Logic *for* algorithms

R. Ramanujam

The Institute of Mathematical Sciences, Chennai (Retd)

Azim Premji University, Bengaluru (Visiting)

jam@imsc.res.in

ramanujam.r@apu.edu.in

First words

I thank Aditya and the other organisers of this conference for giving me this opportunity.

- Statutory Warning: My expertise is mainly in decidability theory and am only a student in the study of descriptive complexity of graphs.
- Please do feel free to interrupt at any time.

Logic

What is logic?

- What do you know of logic?

Logic

What is logic?

- What do you know of logic?
- Do you think logic is useful?

Logic

What is logic?

- What do you know of logic?
- Do you think logic is useful?
- Do you think logic is useful for theoretical computer science?

Algorithms

This winter school is mainly on algorithmic techniques.

- Have you learnt algorithms formally in a course (or two)?

Algorithms

This winter school is mainly on algorithmic techniques.

- Have you learnt algorithms formally in a course (or two)?
- What are the algorithmic techniques you know of / remember?

Algorithms

This winter school is mainly on algorithmic techniques.

- Have you learnt algorithms formally in a course (or two)?
- What are the algorithmic techniques you know of / remember?
- What are your core insights from algorithm theory?

Theoretical computer science

What is theoretical computer science?

- Tell me some theorems in tcs.

Theoretical computer science

What is theoretical computer science?

- Tell me some theorems in tcs.
- Do you know any theorems on algorithms?

Theoretical computer science

What is theoretical computer science?

- Tell me some theorems in tcs.
- Do you know any theorems on algorithms?
- Do you know any theorems on algorithms involving logic?

Algorithmic metatheorems

This industry was inaugurated by Ron Fagin in 1973.

- Theorem (Fagin 74): A problem is in class *NP* iff it has a description in existential second order logic.

Algorithmic metatheorems

This industry was inaugurated by Ron Fagin in 1973.

- Theorem (Fagin 74): A problem is in class NP iff it has a description in existential second order logic.
- Theorem (Immerman, Vardi 80): A problem is in class P iff it has a description in first order logic + LFP.

Algorithmic metatheorems

This industry was inaugurated by Ron Fagin in 1973.

- Theorem (Fagin 74): A problem is in class NP iff it has a description in existential second order logic.
- Theorem (Immerman, Vardi 80): A problem is in class P iff it has a description in first order logic + LFP.
- Thus the $P = NP$? question has a formulation in logic as well.

Algorithmic metatheorems

This industry was inaugurated by Ron Fagin in 1973.

- Theorem (Fagin 74): A problem is in class NP iff it has a description in existential second order logic.
- Theorem (Immerman, Vardi 80): A problem is in class P iff it has a description in first order logic + LFP.
- Thus the $P = NP$? question has a formulation in logic as well.
- Since then there have been many logical descriptions of complexity classes. This area of study is broadly termed descriptive complexity theory.

A very live area

Here is a result from two months ago!

- Theorem (Carmosino, Fagin, Immerman, Kolaitis, Lenchner, Sengupta MFCS2024): Every Boolean function on n-bit inputs can be defined by a sentence in first order logic having $(1+\epsilon)nlog(n) + O(1)$ quantifiers, and that this is essentially tight. This number reduces to $(1+\epsilon)log(n) + O(1)$ when the Boolean function in question is sparse.
- The proof proceeds by studying winning strategies in a class of two-player combinatorial games called multi-structural games.

In this talk

We will discuss a particular type of algorithmic metatheorems.

- The general pattern: If the problem can be described in a certain logic and the input can be decomposed in a certain way, then there is a certain kind of algorithm for it.

In this talk

We will discuss a particular type of algorithmic metatheorems.

- The general pattern: If the problem can be described in a certain logic and the input can be decomposed in a certain way, then there is a certain kind of algorithm for it.
- The classic: Theorem (Courcelle 90): If the problem can be described in Monadic second order logic and the input has tree width at most k, then there is a linear time algorithm for it.

A *real life* problem

I live in an apartment in a block of flats. Typically every flat has a WiFi router.

- Adjacent routers interfere, and we wish to link routers so that no two adjacent ones interfere.

A *real life* problem

I live in an apartment in a block of flats. Typically every flat has a WiFi router.

- Adjacent routers interfere, and we wish to link routers so that no two adjacent ones interfere.
- The abstract problem, of course, is 3-colourability. What do you know about the problem?

A *real life* problem

I live in an apartment in a block of flats. Typically every flat has a WiFi router.

- Adjacent routers interfere, and we wish to link routers so that no two adjacent ones interfere.
- The abstract problem, of course, is 3-colourability. What do you know about the problem?
- It is NP-complete and unless $\mathsf{P} = \mathsf{NP}$, we cannot solve it efficiently.

A *real life* problem

I live in an apartment in a block of flats. Typically every flat has a WiFi router.

- Adjacent routers interfere, and we wish to link routers so that no two adjacent ones interfere.
- The abstract problem, of course, is 3-colourability. What do you know about the problem?
- It is NP-complete and unless $\mathsf{P} = \mathsf{NP}$, we cannot solve it efficiently.
- But sometimes, we can apply a technique like divide and conquer to 3-colourability!

A good case

Sometimes divide and conquer can be applied to 3-colourability.

- Pick two appropriate vertices and consider the 6 possible colourings.
- We can think of a vertex defending a region of the graph.
- We can then solve the problem recursively on the independent parts.

A good case

Sometimes divide and conquer can be applied to 3-colourability.

- Pick two appropriate vertices and consider the 6 possible colourings.
- We can think of a vertex defending a region of the graph.
- We can then solve the problem recursively on the independent parts.
- Over to the board!

A game

We can think of this as a game of Cops and robbers, played on a graph.

- k cops try to catch a robber by being on the same vertex as her.

A game

We can think of this as a game of Cops and robbers, played on a graph.

- k cops try to catch a robber by being on the same vertex as her.
- First the cops, then the robber pick a start vertex to occupy,

A game

We can think of this as a game of Cops and robbers, played on a graph.

- k cops try to catch a robber by being on the same vertex as her.
- First the cops, then the robber pick a start vertex to occupy,
- A cop gets on a helicopter and heads towards some vertex.

A game

We can think of this as a game of Cops and robbers, played on a graph.

- k cops try to catch a robber by being on the same vertex as her.
- First the cops, then the robber pick a start vertex to occupy,
- A cop gets on a helicopter and heads towards some vertex.
- Meanwhile, the robber moves along any path of unoccupied vertices.

An abstraction

The cops' strategy = a tree decomposition of the graph!

- The nodes are positions of the cops.
- The root is their initial position.
- The children of a tree node are the graph components that could contain the robber.

Why we want metatheorems

When restricted to graphs of bounded treewidth, all the following problems are in NC:

- Vertex cover, feedback vertex set, minimum maximal matching, clique, independent set ...
- Partition into – triangles, isomorphic subgraphs, Hamiltonian subgraphs, forests, cliques, perfect matchings, ...
- For fixed H – subgraph isomorphism, graph homomorphism, path with forbidden pairs, ...
- Degree constrained spanning tree, bounded diameter spanning tree, max cut, chromatic index, chordal graph completion for fixed k, ...

Tree decompositions

Nice work, if you can get it!

- . . . Courcelle's theorem: and if you get it, who could ask for anything more?

What's common?

All these problems share one very nice property.

- In 1990, Courcelle noticed that all of these problems can be described in monadic second order logic (*MSO* logic).
- This is the familiar language of quantifiers that you all know, extended with quantification over sets.
- The logic is defined over graphs, so that $\forall x$ means universal quantification over graph vertices and $\exists X$ means existential quantification over subsets of graph vertices.

What's common?

All these problems share one very nice property.

- In 1990, Courcelle noticed that all of these problems can be described in monadic second order logic (*MSO* logic).
- This is the familiar language of quantifiers that you all know, extended with quantification over sets.
- The logic is defined over graphs, so that $\forall x$ means universal quantification over graph vertices and $\exists X$ means existential quantification over subsets of graph vertices.
- Over to the board!

Courcelle's theorem

Let ϕ be an MSO formula and k a number. Then $\{G \mid G \models \phi$, and $tw(G) \leq k\}$ can be decided in linear time.

- Compute a tree decomposition of G in time linear in the size of G (and some computable function of k).

Courcelle's theorem

Let ϕ be an MSO formula and k a number. Then $\{G \mid G \models \phi, \text{ and } tw(G) \leq k\}$ can be decided in linear time.

- Compute a tree decomposition of G in time linear in the size of G (and some computable function of k).
- Adjust the formula so that it applies to the tree.

Courcelle's theorem

Let ϕ be an MSO formula and k a number. Then $\{G \mid G \models \phi, \text{ and } tw(G) \leq k\}$ can be decided in linear time.

- Compute a tree decomposition of G in time linear in the size of G (and some computable function of k).
- Adjust the formula so that it applies to the tree.
- Transform the formula into a finite state tree automaton.

Courcelle's theorem

Let ϕ be an MSO formula and k a number. Then $\{G \mid G \models \phi$, and $tw(G) \leq k\}$ can be decided in linear time.

- Compute a tree decomposition of G in time linear in the size of G (and some computable function of k).
- Adjust the formula so that it applies to the tree.
- Transform the formula into a finite state tree automaton.
- The formula is true on G iff A_{T_G} reaches an accepting state on T_G. (This latter property is linear time checkable,)

Refinements

Once you have Courcelle's theorem, you can do better.

- Theorem(Bodlaender, Courcelle): If the problem can be described in MSO logic and the input has bounded tree width, then there is a parallel algorithm running in time O(log n).
- Theorem(Bodlaender, Courcelle): If the problem can be described in MSO logic and the input has bounded clique width, then there is a polynomial time algorithm for it.
- Theorem(Frick, Grohe): If the problem can be described in FO logic and the input is planar, then there is a linear time algorithm for it.
- Theorem(Flum, Grohe): If the problem can be described in FO logic and the input has a forbidden minor, then there is a polynomial time algorithm for it.

Recent results

While these theorems yield tight upper time bounds, they yield no completeness results.

- Recent metatheorems concern space complexity and circuit complexity.
- They also yield completeness results.

A space result

$L =$ problems solvable by deterministic Turning machines, using work space of size only logarithmic in the input size.

- Courcelle's theorem can be refined to show that over graphs of bounded treewidth, MSO-definable properties are in L.
- Corollary: Over graphs of bounded treewidth, his gives logspace algorithms for 3-colourability, perfect matching, reachability from a source vertex, ...

A space result

$L =$ problems solvable by deterministic Turning machines, using work space of size only logarithmic in the input size.

- Courcelle's theorem can be refined to show that over graphs of bounded treewidth, MSO-definable properties are in L.
- Corollary: Over graphs of bounded treewidth, his gives logspace algorithms for 3-colourability, perfect matching, reachability from a source vertex, ...
- Each of these results would have been a paper not too long ago.

New structure theorems and algorithmic applications

Thomassen has shown that all graphs of sufficiently large tree width have a cycle whose length is a multiple of 3.

- This suggests the following algorithm for checking whether an arbitrary graph G has a cycle whose length is a multiple of 3.
- Check, whether G has small tree width. If yes, subdivide all edges and apply the metatheorem to the formula specifying the property.
- Otherwise say "yes".

Quantifier classes

Once we relate logic and algorithms, there are finer details to consider.

- Fagin's theorem shows that a property is in NP iff it is of the form $\exists X \exists Y \exists Z \forall u \forall v \phi$.

Quantifier classes

Once we relate logic and algorithms, there are finer details to consider.

- Fagin's theorem shows that a property is in NP iff it is of the form $\exists X \exists Y \exists Z \forall u \forall v \phi$.
- What about properties of the form $\exists X \exists Y \exists Z \forall u \exists v$?
- What about properties of the form $\exists X \exists Y \forall u \forall v$?
- What about properties of the form $\exists X \forall u \exists v \forall w$?
- and so on.

One quantifier class

Theorem (Gottlob, Kolaitis, Schwentick, 2004): $\exists_1^*\forall\exists$ formulas describe properties checkable in polynomial time over undirected simple graphs.

- Preprocess the input graph.
- If the treewidth is small, apply Courcelle's theorem.
- If not, check whether a special cycle of constant length exists.
- Otherwise say yes.
- This is a very difficult proof, running to about 35 pages.

Developments

Results on $\exists_1^*\forall\exists$ formulas have now been refined to logspace.

- Further quantifier classes have been characterized for other complexity classes.
- Examples: $\exists_1\forall\exists$ in AC_0, $\exists_1^2\forall\exists$ in L, $\exists_1\exists^*\forall\exists$ in NL, $\exists_1^2\forall\forall$ in NP, and so on.
- The proofs show how logical definability is a key constraint in our ability to formulate algorithms.

A range of applications

A similar argument shows that there are space efficient pseudo-polynomial time algorithms for:

- knapsack problems,
- bin packing problems,
- scheduling problems, and
- integer programming for a fixed number of inequalities.

In conclusion

The logic – algorithms connection is deeply insightful, and gives us uniform results on existence of algorithms for checking a wide ranging class of properties.

- Now there are algorithmic metatheorems for constant depth circuits.
- Open: Are there space efficient metatheorems for graphs of bounded clique-width?
- Algorithmic theory of nowhere dense graphs, sparse graphs, . . .
- We found the use of tree automata here, what about automata running on graphs, and their connection to logic?

In conclusion

The logic – algorithms connection is deeply insightful, and gives us uniform results on existence of algorithms for checking a wide ranging class of properties.

- Now there are algorithmic metatheorems for constant depth circuits.
- Open: Are there space efficient metatheorems for graphs of bounded clique-width?
- Algorithmic theory of nowhere dense graphs, sparse graphs, . . .
- We found the use of tree automata here, what about automata running on graphs, and their connection to logic?
- Welcome to the study of logic and complexity theory.