HornSat and φ-Graphs

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

Seminar Presentation

Lake Superior State University

Satisfiability and Validity

- Given a Boolean expression φ and a truth assignment T to its variables (said to be **appropriate to** φ), we write $T \models \varphi$ if the truth value of φ under T is TRUE.
- We say that a Boolean expression φ is **satisfiable** if there is a truth assignment T appropriate to it such that $T \models \varphi$.
- We say that φ is **valid** or a **tautology**, written $\models \varphi$, if $T \models \varphi$ for all T appropriate to φ.

Proposition

A Boolean expression is unsatisfiable if and only if its negation is valid.

$$\begin{aligned} \varphi \text{ is unsatisfiable} \quad & \text{iff} \quad \text{for all } T,\ T \not\models \varphi \\ & \text{iff} \quad \text{for all } T,\ T \models \neg\varphi \\ & \text{iff} \quad \neg\varphi \text{ is valid.} \end{aligned}$$

Examples

- Consider $\varphi = ((x_1 \vee \neg x_2) \wedge \neg x_1)$.

  φ is satisfiable, satisfied by

  $$T(x_1) = T(x_2) = \text{FALSE}.$$

- Consider

  $$\varphi = ((x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)).$$

  φ is not satisfiable.

  - Since it is in conjunctive normal form, a satisfying truth assignment has to satisfy all clauses.
  - The first clause requires that one of $T(x_1)$, $T(x_2)$, $T(x_3)$ be TRUE.
  - The next three clauses require that all three values be the same.
  - Finally, the last clause requires that one of the three values be false.

  Once we have established this, we also know that $\neg\varphi$ is valid.

Satisfiability

- A Boolean expression can be represented for processing by an algorithm as a string over an alphabet that contains the symbols

$$x, 0, 1, (,), \vee, \neg, \wedge.$$

- The **length of a Boolean expression** is the length of the corresponding string.
- Satisfiability (or Sat, for short) is the following problem:

  Given a Boolean expression φ in conjunctive normal form (a conjunction of disjunctions, called **clauses**), is it satisfiable?
- We require that the expression be given in conjunctive normal form for two reasons:
  - We know that all expressions can be so represented.
  - This special form seems to capture the intricacy of the whole problem.

Satisfiability is in NP

- Sat can be solved in $O\left(n^2 2^n\right)$ time by an exhaustive algorithm:
  - Try all possible combinations of truth values for the variables that appear in the expression;
  - Report "yes" if one of them satisfies it, and "no" otherwise.
- Sat can be very easily solved by a nondeterministic polynomial algorithm:
  - Guess the satisfying truth assignment;
  - Check that it indeed satisfies all clauses.

  Therefore Sat is in NP.
- Presently we do not know whether Sat is in P (and, since it is NP-complete, we strongly suspect that it is not).

Horn Clauses

- A clause is a **Horn clause** if it has at most one positive literal.

  I.e., all its literals, except possibly for one, are negations of variables.
- The following clauses are Horn:

$$(\neg x_2 \vee x_3), \quad (\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_4), \quad (x_1).$$

  Of these clauses:
  - the second is a **purely negative** clause (it has no positive literals);
  - the rest do have a positive literal, and are called **implications**.
- They are so called because they can be rewritten as

$$(x_2 \to x_3), \quad (\mathrm{TRUE} \to x_1)$$

  (in the last clause, the conjunction of no variables is taken to be the "expression" TRUE).

The Problem HornSat

- The problem HornSat is similar to Sat, but restricted to inputs that consist of Horn clauses.
- Formally HornSat is the following problem:

  Given a Boolean expression φ in conjunctive normal form, where all clauses are Horn clauses, is it satisfiable?

HornSat is in P

Theorem

HornSat is in P.

- We consider a truth assignment as the set T of TRUE variables.
  Let φ, be a conjunction of Horn clauses.
  Initially, we only consider the implications of φ.
  $T := \emptyset$;
  Repeat the following step, until all implications are satisfied:
  Pick any unsatisfied $((x_1 \wedge \cdots \wedge x_m) \rightarrow y)$ and add y to T.
- This algorithm will terminate, since T gets bigger at each step.
- The truth assignment obtained must satisfy all implications in φ, since this is the only way for the algorithm to end.
- If T' also satisfies all implications in φ, then $T \subseteq T'$.
  - If not, consider the first time during the execution of the algorithm at which T ceased being a subset of T'.
  - The clause that caused this insertion to T cannot be satisfied by T'.

Proof (Cont'd)

- We can now determine the satisfiability of the whole expression φ.

  Claim: φ is satisfiable if and only if the truth assignment T obtained by the algorithm satisfies φ.

  Suppose that there is a purely negative clause of φ that is not satisfied by T, say

  $$(\neg x_1 \vee \neg x_2 \vee \cdots \vee \neg x_m).$$

  Thus $\{x_1, x_2, \ldots, x_m\} \subseteq T$.
  It follows that no superset of T can satisfy this clause.
  But we showed in the preceding slide that all truth assignments that satisfy φ are supersets of T.

- The algorithm can obviously be carried out in polynomial time.

It follows that HornSat is in P.

First-Order Expressions Over Σ_{Graph}

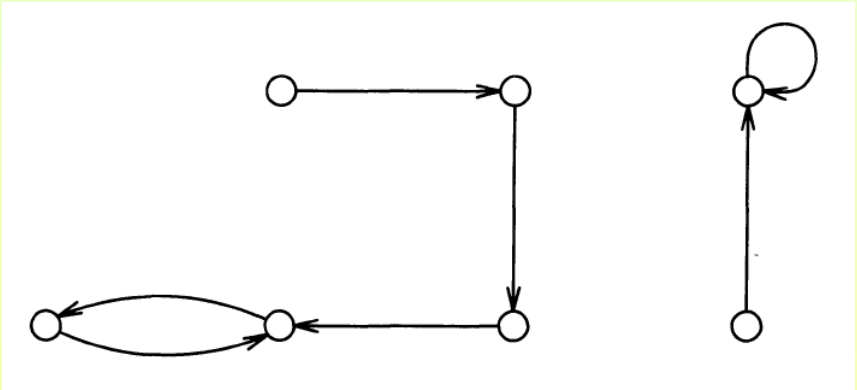
- The vocabulary Σ_{Graph} of graph theory is intended to express first-order properties of graphs.
- The **vocabulary of graph theory**, $\Sigma_{\text{Graph}} = (\Phi_{\text{Graph}}, \Pi_{\text{Graph}}, r_{\text{Graph}})$, consists of:
  - No function symbols, i.e., $\Phi_{\text{Graph}} = \emptyset$;
  - A single binary relation (besides $=$), called G, i.e., $\Pi_{\text{Graph}} = \{G\}$ and $r_{\text{Graph}}(G) = 2$.
- Typical expressions in graph theory are:
  - $G(x, x)$;
  - $\exists x(\forall y G(y, x))$;
  - $\forall x(\forall y(G(x, y) \rightarrow G(y, x)))$;
  - $\forall x(\forall y(\forall z((G(x, z) \wedge G(z, y)) \rightarrow G(x, y))))$.

Example

- Any model appropriate to Σ_{Graph} is a graph.
- We shall be interested only in finite graphs (omitting "finite").
- Consider

$$\varphi_1 = (\forall x \exists y G(x,y) \wedge \forall x \forall y \forall z((G(x,y) \wedge G(x,z)) \rightarrow y = z)).$$

- A model Γ for φ_1 is shown in the figure.



- What other graphs satisfy φ_1?

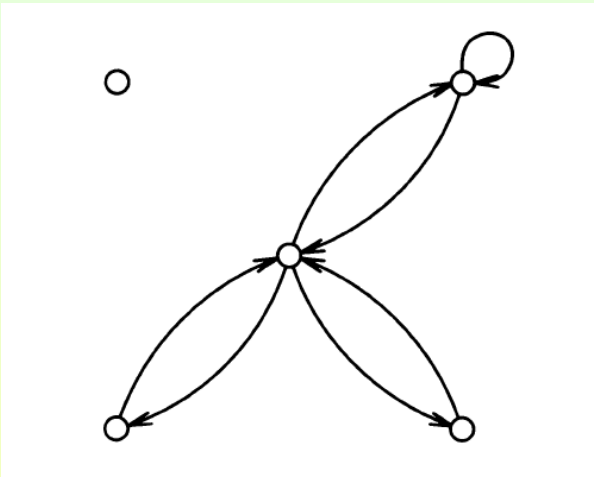
  We claim φ_1 is a sentence that essentially states "G is a function".

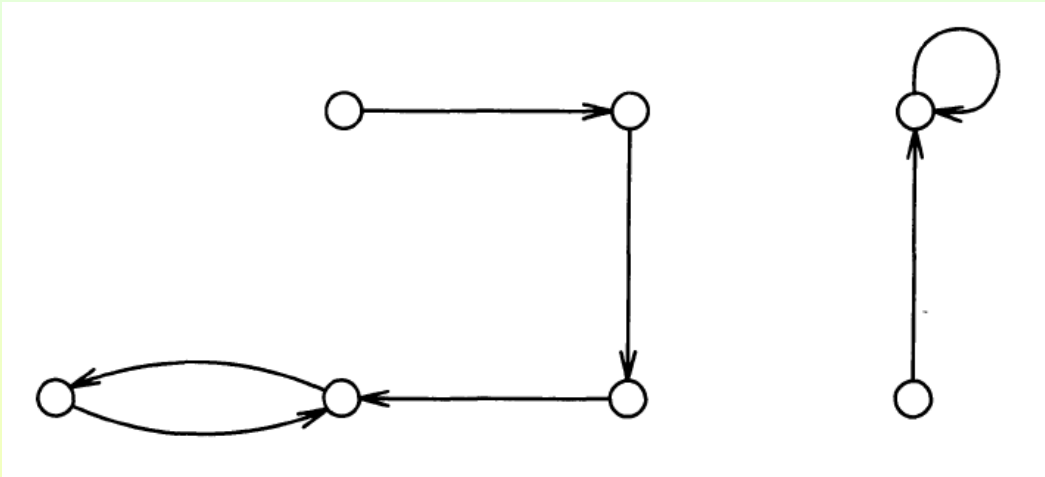
Example

- Consider now the sentence

$$\varphi_2 = \forall x(\forall y(G(x,y) \to G(y,x))).$$

The graph on the left satisfies φ_2; the one on the right does not.





φ_2 is the statement "G is symmetric".

- The sentence $\forall x(\forall y(\forall z(G(x,z) \wedge G(z,y)) \to G(x,y)))$ states "G is transitive".
- Notice that all these three graph properties (outdegree one, symmetry and transitivity) can be checked in polynomial time.

The Problem φ-Graphs

- Each sentence in graph theory describes a property of graphs.
- Each property of graphs corresponds in turn to a computational problem:

  Given a graph G, does it have the property?
- Let φ be any expression over Σ_{Graph} (not necessarily a sentence).
- Define the following problem φ-Graphs:

  Given a model Γ for φ (i.e., a graph G_Γ together with an assignment of nodes of G_Γ to the free variables in φ), does $\Gamma \models \varphi$?
- E.g., if $\varphi = \forall x(\forall y(G(x,y) \rightarrow G(y,x)))$, φ-Graphs is Symmetry, the computational problem of deciding whether a given graph is symmetric.

φ-GRAPHS is in P

Theorem

For any expression φ over Σ_{Graph}, the problem φ-GRAPHS is in P.

- The proof is by induction on the structure of φ.

Base The statement is obviously true when φ is an atomic expression of the form $G(x, y)$ or $G(x, x)$.

Step Suppose $\varphi = \neg\psi$. By induction there is a polynomial algorithm that solves the problem ψ-GRAPHS. The same algorithm, with answer reversed from "yes" to "no" and vice-versa, solves φ-GRAPHS.

Suppose $\varphi = \psi_1 \vee \psi_2$. By induction there are polynomial algorithms that solve ψ_1-GRAPHS and ψ_2-GRAPHS. Our algorithm for φ-GRAPHS executes these algorithms one after the other, and replies "yes" if at least one of them answers "yes". The running time of this algorithm is the sum of the two polynomials, and thus a polynomial.

Similarly for $\varphi = \psi_1 \wedge \psi_2$.

Proof (Conclusion)

- Finally, suppose that $\varphi = \forall x \psi$.

  By induction, there is a polynomial time algorithm for ψ-GRAPHS.

  Our algorithm for φ-GRAPHS does the following:

  Repeat the following for each node v of G_Γ

  Given the alleged model Γ for φ, attach the value $x = v$ to Γ to produce a model for ψ (recall that Γ contains no value for x, as x is bound in φ).

  Test whether the resulting model satisfies ψ.

  Answer "yes" if the answer is "yes" for all vertices v.

  Answer "no" otherwise.

  The overall algorithm is polynomial, with a polynomial which equals that for ψ-GRAPHS times n, the number of nodes in the universe of Γ.

Thank you!

- In closing...

Thank you for your Attention!!