathsf{MSO}}(\mathbb{B})$$

There is a single formula $\theta_{\mathbb{A}}$ that characterizes $\mathsf{Type}_m^{\mathsf{MSO}}(\mathbb{A})$.

It turns out that we can compute $\theta_{S_{w \cdot a}}$ from θ_{S_w}.

Trees

An (undirected) *forest* is an *acyclic* graph and a *tree* is a connected forest.

We next aim to show that there is an algorithm that decides, given a tree T and an MSO sentence ϕ whether

$$T \models \phi$$

and runs in time $O(f(l)n$ where l is the length of ϕ and n is the size of T.

Rooted Directed Trees

A *rooted, directed tree* (T, a) is a directed graph with a distinguished vertex a such that for every vertex v there is a *unique* directed path from a to v.

We will actually see that MSO satisfaction is FPT for rooted, directed trees.

The result for undirected trees follows, as any undirected tree can be turned into a rooted directed one by choosing any vertex as a root and directing edges away from it.

Sums of Rooted Trees

Given rooted, directed trees (T, a) and (S, b) we define the sum

$$(T, a) \oplus (S, b)$$

to be the rooted directed tree which is obtained by taking the *disjoint union* of the two trees while *identifying* the roots.

That is,

- the set of vertices of $(T, a) \oplus (S, b)$ is $V(T) \uplus V(S) \setminus \{b\}$.
- the set of edges is $E(T) \cup E(S) \cup \{(a, v) \mid (b, v) \in E(S)\}$.

Congruence

If $(T_1, a_1) \equiv_m^{\mathsf{MSO}} (T_2, a_2)$ and $(S_1, b_1) \equiv_m^{\mathsf{MSO}} (S_2, b_2)$, then

$$(T_1, a_1) \oplus (S_1, b_1) \equiv_m^{\mathsf{MSO}} (T_2, a_2) \oplus (S_2, b_2).$$

This can be proved by an application of Ehrenfeucht games.

Moreover (though we skip the proof), $\mathsf{Type}_m^{\mathsf{MSO}}((T, a) \oplus (S, b))$ can be computed from $\mathsf{Type}_m^{\mathsf{MSO}}((T, a))$ and $\mathsf{Type}_m^{\mathsf{MSO}}((S, b))$.

Add Root

For any rooted, directed tree (T, a) define $r(T, a)$ to be rooted directed tree obtained by adding to (T, a) a new vertex, which is the root and whose only child is a.

That is,

- the vertices of $r(T, a)$ are $V(T) \cup \{a'\}$ where a' is not in $V(T)$;
- the root of $r(T, a)$ is a'; and
- the edges of $r(T, a)$ are $E(T) \cup \{(a', a)\}$.

Again, $\mathsf{Type}_m^{\mathsf{MSO}}(r(T, a))$ can be computed from $\mathsf{Type}_m^{\mathsf{MSO}}(T, a)$.

MSO satisfaction is FPT on Trees

Any *rooted, directed tree* (T, a) can be obtained from *singleton trees* by a sequence of applications of $\oplus$ and r.

The length of the sequence is linear in the size of T.

We can compute $\mathsf{Type}_m^{\mathsf{MSO}}(T, a)$ in linear time.

The Method of Decompositions

Suppose $\mathcal{C}$ is a class of graphs such that there is a finite class $\mathcal{B}$ and a finite collection Op of operations such that:

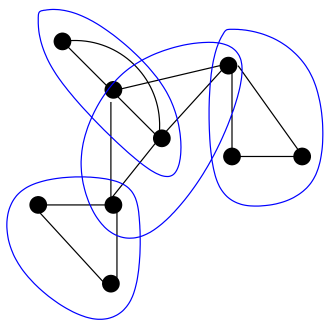
- $\mathcal{C}$ is contained in the closure of $\mathcal{B}$ under the operations in Op;
- there is a polynomial-time algorithm which computes, for any $G \in \mathcal{C}$, an Op-decomposition of G over $\mathcal{B}$; and
- for each m, the equivalence class $\equiv_m^{\mathsf{MSO}}$ is an *effective* congruence with respect to to all operations $o \in \mathsf{Op}$ (i.e., the $\equiv_m^{\mathsf{MSO}}$-type of $o(G_1, \ldots, G_s)$ can be computed from the $\equiv_m^{\mathsf{MSO}}$-types of $G_1, \ldots, G_s$).

Then, MSO satisfaction is fixed-parameter tractable on $\mathcal{C}$.

Treewidth

The *treewidth* of an undirected graph is a measure of how tree-like the graph is.

A graph has treewidth k if it can be covered by subgraphs of at most $k + 1$ nodes in a tree-like fashion.



This gives a *tree decomposition* of the graph.

Treewidth

Treewidth is a measure of how *tree-like* a graph is.

For a graph $G = (V, E)$, a *tree decomposition* of G is a relation $D \subset V \times T$ with a tree T such that:

- for each $v \in V$, the set $\{t \mid (v, t) \in D\}$ forms a connected subtree of T; and
- for each edge $(u, v) \in E$, there is a $t \in T$ such that $(u, t), (v, t) \in D$.

The *treewidth* of G is the least k such that there is a tree T and a tree decomposition $D \subset V \times T$ such that for each $t \in T$,

$$|\{v \in V \mid (v, t) \in D\}| \leq k + 1.$$

Dynamic Programming

It has long been known that graphs of small treewidth admit efficient *dynamic programming* algorithms for intractable problems.

In general, these algorithms proceed bottom-up along a tree decomposition of G.

At any stage, a small set of vertices form the "*interface*" to the rest of the graph.

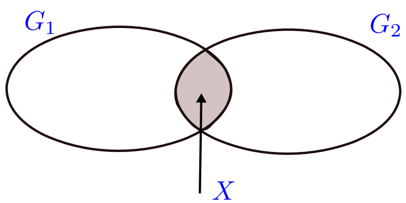
This allows a recursive decomposition of the problem.

Treewidth

Looking at the decomposition *bottom-up*, a graph of treewidth k is obtained from graphs with at most $k+1$ nodes through a finite sequence of applications of the operation of taking *sums over sets* of at most k elements.

$G_1 \oplus_X G_2$

$|X| \leq k$



We let $\mathcal{T}_k$ denote the class of graphs G such that $\mathrm{tw}(G) \leq k$.

Treewidth

More formally,

Consider graphs with up to $k+1$ distinguished vertices $C = \{c_0, \ldots, c_k\}$.

Define a *merge* operation $(G \oplus_C H)$ that forms the union of G and H *disjointly apart from* C.

Also define $\mathsf{erase}_i(G)$ that erases the name c_i.

Then a graph G is in $\mathcal{T}_k$ if it can be formed from graphs with at most $k+1$ vertices through a sequence of such operations.

Congruence

- Any $G \in \mathcal{T}_k$ is obtained from $\mathcal{B}_k$ by finitely many applications of the operations erase_i and $\oplus_C$.
- If $G_1, \rho_1 \equiv_m^{\mathsf{MSO}} G_2, \rho_2$, then
$$\mathsf{erase}_i(G_1, \rho_1) \equiv_m^{\mathsf{MSO}} \mathsf{erase}_i(G_2, \rho_2)$$
- If $G_1, \rho_1 \equiv_m^{\mathsf{MSO}} G_2, \rho_2$, and $H_1, \sigma_1 \equiv_m^{\mathsf{MSO}} H_2, \sigma_2$ then
$$(G_1, \rho_1) \oplus_C (H_1, \sigma_1) \equiv_m^{\mathsf{MSO}} (G_2, \rho_2) \oplus_C (H_2, \sigma_2)$$

Note: a special case of this is that $\equiv_m^{\mathsf{MSO}}$ is a congruence for *disjoint union* of graphs.

Courcelle's Theorem

Theorem (Courcelle)
For any MSO sentence ϕ and any k there is a linear time algorithm that decides, given $G \in \mathcal{T}_k$ whether $G \models \phi$.

Given $G \in \mathcal{T}_k$ and ϕ, compute:

- from G a labelled tree T; and
- from ϕ a bottom-up tree automaton $\mathcal{A}$

such that $\mathcal{A}$ accepts T if, and only if, $G \models \phi$.

Bounded Degree Graphs

In a graph $G = (V, E)$ the *degree* of a vertex $v \in V$ is the number of neighbours of v, i.e.

$$|\{u \in V \mid (u, v) \in E\}|.$$

We write $\delta(G)$ for the *smallest* degree of any vertex in G.

We write $\Delta(G)$ for the *largest* degree of any vertex in G.

$\mathcal{D}_k$—the class of graphs G with $\Delta(G) \leq k$.

Bounded Degree Graphs

Theorem (Seese)
For every sentence ϕ of FO and every k there is a linear time algorithm which, given a graph $G \in \mathcal{D}_k$ determines whether $G \models \phi$.

A proof is based on *locality* of first-order logic.

To be precise a strengthening of *Hanf's theorem*.

> **Note:** this is not true for MSO unless P = NP.
>
> Construct, for any graph G, a graph G' such that $\Delta(G') \leq 5$ and G' is 3-colourable iff G is, and the map $G \mapsto G'$ is polynomial-time computable.

Hanf Types

For an element a in a structure $\mathbb{A}$, define

> $N_r^{\mathbb{A}}(a)$—the substructure of $\mathbb{A}$ generated by the elements whose distance from a (in $G\mathbb{A}$) is at most r.

We say $\mathbb{A}$ and $\mathbb{B}$ are *Hanf equivalent* with radius r and threshold q ($\mathbb{A} \simeq_{r,q} \mathbb{B}$) if, for every $a \in A$ the two sets

$$\{a' \in a \mid N_r^{\mathbb{A}}(a) \cong N_r^{\mathbb{A}}(a')\} \quad \text{and} \quad \{b \in B \mid N_r^{\mathbb{A}}(a) \cong N_r^{\mathbb{B}}(b)\}$$

either have the same size or both have size greater than q;

and, similarly for every $b \in B$.

Hanf Locality Theorem

Theorem (Hanf)
For every vocabulary σ and every m there are $r \leq 3^m$ and $q \leq m$ such that for any σ-structures $\mathbb{A}$ and $\mathbb{B}$: if $\mathbb{A} \simeq_{r,q} \mathbb{B}$ then $\mathbb{A} \equiv_m \mathbb{B}$.

In other words, if $r \geq 3^m$, the equivalence relation $\simeq_{r,m}$ is a refinement of $\equiv_m$.

For $\mathbb{A} \in \mathcal{D}_k$:

$N_r^{\mathbb{A}}(a)$ has at most $k^r + 1$ elements

each $\simeq_{r,m}$ has finite index.

Each $\simeq_{r,m}$-class t can be characterised by a finite table, I_t, giving isomorphism types of neighbourhoods and numbers of their occurrences up to threshold m.

Satisfaction on $\mathcal{D}_k$

For a sentence ϕ of FO, we can compute a set of tables $\{I_1, \ldots, I_s\}$ describing $\simeq_{r,m}$-classes consistent with it.

This computation is independent of any structure $\mathbb{A}$.

Given a structure $\mathbb{A} \in \mathcal{D}_k$,

for each a, determine the isomorphism type of $N_r^{\mathbb{A}}(a)$

construct the table describing the $\simeq_{r,m}$-class of $\mathbb{A}$.

compare against $\{I_1, \ldots, I_s\}$ to determine whether $\mathbb{A} \models \phi$.

For fixed k, r, m, this requires time *linear* in the size of $\mathbb{A}$.

Note: satisfaction for FO is in $O(f(l, k)n)$.

Reading List for this Handout

1. Libkin. Sections 7.6 and 7.6