When Model Checking Met Deduction

N. Shankar

Computer Science Laboratory
SRI International
Menlo Park, CA

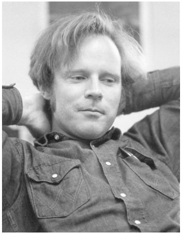
Sep 19, 2014

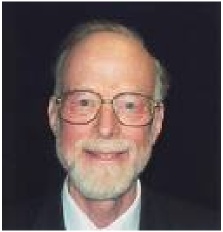
Alan Turing

It is of course important that some efforts be made to verify the correctness of assertions that are made about a routine. There are essentially two types of method available, the theoretical and the experimental. In the extreme form of the theoretical method a watertight mathematical proof is provided for the assertion. In the extreme form of the experimental method, the routine is tried out on the machine with a variety of initial conditions and is pronounced fit if the assertions hold in each case.

Alan Turing (quoted by D. MacKenzie in *Risk and Reason*)

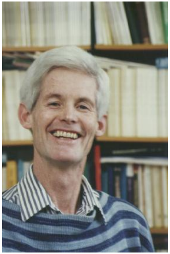
Floyd, Hoare, and Dijkstra

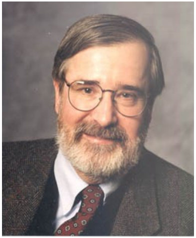




Skip	$\{P\}skip\{P\}$
Assignment	$\{P[\overline{e}/\overline{y}]\}\overline{y} := \overline{e}\{P\}$
Conditional	$\dfrac{\{C \wedge P\}S_1\{Q\} \quad \{\neg C \wedge P\}S_2\{Q\}}{\{P\}C \ ?\ S_1 \ :\ S_2\{Q\}}$
Loop	$\dfrac{\{P \wedge C\}S\{P\}}{\{P\}while\ C\ do\ S\{P \wedge \neg C\}}$
Composition	$\dfrac{\{P\}S_1\{R\} \quad \{R\}S_2\{Q\}}{\{P\}S_1; S_2\{Q\}}$
Consequence	$\dfrac{P \Rightarrow P' \quad \{P'\}S\{Q'\} \quad Q' \Rightarrow Q}{\{P\}S\{Q\}}$

Cook Completeness and Clarke Incompleteness [Apt'81]





7.1 Clarke's Incompleteness Result

A satisfactory treatment of procedures having procedures as parameters is impossible in full generality within the framework of Hoare's logic. This rather astonishing result was proved by Clarke [9] and is the contents of the following theorem.

THEOREM 4. *There exists no Hoare's proof system which is sound and complete in the sense of Cook for a programming language which allows*

1. *procedures as parameters in procedure calls,*
2. *recursion,*
3. *static scope,*
4. *global variables in procedure bodies, and*
5. *local procedure declarations.*

The Role of Fixpoints

- Dijkstra's predicate transformer semantics is key to the completeness argument.
- Verifying $\{P\}S\{Q\}$ reduces to showing
  1. $\{wlp(S)(Q)\}S\{Q\}$, which is always valid, and
  2. $P \implies wlp(S)(Q)$.
- $wlp(while\ C\ do\ S)(Q) = \nu X.(\neg C \wedge Q) \vee (C \wedge wlp(S)(X))$.
- Incompleteness is related to the undecidability of the Halting problem over finite interpretations for the given class of programs.

Fixpoints to Model Checking

- Temporal logics have a fixpoint characterization.

$$\mathbf{EF}p = \mu Y.p \vee EXY$$

$$\mathbf{EG}p = \nu Y.p \wedge EXY$$

- For finite-state systems, the states can be classified in a bounded number of steps.
- With symbolic representations (Binary Decision Diagrams, Difference-Bounded Matrices), the image computations and equivalence checks can be done using logic operations.
- Predicate abstraction allowed logic to sneak in even further through finite-state over-approximations of infinite-state behavior.
- Bounded model checking, k-induction, and interpolation lean more heavily on deduction than model checking.

Bradley's Algorithm

- Bradley's algorithm works by Conflict Directed Reachability (CDR) captured by the abstract system below.[1]
- Given a transition system $M = \langle I, N \rangle$, let $M[X] = I \sqcup N[X]$, where $N[X]$ is the image and $N^{-1}[X]$ is the preimage.
- The state of the algorithm, initially $n = 0$, $Q_0 = I$, $C_0 = \emptyset$, consists of
  1. Inductive candidates (sets of clauses) $Q_0, \ldots, Q_n$:
     1. $Q_0 = I$
     2. $Q_i \sqsubseteq Q_j \sqcap P$ for $i < j \leq n$
     3. $N[Q_i] \sqsubseteq Q_{i+1}$
  2. Counterexample candidates (sets of cubes) $C_0, \ldots, C_n$, where each C_i is a set of symbolic counterexamples: for each (cube) $R \in C_i$
     1. $R = \neg P$ and $i = n$, or there is an S in $C_{i'}$, $R \sqsubseteq N^{-1}[S]$, where $i' = n$ if $i = n$, and $i' = i + 1$, otherwise.
     2. $Q_j \sqsubseteq \neg R$ for all $j < i$.
     3. $R \sqcap Q_i$ is nonempty, i.e., $Q_i \not\sqsubseteq \neg R$.

[1] Thanks to Aaron Bradley, Dejan Jovanović, and Bruno Dutertre for feedback.

Abstract Conflict Directed Reachability

- **Fail:** If C_0 is nonempty, $\neg P$ is reachable.
- **Succeed:** If $Q_i = Q_{i+1}$ for some $i < n$, we have an inductive weakening of P.
- **Extend:** If C_i is empty for each $i \leq n$, add Q_{n+1} such that $M[Q_n] \sqsubseteq Q_{n+1}$ and $C_{n+1} = \emptyset$, if $Q_{n+1} \sqsubseteq P$, and $C_{n+1} = \{\neg P\}$, otherwise.
- **Refine:** Check $N[Q_i] \sqsubseteq \neg R$ for some R in C_{i+1}, where C_i is empty, for $j \leq i$:
    1. **Strengthen:** If the query succeeds, find an $\widehat{R}$ weakening R that is relatively inductive: $M[Q_i \sqcap \neg\widehat{R}] \sqsubseteq \neg\widehat{R}$: conjoin $\neg\widehat{R}$ to each Q_j for $1 \leq j \leq i+1$, move any $S \in C_{i+1}$ such that $Q_{i+1} \sqsubseteq \neg S$ (including R) to C_{i+2} if $i+1 < n$.
    2. **Reverse:** If the query fails with counterexample s, weaken s to S such that $S \sqsubseteq N^{-1}[R]$ and add S to C_i.
- **Propagate:** Whenever Q_i is strengthened, strengthen Q_{i+1} with Q where $M[Q_i] \sqsubseteq Q$, move any $R \in C_{i+1}$ such that $Q_{i+1} \sqsubseteq \neg R$ to C_{i+2} if $i+1 < n$.

Model Checking Becomes Deduction

- Automated verification through model checking of temporal formulas was very successful for essentially finite-state systems.
- Many systems could be reduced to tractable finite-state systems through abstraction, composition, and a little deduction.
- In the Handbook article, *deductive* is interpreted (narrowly) as syntax-directed, and *model checking* (broadly) as based on semantic unfolding.
- "Bounded model checking", k-induction, and interpolation, as practised, are really *deductive* approaches.
- Techniques like Bradley's CDR are squarely deductive, but owe a lot to model checking.
- Ed's contributions have radically advanced verification as a whole, and not merely model checking.

"Happy Retirement, Ed"

We obviously have a long way to go, so it's good that Ed will soon have few other distractions.