Symmetric Computation: Part 1

Anuj Dawar

Department of Computer Science and Technology, University of Cambridge

PhD Open, Warsaw, 17 October 2019

The Science of Abstraction

Computer Science is the *Mechanization of Abstraction*.

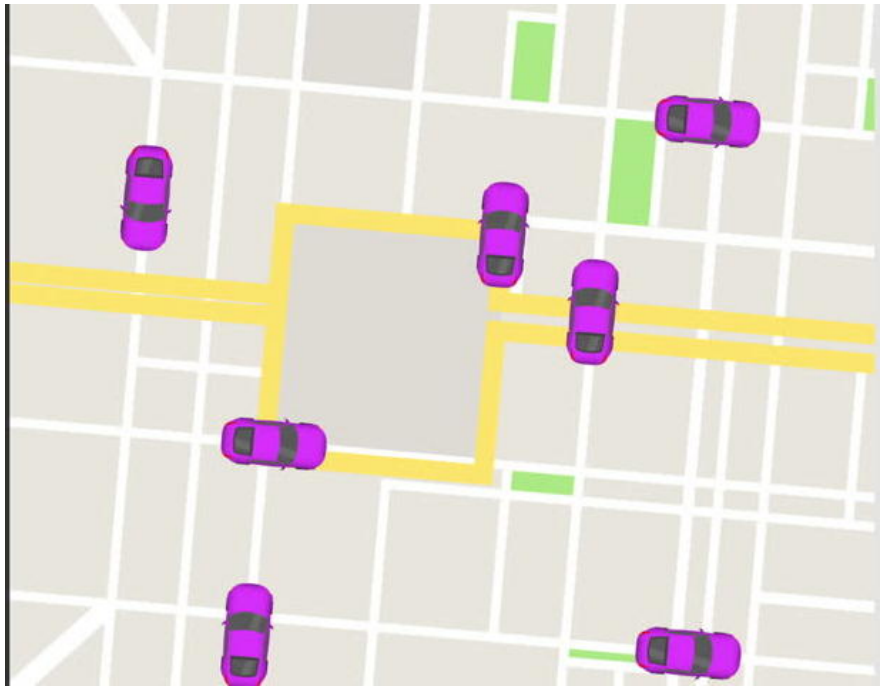
(Aho and Ullman)

The first step to solving a problem *computationally* is to strip away irrelevant concrete detail and formulate it as an *abstract* problem.

In the terms of **Aho and Ullman**, this means constructing an abstract *data model* and deciding which aspects of *concrete, messy* reality are represented there.

Example: Matching Taxis to Passengers

Example: Consider the problem of assigning a set of available taxis to a set of *waiting passengers*.



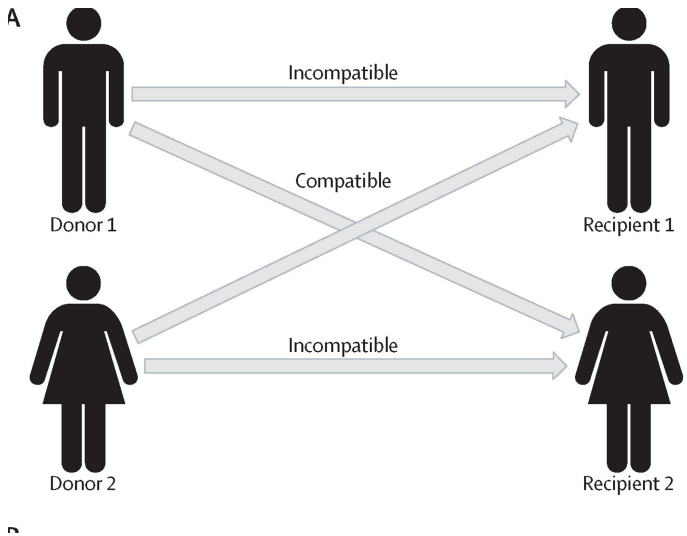
The assignment may have to minimize distance travelled by the taxis, and respond in *real-time*.

Example: Matching Kidney Patients to Donors

The English *National Health Service* runs a *kidney matching programme*.

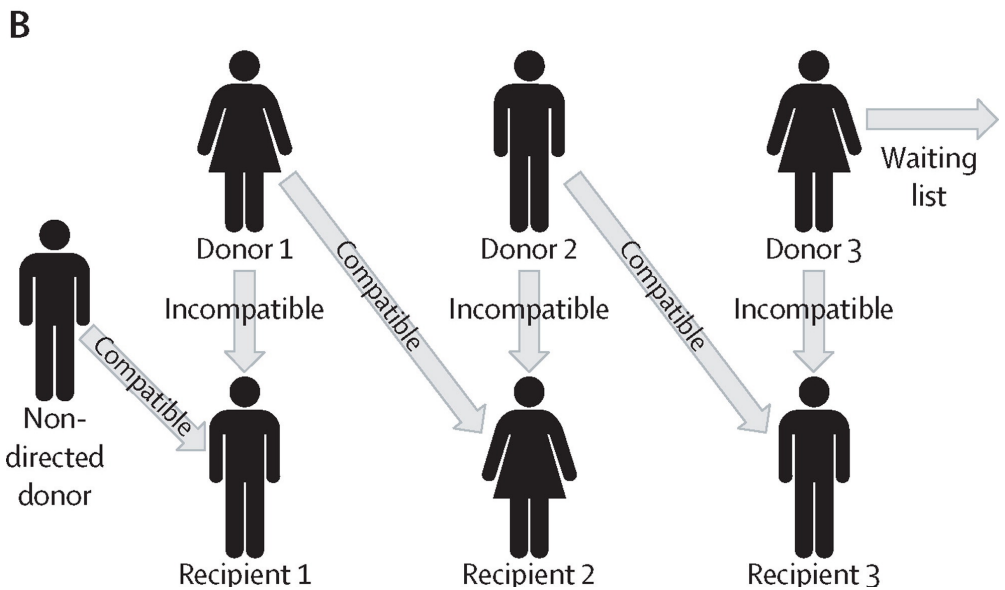
Patients needing a *kidney transplant* often have a relative who is willing to donate a kidney.
However, thee may be tissue incompatibility.

If one is fortunate, a matching donor/recipient *pair* can be found and a kidney *swap* arranged:



Kidney Matching

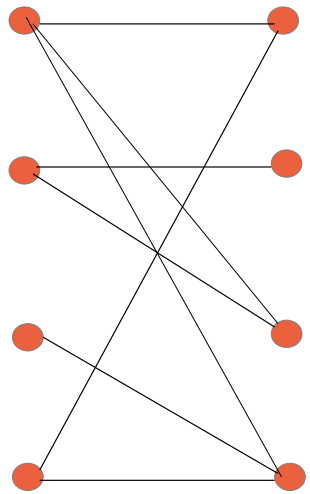
The likelihood of finding a match is greatly increased if we look for longer chains of donor/recipient matches:



Graph Matching

At the core of both is the same *algorithmic problem*.

We have a *bipartite graph* in which to find a *perfect matching*.



Thinking about it at this *abstract* level is what allows us *re-use* algorithmic ideas.

Note, this is *not* just about re-using *code* but also more *fundamental insights*.

A Theoretician's View

As a *theoretical computer scientist*, abstraction is my stock in trade.

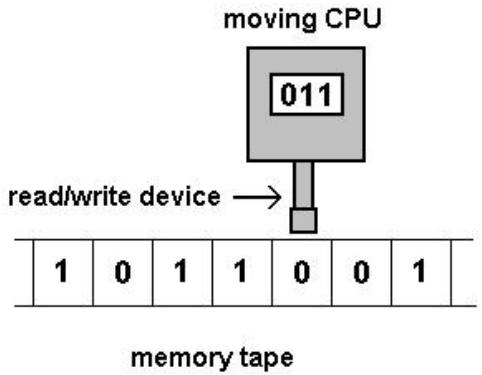
Abstraction, Symmetry and Complexity

Abstract

I'm a theoretician, of course it's abstract. If you want concrete, go to a building supplies merchant ...

Algorithms and Abstraction

Algorithms are usually described by operations on *abstract data* such as graphs.



The *complexity* of algorithms, and particularly *complexity classes* are defined by machine models (*e.g. Turing machines*) operating on *strings of symbols*.

The *mismatch* is generally considered harmless as data structures can be encoded as strings.

However, it does *break abstraction*.

Sometimes even *high-level* descriptions of algorithms break the level of abstraction.

Graph Matching

Given a bipartite graph $G = (A \cup B, E)$,

> *Start with an empty matching $M = \emptyset$.*
> *Choose an $a \in A$ that is currently unmatched and find an augmenting path P starting at a and ending in an unmatched $b \in B$.*
> *Set M to be $M \oplus P$.*

The *choice* of a is arbitrary and generally relies on *concrete* hidden data.

Abstract data has *symmetries* that an abstract algorithm should respect.

The Role of Symmetry

If we expect an algorithm to work *at the level of abstraction* of graphs, then it must respect *symmetries* of the graph.

If two nodes in a graph G are *indistinguishable* by properties of the graph, then they should not be distinguished in any way by the algorithm.

This opens up the question of what algorithms have the property of respecting symmetry?

For instance, it can be shown that the **Micali-Vazirani** algorithm for *graph matching* is *not* symmetry-respecting.

Can *Matching* be solved by an *efficient* and *symmetry respecting* algorithm?

Symmetry from High-level Description

Algorithms that are *automatically generated* from high-level descriptions, will preserve symmetries.

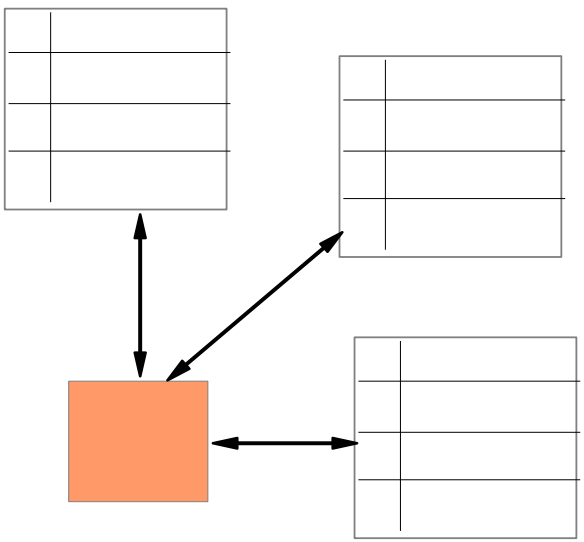
This sentence says that the relation M is a matching in the graph with edge relation E.

$$\forall x \forall y [M(x, y) \rightarrow E(x, y)] \wedge \forall x \exists! y M(x, y)$$

An algorithm to search for such an M in a graph, generated from this description, would *most likely* be *exponential*.

Relational Machines

Formal Models of algorithms that work on *abstract* structures have been well-studied in the context of *database query languages*.



Input: A relational database

Store: relational and numerical registers

Operations: *join, projection, complementation, counting*

Logic

Query languages for relational databases are often modelled in *Logic*.

Finite Model Theory studies methods for analyzing the *expressive power* of logics over finite relational structures.

Consider *first-order predicate logic*.

> *Fix a vocabulary* σ *of relation symbols* $(R_1, \ldots, R_m)$ *and a collection* X *of variables.*

The formulas are given by

$$R_i(\mathbf{x}) \mid x = y \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$$

First-Order Logic

For a first-order sentence φ, we ask what is the *computational complexity* of the problem:

Given: a structure $\mathbb{A}$
Decide: if $\mathbb{A} \models \varphi$

In other words, how complex can the collection of finite models of φ be?

In order to talk of the complexity of a class of finite structures, we need to fix some way of representing finite structures as strings.

Encoding Structures

We use an alphabet $\Sigma = \{0, 1, \#\}$.
For a structure $\mathbb{A} = (A, R_1, \ldots, R_m)$, fix a linear order $<$ on $A = \{a_1, \ldots, a_n\}$.
R_i (of arity k) is encoded by a string $[R_i]_<$ of 0s and 1s of length n^k.

$$[\mathbb{A}]_< = \underbrace{1 \cdots 1}_{n} \#[R_1]_< \# \cdots \#[R_m]_<$$

The exact string obtained depends on the choice of order.

Invariance

Note that the decision problem:

Given a string $[\mathbb{A}]_<$ *decide whether* $\mathbb{A} \models \varphi$

has a natural invariance property.

It is invariant under the following equivalence relation

Write $w_1 \sim w_2$ *to denote that there is some structure* $\mathbb{A}$ *and orders* $<_1$ *and* $<_2$ *on its universe such that*

$$w_1 = [\mathbb{A}]_{<_1} \text{ and } w_2 = [\mathbb{A}]_{<_2}$$

Note: deciding the equivalence relation $\sim$ is just the same as deciding structure isomorphism.

Naïve Algorithm

The straightforward algorithm proceeds recursively on the structure of φ:

- Atomic formulas by direct lookup.
- Boolean connectives are easy.
- If $\varphi \equiv \exists x\, \psi$ then for each $a \in \mathbb{A}$ check whether

$$(\mathbb{A}, c \mapsto a) \models \psi[c/x],$$

where c is a new constant symbol.

This runs in time $O(ln^m)$ and $O(m \log n)$ space, where l is the length of φ and m is the nesting depth of quantifiers in φ.

$$\mathrm{Mod}(\varphi) = \{\mathbb{A} \mid \mathbb{A} \models \varphi\}$$

is in *logarithmic space* and *polynomial time*.

Second-Order Logic

There are computationally easy properties that are not definable in first-order logic.

- There is no sentence φ of first-order logic such that $\mathbb{A} \models \varphi$ if, and only if, $|A|$ is even.
- There is no formula $\varphi(E, x, y)$ that defines the transitive closure of a binary relation E.

Consider second-order logic, extending first-order logic with *relational quantifiers* — $\exists X \varphi$

Examples

Evennness

This formula is true in a structure if, and only if, the size of the domain is even.

$$\exists B \exists S \quad \forall x \exists y B(x,y) \wedge \forall x \forall y \forall z B(x,y) \wedge B(x,z) \rightarrow y = z$$
$$\forall x \forall y \forall z B(x,z) \wedge B(y,z) \rightarrow x = y$$
$$\forall x \forall y S(x) \wedge B(x,y) \rightarrow \neg S(y)$$
$$\forall x \forall y \neg S(x) \wedge B(x,y) \rightarrow S(y)$$

Examples

Transitive Closure

The following formula is true of a pair of elements a, b in a structure if, and only if, there is an E-path from a to b.

$$\forall S\big(S(a) \wedge \forall x \forall y[S(x) \wedge E(x,y) \rightarrow S(y)] \rightarrow S(b)\big)$$

Matching

The following formula is true in a graph (V, E) if, and only if, the graph contains a perfect matching.

$$\exists M\ \forall x \forall y[M(x,y) \rightarrow E(x,y)] \wedge \forall x \exists! y M(x,y)$$

Examples

3-Colourability

The following formula is true in a graph (V, E) if, and only if, it is 3-colourable.

$$\begin{aligned}
\exists R \exists B \exists G \quad & \forall x(Rx \vee Bx \vee Gx) \wedge \\
& \forall x(\quad \neg(Rx \wedge Bx) \wedge \neg(Bx \wedge Gx) \wedge \neg(Rx \wedge Gx)) \wedge \\
& \forall x \forall y(Exy \rightarrow (\quad \neg(Rx \wedge Ry) \wedge \\
& \qquad\qquad\qquad\quad \neg(Bx \wedge By) \wedge \\
& \qquad\qquad\qquad\quad \neg(Gx \wedge Gy)))
\end{aligned}$$

Fagin's Theorem

Theorem (Fagin)
A class $\mathcal{C}$ of finite structures is definable by a sentence of *existential second-order logic* if, and only if, it is decidable by a *nondeterminisitic machine* running in polynomial time.

$$\text{ESO} = \text{NP}$$

Is there a logic for P?

The major open question in *Descriptive Complexity* (first asked by Chandra and Harel in 1982) is whether there is a logic $\mathcal{L}$ such that

> *for any class of finite structures $\mathcal{C}$, $\mathcal{C}$ is definable by a sentence of $\mathcal{L}$ if, and only if, $\mathcal{C}$ is decidable by a deterministic machine running in polynomial time.*

Formally, we require $\mathcal{L}$ to be a *recursively enumerable* set of sentences, with a computable map taking each sentence to a Turing machine M and a polynomial time bound p such that (M, p) accepts a *class of structures*.

(Gurevich 1988)

Inductive Definitions

Let $\varphi(R, x_1, \ldots, x_k)$ be a first-order formula in the vocabulary $\sigma \cup \{R\}$
Associate an operator Φ on a given σ-structure $\mathbb{A}$:

$$\Phi(R^{\mathbb{A}}) = \{\mathbf{a} \mid (\mathbb{A}, R^{\mathbb{A}}, \mathbf{a}) \models \varphi(R, \mathbf{x})\}$$

We define the *non-decreasing* sequence of relations on $\mathbb{A}$:

$$\Phi^0 = \emptyset$$

$$\Phi^{m+1} = \Phi^m \cup \Phi(\Phi^m)$$

The *inflationary fixed point* of Φ is the limit of this sequence.
On a structure with n elements, the limit is reached after at most n^k stages.

FP

The logic FP is formed by closing first-order logic under the rule:

> *If φ is a formula of vocabulary $\sigma \cup \{R\}$ then $[\mathbf{ifp}_{R,\mathbf{x}}\varphi](\mathbf{t})$ is a formula of vocabulary σ.*

The formula is read as:

> *the tuple $\mathbf{t}$ is in the inflationary fixed point of the operator defined by φ*

LFP is the similar logic obtained using *least fixed points* of *monotone* operators defined by *positive* formulas.
LFP and FP have the same expressive power **(Gurevich-Shelah 1986; Kreutzer 2004)**.

Transitive Closure

The formula

$$[\mathbf{ifp}_{T,xy}(x = y \vee \exists z(E(x,z) \wedge T(z,y)))](u,v)$$

defines the *transitive closure* of the relation E

The expressive power of FP properly extends that of first-order logic.

Theorem

On structures which come equipped with a linear order FP *expresses exactly the properties that are in* P.

(Immerman; Vardi 1982)

FP vs. Ptime

The order cannot be built up inductively.
It is an open question whether a *canonical* string representation of a structure can be constructed in polynomial-time.

> *If it can, there is a logic for P.*
> *If not, then $P \neq NP$.*

All P classes of structures can be expressed by a sentence of FP with $<$, which is invariant under the choice of order. The set of all such sentences is not *r.e.*

FP by itself is too weak to express all properties in P.
Evenness is not definable in FP.

Finite Variable Logic

We write L^k for the first order formulas using only the variables $x_1, \ldots, x_k$.

$$(\mathbb{A}, \mathbf{a}) \equiv^{L^k} (\mathbb{B}, \mathbf{b})$$

denotes that there is no formula φ of L^k such that $\mathbb{A} \models \varphi[\mathbf{a}]$ and $\mathbb{B} \not\models \varphi[\mathbf{b}]$

If $\varphi(R, \mathbf{x})$ has k variables all together, then each of the relations in the sequence:

$$\Phi^0 = \emptyset; \Phi^{m+1} = \Phi^m \cup \Phi(\Phi^m)$$

is definable in L^{2k}.

Proof by induction, using *substitution* and *renaming* of bound variables.

Pebble Game

The k-pebble game is played on two structures $\mathbb{A}$ and $\mathbb{B}$, by two players—*Spoiler* and *Duplicator*—using k pairs of pebbles $\{(a_1, b_1), \ldots, (a_k, b_k)\}$.

> *Spoiler moves by picking a pebble and placing it on an element (a_i on an element of $\mathbb{A}$ or b_i on an element of $\mathbb{B}$).*
>
> *Duplicator responds by picking the matching pebble and placing it on an element of the other structure*
>
> *Spoiler wins at any stage if the partial map from $\mathbb{A}$ to $\mathbb{B}$ definedby the pebble pairs is not a partial isomorphism*
>
> *If Duplicator has a winning strategy for q moves, then $\mathbb{A}$ and $\mathbb{B}$ agree on all sentences of L^k of quantifier rank at most q.*
>
> **(Barwise)**

$\mathbb{A} \equiv^{L^k} \mathbb{B}$ if, for every q, *Duplicator* wins the q round, k pebble game on $\mathbb{A}$ and $\mathbb{B}$. Equivalently (on finite structures) *Duplicator* has a strategy to play forever.

Evenness

To show that *Evenness* is not definable in FP, it suffices to show that:

> *for every* k*, there are structures* $\mathbb{A}_k$ *and* $\mathbb{B}_k$ *such that* $\mathbb{A}_k$ *has an even number of elements,* $\mathbb{B}_k$ *has an odd number of elements and*
>
> $$\mathbb{A} \equiv^{L^k} \mathbb{B}.$$

It is easily seen that *Duplicator* has a strategy to play forever when one structure is a set containing k elements (and no other relations) and the other structure has $k+1$ elements.

Matching

Take $K_{k,k}$—the complete bipartite graph on two sets of k vertices.
and $K_{k,k+1}$—the complete bipartite graph on two sets, one of k vertices, the other of $k+1$.

These two graphs are $\equiv^{L^k}$ equivalent, yet one has a perfect matching, and the other does not.

Fixed-point Logic with Counting

Immerman proposed FPC—the extension of FP with a mechanism for *counting*

Two sorts of variables:

- $x_1, x_2, \ldots$ range over $|A|$—the domain of the structure;
- $\nu_1, \nu_2, \ldots$ which range over *non-negative integers*.

If $\varphi(x)$ is a formula with free variable x, then $\#x\varphi$ is a *term* denoting the *number* of elements of $\mathbb{A}$ that satisfy φ.

We have arithmetic operations $(+, \times)$ on *number terms*.

Quantification over number variables is *bounded*: $(\exists \nu < t)\, \varphi$

Examples

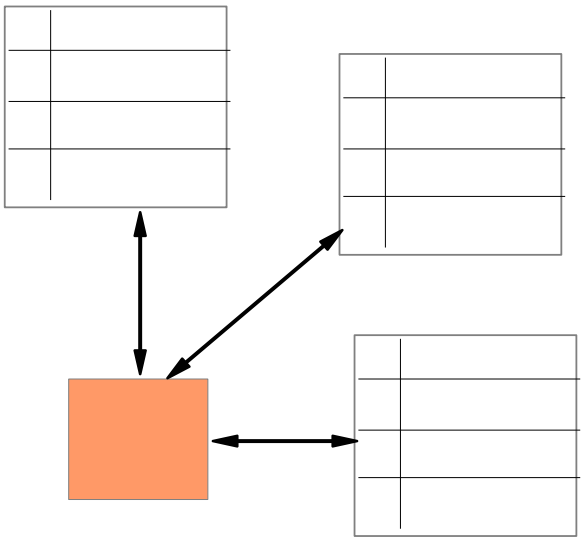
The following formula is true in a graph if, and only if, it has an even number of edges.

$$\begin{aligned}\exists \nu_1 \leq \#x(x=x)\ \exists \nu_2 \leq \#x(x=x)\ \exists \nu_3 \leq \nu_1 \times \nu_2 \\ \nu_3 = \textstyle\sum_{\mu < \#x(x=x)} \mu \times (\#x(\#yE(x,y) = \mu)) \\ \exists \nu_4 \leq \nu_3 (\nu_4 \times 4 = \nu_3)\end{aligned}$$

where the sum in the second line can be expressed using the fixed-point operator.

$$\begin{aligned}&\mathbf{ifp}_{T,\mu,\tau}[\mu = 0 \wedge \tau = \#x(\forall y \neg E(x,y)) \vee \\ &\quad \exists \mu' \leq \mu \exists \alpha' \leq \alpha (\mu = \mu' + 1 \wedge T(\mu', \alpha') \wedge \\ &\quad\quad \alpha = \alpha' + \mu \times \#x(\#yE(x,y) = \mu))](\nu_3, \#x(x=x)).\end{aligned}$$

Relational Machines



Input: A relational database

Store: relational and numerical registers

Operations: *join, projection, complementation, counting*

Properites expressible in FPC are *exactly* those decidable by such a machine in *polynomial time*.

Counting Quantifiers

C^k is the logic obtained from *first-order logic* by allowing:

- *counting quantifiers*: $\exists^i x\, \varphi$; and
- only the variables $x_1, \ldots . x_k$.

Every formula of C^k is equivalent to a formula of first-order logic, albeit one with more variables.

For every sentence φ of FPC, there is a k such that if $\mathbb{A} \equiv^{C^k} \mathbb{B}$, then

$$\mathbb{A} \models \varphi \quad \text{if, and only if,} \quad \mathbb{B} \models \varphi.$$

Weisfeiler-Leman Equivalences

$G \equiv^{C^k} H$ iff G and H cannot be distinguished by a sentence of first-order logic with *counting quantifiers* using only k variables.

$G \equiv^{C^{k+1}} H$ iff G and H are not distinguished by the coarsest partition of the k-tuples of G into classes $P_1, \ldots, P_t$ satisfying:

> *two tuples* $\mathbf{u}$ *and* $\mathbf{v}$ *in the same class* P_i *cannot be distinguished by counting the number of substitutions we can make in them to get a tuple in class* P_j.

Weisfeiler-Leman Equivalences

The k-*dimensional Weisfeiler-Leman* equivalence relation is an *overapproximation* of the isomorphism relation.

If G, H are n-vertex graphs and $k < n$, we have:

$$G \cong H \quad \Leftrightarrow \quad G \equiv^n H \quad \Rightarrow \quad G \equiv^{k+1} H \quad \Rightarrow \quad G \equiv^k H.$$

$G \equiv^k H$ is decidable in time $n^{O(k)}$.

It has many equivalent characterisations arising from

- *combinatorics* **(Babai)**
- *logic* **(Immerman-Lander)**
- *algebra* **(Weisfeiler; Holm)**
- *linear optimization* **(Atserias-Maneva; Malkin)**

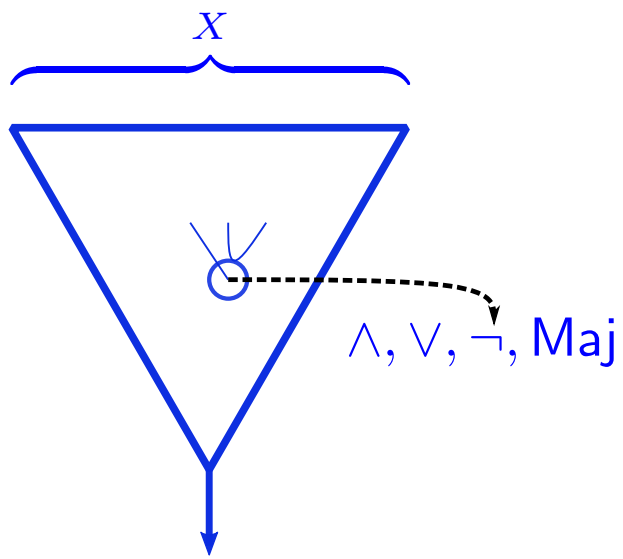
Circuits

A *language* $L \subseteq \{0,1\}^*$ can be described by a family of *Boolean functions*:

$$(f_n)_{n\in\omega} : \{0,1\}^n \to \{0,1\}.$$

Each f_n may be computed by a *circuit* C_n made up of

- Gates labeled by Boolean operators: $\wedge, \vee, \neg$,
- Boolean inputs: $x_1, \ldots, x_n$, and
- A distinguished gate determining the output.



Symmetric and Invariant Boolean functions

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *symmetric* if it is invariant under *all* permutations of its inputs.

For a symmetric f, the $f(x)$ is determined by the number of 1s in x.

A Boolean function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is *graph-invariant* if it is invariant under the natural action of S_n on its inputs.

More generally, for a relational vocabulary τ, we say that f is τ-invariant if it is defined on string encodings of τ-structures and is invariant under $\sim$.

Circuit Complexity

Circuits are just the *unfoldings* of the behaviour of an algorithm on inputs of a fixed size n into simple actions such as Boolean *AND*, *OR* and *NOT* operations.

If there is a polynomial $p(n)$ bounding the *size* of C_n, i.e. the number of gates in C_n, the language L is in the class $\mathsf{P/poly}$.

If, in addition, the function $n \mapsto C_n$ is computable in *polynomial time*, L is in P.

Note: For these classes it makes no difference whether the circuits only use $\{\wedge, \vee, \neg\}$ or a richer basis with *threshold* or *majority* gates.

Circuit Lower Bounds

It is conjectured that NP $\not\subseteq$ P/poly.

Lower bound results have been obtained by putting further restrictions on the circuits:

- No *constant-depth* (unbounded fan-in), *polynomial-size* family of circuits decides *parity*. **(Furst, Saxe, Sipser 1983)**.
- No *polynomial-size* family of *monotone* circuits decides *clique*. **(Razborov 1985)**.
- No *constant-depth*, $O(n^{\frac{k}{4}})$-*size* family of circuits decides *k-clique*. **(Rossman 2008)**.

No known result separates NP from *constant-depth*, *polynomial-size* families of circuits with *majority gates*.

Circuits for Graph Properties

We want to study families of circuits that decide properties of *graphs* (or other relational structures—for simplicity of presentation we restrict ourselves to graphs).

We have a family of Boolean circuits $(C_n)_{n\in\omega}$ where there are n^2 inputs labelled $(i,j) : i,j \in [n]$, corresponding to the *potential edges*.
Each input takes value 0 or 1;

Graph properties in P are given by such families where:

- the size of C_n is bounded by a polynomial $p(n)$; and
- the family is uniform, so the function $n \mapsto C_n$ is in P (or DLogTime).

Invariant Circuits

C_n is *invariant* if, for every input graph, the output is unchanged under a permutation of the inputs induced by a permutation of $[n]$.

That is, given any input $G : [n]^2 \to \{0, 1\}$, and a permutation $\pi \in S_n$,

C_n *accepts* G *if, and only if,* C_n *accepts the input* πG *given*

$$(\pi G)(i, j) = G(\pi(i), \pi(j)).$$

Note: this is not the same as requiring that the result is invariant under *all* permutations of the input. That would only allow us to define functions of the *number* of 1s in the input. The functions we define include all *isomorphism-invariant* graph properties such as *connectivity*, *perfect matching*, *Hamiltonicity*, *3-colourability*.

Symmetric Circuits

Say C_n is *symmetric* if any permutation of $[n]$ applied to its inputs can be extended to an automorphism of C_n.

> *i.e., for each $\pi \in S_n$, there is an automorphism of C_n that takes input (i, j) to $(\pi i, \pi j)$.*

Any symmetric circuit is invariant, but *not* conversely.

> *Consider the natural circuit for deciding whether the number of edges in an n-vertex graph is even.*

Any invariant circuit can be converted to a symmetric circuit, but with potentially *exponential blow-up*.