Introduction to Kleene Algebra	**Lecture 25**
CS786 Spring 2002	**Thursday, April 25, 2002**

Complexity of PHL

In this lecture we show the *PSPACE*-hardness of propositional Hoare logic (PHL). This result was first proved in [6] by a direct encoding of arbitrary polynomial-space Turing machines. The proof given here is a simpler proof from [2] encoding the universality problem for nondeterministic finite automata, a well-known *PSPACE*-complete problem [3] (see Lecture **??**).

Recall from Lecture **??** that the deduction rules of PHL consist of the usual composition, conditional, while, and weakening rules of Hoare logic, as well as the and- and or-rule

$$\frac{\{c\}\,p\,\{d\},\ c \in C}{\{\bigvee C\}\,p\,\{d\}} \qquad\qquad \frac{\{b\}\,p\,\{c\},\ c \in C}{\{b\}\,p\,\{\bigwedge C\}}$$

for any finite set C of propositions. The and- and or-rule are not part of the traditional formulation [1] but are necessary for completeness [7]. The assignment axiom is meaningless in PHL and is omitted. We are interested in the validity of rules of the form

$$\frac{\{b_1\}\,p_1\,\{c_1\},\ldots,\{b_n\}\,p_n\,\{c_n\}}{\{b\}\,p\,\{c\}} \tag{1}$$

interpreted as universal Horn sentences over relational models.

We consider two related decision problems: given a rule of the form (1),

(i) is it relationally valid? That is, is it true in all relational models?

(ii) is it derivable in PHL?

We show that both of these problems are *PSPACE*-hard by a single reduction from the universality problem for nondeterministic finite automata: given such an automaton M over input alphabet $\{0,1\}$ with states Q, nondeterministic transition function $\Delta : Q\times\{0,1\} \to 2^Q$, start states $S \subseteq Q$, and final states $F \subseteq Q$, does M accept all strings?

Intuitively, our construction simulates the *subset construction* for forming a deterministic automaton from M (see for example [4, 5]). Typically, one thinks of pebbles placed on the states of M and moved according to the transition rules of M. The pebbles keep track of all the states that M could possibly be in after scanning a prefix of the input string. We start at time 0 with pebbles on all the start states of M. Now say we have a set A of states of M

occupied by pebbles at time t. If the next input symbol is a, at time $t+1$ we place a pebble on each state reachable under input symbol a from a state in A.

To encode this in PHL, let a_u be an atomic proposition for each state $u \in Q$. Let b be another atomic proposition and let p be an atomic program. Let

$$\text{START} \stackrel{\text{def}}{=} \bigwedge_{u \in S} a_u \qquad\qquad \text{FINAL} \stackrel{\text{def}}{=} \bigvee_{u \in F} a_u .$$

Consider the rule

$$\frac{\{a_u \wedge b\}\, p\, \{a_v\} \text{ for all } v \in \Delta(u,1), \quad \{a_u \wedge \neg b\}\, p\, \{a_v\} \text{ for all } v \in \Delta(u,0)}{\{\text{START}\}\ \mathbf{while}\ \text{FINAL}\ \mathbf{do}\ p\ \{\mathbf{0}\}} \tag{2}$$

Note that this rule is linear in the size of the description of the automaton.

Intuitively, a_u says that state u is occupied by a pebble, and b (respectively, $\neg b$) says that the next input symbol is 1 (respectively, 0). The program p says to place pebbles on *at least* all states reachable from a currently pebbled state under the next input symbol according to the transition rules of M. The formula START says that all start states are pebbled, and FINAL says that at least one final state is pebbled. Other subexpressions of (2) have the following intuitive meanings:

$\{a_u \wedge b\}\, p\, \{a_v\}$	"If state u is pebbled at time t, and if the next input symbol is 1, then there must be a pebble on state v at time $t+1$."
while FINAL **do** p	"Continue updating the pebble positions as long as there is a final state occupied by a pebble."

The formula (2) says intuitively that if all start states are initially pebbled, and if in each step the pebbles are moved such that *at least* those states that are reachable under the current input symbol from a currently pebbled state are pebbled in the next step, then there is always a pebble on a final state.

Now we proceed to the formal proof of the correctness of this construction.

Theorem 0.1 *The following are equivalent:*

(i) *The rule* (2) *is relationally valid.*

(ii) *The rule* (2) *is derivable in* PHL.

(iii) *The automaton M accepts all strings.*

Proof. We show (ii) $\Rightarrow$ (i) $\Rightarrow$ (iii) $\Rightarrow$ (ii). The first implication is immediate from the soundness of PHL over relational models.

For the second implication, let $x \in \{0,1\}^*$ be any input string. Build a relational model of PHL as follows: the elements are the prefixes of x; the formula b is true at y if $x = yz$ and the first symbol of z is 1; the formula a_u is true at y if $u \in \Delta(S, y)$, that is, if the state u is reachable under input string y from a start state of M; and the program p is the relation consisting of all pairs (y, z) for $|z| = |y| + 1$. An easy argument shows that all premises of (2) hold in this model. By (i), the conclusion holds, thus x satisfies FINAL, so there is a final state reachable from a start state of M under input string x.

Finally, for the third implication, we prove (2) in PHL. Let

$$\mathcal{R} \stackrel{\text{def}}{=} \{\Delta(S,x) \mid x \in \{0,1\}^*\} \qquad\qquad \varphi \stackrel{\text{def}}{=} \bigvee_{A \in \mathcal{R}} \bigwedge_{s \in A} a_s.$$

The set $\mathcal{R}$ is just the set of reachable states of the subset automaton. It follows in a straightforward way from the premises of (2) using the and-, or-, and weakening rules that φ is an invariant of p, or in other words $\{\varphi\}\, p\, \{\varphi\}$. Since START $\rightarrow \varphi$ and $\varphi \rightarrow$ FINAL, the latter being a consequence of (iii), the conclusion of (2) follows from the while rule and weakening. $\square$

References

[1] K. R. Apt. Ten years of Hoare's logic: a survey—part I. *ACM Trans. Programming Languages and Systems*, 3:431–483, 1981.

[2] Ernie Cohen and Dexter Kozen. A note on the complexity of propositional Hoare logic. *Trans. Computational Logic*, 1(1):171–174, July 2000.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[4] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[5] Dexter Kozen. *Automata and Computability.* Springer-Verlag, New York, 1997.

[6] Dexter Kozen. On Hoare logic and Kleene algebra with tests. *Trans. Computational Logic*, 1(1):60–76, July 2000.

[7] Dexter Kozen and Jerzy Tiuryn. On the completeness of propositional Hoare logic. In J. Desharnais, editor, *Proc. 5th Int. Seminar Relational Methods in Computer Science (RelMiCS 2000)*, pages 195–202, January 2000.