Computation and Inference

N. Shankar

Computer Science Laboratory
SRI International
Menlo Park, CA

July 13, 2018

Length of the Longest Increasing Subsequence

- You have a sequence of numbers, e.g.,
  9, 7, 10, 9, 5, 4, 10.
- The task is to find the length of the longest increasing subsequence.
- Here the longest subsequence is 7, 9, 10, and its length is 3.
- Patience solitaire is a card game where cards are placed, one by one, into a sequence of columns.
- Each card is placed at the bottom of the leftmost column where it is no bigger than the current bottom card in the column.
- If there is no such column, we start a new column at the right.
- Show that the number of columns left at the end yields the length of the longest increasing subsequence.

Computation and Inference

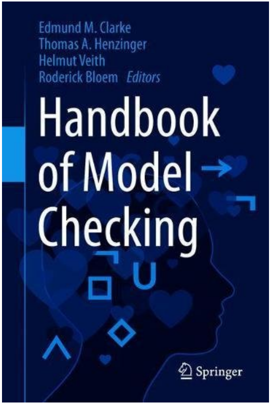
- Computing, like mathematics, is the study of reusable abstractions.
- Abstractions in computing include numbers, lists, channels, processes, protocols, and programming languages.
- Computing is *abstraction engineering* and logic is the calculus of computing.
- Inference is a mechanism for generating knowledge about abstractions through abduction, deduction, and induction.
- Inference algorithms employ inference steps to perform computation as solvability-preserving problem transformations.
- *Algorithm = Inference + Strategy + Indexing*

The Unreasonable Effectiveness of Logic in Computing

- Beyond computing, the world itself is increasingly an interplay of abstractions.
- Caches, files, IP addresses, avatars, friends, likes, hyperlinks, packets, network protocols, and cyber-physical systems are all examples of abstractions in daily use.
- Such abstract entities and the relationships can be expressed clearly and precisely in logic.
- In computing, and elsewhere, we are increasingly dependent on formalization as a way of managing the abstract universe.
- Logic has been *unreasonably* effective in computing, with an impact that spans theory of computation, hardware design, software verification, databases, programming languages, artificial intelligence, and systems biology.

Overview

We give a brief introduction to the mechanisms of deductive inference based on an article in the recent *Handbook of Model Checking* (eds. Clarke, Henzinger, Veith, and Bloem).



- These deduction techniques are used heavily in hardware and software specification and verification.
- But they have broad applicability beyond verification.

Language, Meaning, Proof: The Logical Trinity

- Logic studies the *trinity* between *language*, *interpretation*, and *proof*.
- *Language*: What can be expressed?
- *Interpretation*: What is the intended meaning?
  - Meaning is usually *compositional*: Follows the syntax
  - Some symbols have fixed interpretation: connectives, equality, quantifiers
  - The interpretation of other symbols is allowed to vary variables, functions, and predicates
  - *Assertions* either hold or fail to hold in a given interpretation
  - A *valid* assertion holds in every interpretation
  - If an assertion is not valid, then its negation is *satisfiable*
- *Proofs*: How do we constructively demonstrate the validity of a statement?

- **Language:** The syntactic representation of conditions is using propositional formulas:

$$\phi := P \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \Rightarrow \phi_2$$

- Examples of formulas are p, $p \wedge \neg p$, $p \vee \neg p$, $(p \wedge \neg q) \vee \neg p$.
- **Interpretation:** $M[\![\phi]\!]$ is the meaning of ϕ in interpretation M and is computed using truth tables:

ϕ	p	q	$\neg p$	$p \vee q$	$p \wedge q$	$p \Rightarrow q$
$M_1(\phi)$	$\perp$	$\perp$	$\top$	$\perp$	$\perp$	$\top$
$M_2(\phi)$	$\perp$	$\top$	$\top$	$\top$	$\perp$	$\top$
$M_3(\phi)$	$\top$	$\perp$	$\perp$	$\top$	$\perp$	$\perp$
$M_4(\phi)$	$\top$	$\top$	$\perp$	$\top$	$\top$	$\top$

A formula ϕ is *satisfiable* if $M \models \phi$ for some M, and is *unsatisfiable*, otherwise.

The basic judgement is $\Gamma \vdash \Delta$ asserting the validity of $\bigwedge \Gamma \Rightarrow \bigvee \Delta$, where Γ and Δ are sets (or bags) of formulas.

	Left	Right
Ax	$\dfrac{}{\Gamma, A \vdash A, \Delta}$	
$\neg$	$\dfrac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}$	$\dfrac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}$
$\vee$	$\dfrac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}$	$\dfrac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}$
$\wedge$	$\dfrac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}$	$\dfrac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$
$\Rightarrow$	$\dfrac{\Gamma, B \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}$	$\dfrac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}$
Cut	$\dfrac{\Gamma \vdash A, \Delta \quad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}$	

What is an Inference Algorithm?

- We want a systematic way to construct and analyze decision procedures for the satisfiability of formulas in a given class.
- An *Σ-inference structure* $\langle \Psi, \vdash, \Lambda, \mathcal{M} \rangle$ consists of
  - Ψ, a set of *logical states*
  - $\vdash$, the *reduction relation* between states
  - Λ, a map from states to Σ-formulas
  - $\mathcal{M}$, which extracts models from canonical states
- An *inference system* is an inference structure that is
  - *Conservative*: If $\psi \vdash \psi'$, then $\Lambda(\psi)$ and $\Lambda(\psi')$ are equisatisfiable.
  - *Progressive*: $\vdash$ is well-founded.
  - *Canonizing*: If $\psi \not\vdash \psi'$ for any ψ', then either ψ is $\bot$ (i.e., unsatisfiable) or ψ is in a canonical form so that $\mathcal{M}(\psi)$ is a model for $\Lambda(\psi)$.
- An *inference algorithm* is a collection of effective *inference rules* defining the reduction relation.

Ordered Resolution as an Inference Algorithm

- A *clause* is a disjunction $l_1 \vee \ldots \vee l_n$ of literals, where each *literal* l is a propositional variable p or its negation $\neg p$.
- Any propositional formula has an equisatisfiable representation in *conjunctive normal form* as a conjunction of clauses.
- Ordered resolution is an algorithm for CNF satisfiability.
- Input K is a set of ordered clauses (disjunctions of literals p or $\neg p$), where the literals appear in decreasing order w.r.t. some order e.g., $q \prec \neg q \prec p \prec \neg p$.
- Tautologies, i.e., clauses containing both p and $\neg p$, are deleted from initial input.

Res	$\dfrac{K, p \vee \Gamma_1, \neg p \vee \Gamma_2}{K, p \vee \Gamma_1, \neg p \vee \Gamma_2, \Gamma_1 \vee \Gamma_2}$ $\Gamma_1 \vee \Gamma_2 \notin K$, $\Gamma_1 \vee \Gamma_2$ is not tautological
Contrad	$\dfrac{K, p, \neg p}{\bot}$

Ordered Resolution as an Inference Algorithm

$$\cfrac{\cfrac{\cfrac{\cfrac{(K_0 =)\ {\color{red}\neg p} \vee \neg q \vee r, \quad \neg p \vee q, \quad {\color{red}p} \vee r, \quad \neg r}{(K_1 =)\ \neg q \vee r, \quad K_0}\ \textbf{Res}}{(K_2 =)\ q \vee r, \quad K_1}\ \textbf{Res}}{(K_3 =)\ r, \quad K_2}\ \textbf{Res}}{\bot}\ \textbf{Contrad}$$

- Drop the clause $\neg r$, and we reach an irreducible state from which a truth assignment $\{r \mapsto \top, q \mapsto \bot, p \mapsto \bot\}$ can be constructed.

Ordered Resolution as an Inference Algorithm

- The resolution inference system is *strongly conservative* (i.e., model-preserving, not just satisfiability-preserving): $\Gamma_1 \vee \Gamma_2$ is satisfiable if $p \vee \Gamma_1$ and $\neg p \vee \Gamma_2$ are.
- It is *progressive*: Bounded number of new clauses in the input variables.
- It is *canonizing*: Build a model M by assigning to atoms p_1 to p_n within a series of partial assignments $M_0, \ldots, M_n$:
  - M_0 is the empty truth assignment.
  - $M_{i+1} = M_i[p_{i+1} \mapsto v]$, where $v = \top$ iff there is some clause $p_{i+1} \vee \Gamma$ in the irreducible state K such that $M_i \models \neg\Gamma$.
- If $M_i \models \neg\Gamma$, then for any clause $\neg p_i \vee \Delta$, $M_i \models \Delta$ since $\Gamma \vee \Delta \in K$.
- **Invariant:** $M_i \models \Gamma$ for all clauses Γ in K in the atoms $p_1, \ldots, p_i$.

Boolean Satisfiability (SAT) using CDCL

Conflict-Driven Clause Learning (CDCL) builds a partial assignment through guessing (Select), propagation (Propagate), and conflict-triggered retraction (Backjump).

Name	Rule	Condition
Propagate	$\dfrac{h, \langle M \rangle, K, C}{h, \langle M, l[\Gamma] \rangle, K, C}$	$\Gamma \equiv l \vee \Gamma' \in K \cup C$ $M \models \neg\Gamma'$
Select	$\dfrac{h, \langle M \rangle, K, C}{h+1, \langle M; l[] \rangle, K, C}$	$M \not\models l$ $M \not\models \neg l$
Conflict	$\dfrac{0, \langle M \rangle, K, C}{\bot}$	$M \models \neg\Gamma$ for some $\Gamma \in K \cup C$
Backjump	$\dfrac{h+1, \langle M \rangle, K, C}{h', \langle M_{\leq h'}, l[\Gamma'] \rangle, K, C \cup \{\Gamma'\}}$	$M \models \neg\Gamma$ for some $\Gamma \in K \cup C$ $\langle h', \Gamma' \rangle$ $= \mathit{analyze}(\psi)(\Gamma)$ for $\psi = h, \langle M \rangle, K, C$

CDCL state consists of a decision level h, a partial assignment $\langle M \rangle$ (a list of decision literals and derived literals with justification clauses), the input clause set K, and the learned clause set C.

CDCL Example

- Let K be $\{p \vee q, \neg p \vee q, p \vee \neg q, s \vee \neg p \vee q, \neg s \vee p \vee \neg q, \neg p \vee r, \neg q \vee \neg r\}$.
- The partial assignment is extended through selection and propagation until a $K \cup C$ contains a conflict, a clause that is falsified by the current partial assignment.

step	h	M	K	C	Γ
select s	1	$; s$	K	$\emptyset$	_
select r	2	$; s; r$	K	$\emptyset$	_
propagate	2	$; s; r, \neg q[\neg q \vee \neg r]$	K	$\emptyset$	_
propagate	2	$; s; r, \neg q, p[p \vee q]$	K	$\emptyset$	_
conflict	2	$; s; r, \neg q, p$	K	$\emptyset$	$\neg p \vee q$

CDCL Example (contd.)

The conflict clause $p \vee q$ is *analyzed* by resolving it against the justification clauses for assignments at the current level (i.e., level 2) until there is a unique literal at the current level.

step	h	M	K	C	Γ
conflict	2	$; s; r, \neg q, p$	K	$\emptyset$	$\neg p \vee q$
backjump	0	$\emptyset$	K	q	_
propagate	0	$q[q]$	K	q	_
propagate	0	$q, p[p \vee \neg q]$	K	q	_
propagate	0	$q, p, r[\neg p \vee r]$	K	q	_
conflict	0	q, p, r	K	q	$\neg q \vee \neg r$

A conflict at level 0 implies that the input clause set is unsatisfiable since there are no decision literals.

Equality Logic (EL)

Language: In addition to propositional atoms, we add a set of constants κ given by $c_0, c_1, \ldots$ and equalities $c = d$ for constants c and d.

$$\phi \quad := \quad P \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \kappa_1 = \kappa_2$$

Interpretation: The structure M now has a domain $|M|$ and maps propositional variables to $\{\top, \bot\}$ and constants to $|M|$.

$$M[\![c = d]\!] \quad = \quad \begin{cases} \top, & \text{if } M[\![c]\!] = M[\![d]\!] \\ \bot, & \text{otherwise} \end{cases}$$

Proof:

Reflexivity	$\Gamma \vdash a = a, \Delta$
Symmetry	$\dfrac{\Gamma \vdash a = b, \Delta}{\Gamma \vdash b = a, \Delta}$
Transitivity	$\dfrac{\Gamma \vdash a = b, \Delta \qquad \Gamma \vdash b = c, \Delta}{\Gamma \vdash a = c, \Delta}$

Maintaining Equivalence with Union-Find

The logical state is a triple $\langle G; F; D\rangle$ with the input equalities and disequalities G, the processed disequalities D, and the find structure F which is a set of oriented equalities, i.e., orient $y = x$ as $x = y$ if $y \prec x$.

Delete	$\dfrac{x = y, G; F; D}{G; F; D}$ if $F(x) \equiv F(y)$
Merge	$\dfrac{x = y, G; F; D}{G; F' \circ F; D}$ if $F(x) \not\equiv F(y)$, $F' = \{orient(F(x) = F(y))\}$
Diseq	$\dfrac{x \neq y, G; F; D}{G; F; x \neq y, D}$
Contrad	$\dfrac{G; F; x \neq y, D}{\perp}$ if $F(x) = F(y)$

- The above inference system is (strongly) conservative, progressive, and canonizing.
- Example: $x = y, y \neq zx = z, u = v; \emptyset; \emptyset$ reduces to $\perp$ through $\emptyset; x = z, y = z, u = v; y \neq z$.

- **Language:** Signature Σ lists allowable function and predicate symbols with their arities.

$$
\begin{array}{rcl}
\tau & := & X \\
& | & f(\tau_1, \ldots, \tau_n), \text{ for } n \geq 0 \\
\phi & := & \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \tau_1 = \tau_2 \\
& & \mid \forall x.\phi \mid \exists x.\phi \mid q(\tau_1, \ldots, \tau_n), \text{ for } n \geq 0
\end{array}
$$

- **Meaning:**

$$
\begin{array}{rcl}
M[\![a = b]\!] & = & M[\![a]\!] = M[\![b]\!] \\
M[\![f(a_1, \ldots, a_n)]\!] & = & (M[\![f]\!])(M[\![a_1]\!], \ldots, M[\![a_n]\!]) \\
M[\![x]\!]\rho & = & \rho(x) \\
M[\![q(a_1, \ldots, a_n)]\!]\rho & = & M[\![q]\!](M[\![a_1]\!]\rho, \ldots, M[\![a_n]\!]\rho) \\
M[\![\forall x.A]\!]\rho & = & \begin{cases} \top, & \text{if } M[\![A]\!]\rho[x := d] \text{ for all } d \in D \\ \bot, & \text{otherwise} \end{cases} \\
M[\![\exists x.A]\!]\rho & = & \begin{cases} \top, & \text{if } M[\![A]\!]\rho[x := d] \text{ for some } d \in D \\ \bot, & \text{otherwise} \end{cases}
\end{array}
$$

Satisfiability Modulo Theories

- A theory in a signature Σ restricts the meaning of the function and predicate symbols.
- For example, the array theory is given by operations *select* and *store* such that
  1. $select(store(A, i, v), i) = v$
  2. $i \neq j \Rightarrow select(store(A, i, v), j) = A(j)$
- SMT deals with formulas with theory atoms like $x = y$, $x \neq y$, $x - y \leq 3$, and $select(store(A, i, v), j) = w$.
- For SMT, the CDCL search state is augmented with a *theory state* S in addition to the partial assignment.
- When a literal is added to M by unit propagation, it is also *asserted* to S.
- When a literal is implied by S, it is *propagated* to M.
- When backjumping, any literals deleted from M are also *retracted* from S.

SMT example

The state extends CDCL with a find structure F and disquality set D.

Input is $y = z, \quad x = y \vee x = z, \quad x \neq y \vee x \neq z$

Step	M	F	D	C
Assert	$y = z$	$\{y \mapsto z\}$	$\emptyset$	$\emptyset$
Select	$y = z; x \neq y$	$\{y \mapsto z\}$	$\{x \neq y\}$	$\emptyset$
Prop	$\ldots, x \neq z$ $[x \neq z \vee y \neq z \vee x = y]$	$\{y \mapsto z\}$	$\{x \neq y\}$	$\emptyset$
Conflict	$\ldots$	$\{y \mapsto z\}$	$\{x \neq y\}$	$\emptyset$
Analyze	$\ldots$	$\{y \mapsto z\}$	$\{x \neq y\}$	$\{y \neq z \vee x = y\}$
Bkjump	$y = z, x = y$	$\{y \mapsto z\}$	$\emptyset$	$\ldots$
Assert	$y = z, x = y$	$\{x \mapsto y, y \mapsto z\}$	$\emptyset$	$\ldots$
Prop	$\ldots, x = z$ $[x = z \vee x \neq y \vee y \neq z]$	$\{x \mapsto y, y \mapsto z\}$	$\emptyset$	$\ldots$
Conflict				

Generalizing Inference Algorithms: GCD

- Let $gcd(x, y, z)$ represent the formula

$$
\begin{array}{ll}
 & x \geq 0 \wedge y \geq 0 \wedge x + y > 0 \wedge z > 0 \\
\wedge & z|x \wedge z|y \\
\wedge & (\forall z' : z'|x \wedge z'|y \Rightarrow z' \leq z)
\end{array}
$$

- There is only one inference step:

$$
\frac{gcd(x, y, z)}{gcd((x \sqcup y) - (x \sqcap y), x \sqcap y, z)} \quad \text{if } x > 0, y > 0
$$

- *Conservative*, because any divisor of x, y such that $x > y > 0$ is also a divisor of $x - y$.
- *Progressive*, because $x + y$ decreases with each inference step.
- *Canonizing*, because if $x = 0$ or $y = 0$, then $z = x \sqcup y$.

Generalizing Inference Algorithms: Sorting

- Given an array A of k integers, such an array is sorted if $\forall i < j < k : (A(i) \not> A(j))$, i.e., there are no inversions.
- The predicate $sort(A, B)$ holds if B is a sorted, permutation of A.
- The initial state is just the input array a, where $\Lambda(a)$ asserts $sort(a, B)$ for a *result* variable B.
- The inference rule is

$$\frac{A}{swap(A, i, j)}\ i < j, A(i) > A(j)$$

- *Conservative*: Each swap step preserves solvability.
- *Progressive*: Each swap is a decreasing step in a lexicographic ordering.
- *Canonizing*: When no swaps are possible, the array is already sorted.

Generalizing Inference Algorithms: Dijkstra

- Given a weighted directed graph $G = (V, E, W)$, with non-negative edge weights, find the smallest-weight path from a given source vertex s to each vertex, i.e., a map P_s on V:
  $P_s(s) = 0$, and for $v \neq s$,
  $P_s(v) = \prod\{P_s(u) + W(u, v) \mid u \in V\}$.
- Let

$$post(X)(v) = \begin{cases} 0, \text{ if } v = s \\ \prod\{X(u) + W(u, v) \mid u \in dom(X)\}, otherwise. \end{cases}$$

- We therefore want to compute P_s such that $P_s = post(P_s)$.

Generalizing Inference Algorithms: Dijkstra

- The logical state has two partial maps D and Q:
  1. Each $v \in V$ is either in $dom(D)$ or $dom(Q)$, but not both,
  2. $D(v) = post(D)(v)$ for $v \in dom(D)$,
  3. $Q(v) = post(D)(v)$ for $v \in dom(Q)$, and
  4. $D(u) \leq Q(v)$ for $u \in dom(D)$ and $v \in dom(Q)$.
- Initially, $D = [s \mapsto 0]$, and
  $$Q = \left[v \mapsto \begin{cases} W(s,v), & \text{if } \langle s, v\rangle \in E \\ \infty, & \text{otherwise} \end{cases} \mid v \neq s\right].$$
- Each inference step has the form

$$\frac{\langle D, Q\rangle}{\langle D', Q'\rangle}$$

where $u = arg\ min_u Q(u)$, $D' = D[u \mapsto Q(u)]$, and
$Q' = [v \mapsto Q(v) \sqcap (Q(u) + W(u,v)) \mid v \in dom(Q) - \{u\}]$.

Conclusions

- Computing is abstraction engineering.
- The world is full of useful abstractions.
- Deductive inference is both a foundational medium for understanding computation, and a powerful and practical tool for analyzing them.
- Algorithms operate on these abstractions through inference steps, where

$$\textit{Algorithm} = \textit{Inference} + \textit{Strategy} + \textit{Indexing}$$

- Tools based on inference algorithms [1] are making a significant impact across a range of fields.
- For students, this is a golden age with a confluence of theory, tools, and applications creating bountiful research opportunities.

[1] Abduction (e.g., interpolation) and induction (e.g., synthesis, machine learning) are also important.