Exercise Class January 14, 2021

This is a slightly edited and re-formated transcript of the live notes taken in class.

Exercise 7

Task: Rusinowitch-Turuani with specified maximal number of sessions

Input to the problem:

- initial adversary knowledge
- (I_1, k_1)
- (I_2, k_2)
- (I_3, k_3)
- ...
- (I_n, k_n)

k_n: number of "copies" of the protocol instance that the adversary can use

easy case: k_i are represented in unary unary representation: write number k as string of length k (i.e., 1^k), then:

- binary representation of 7: 111
- unary representation of 7: 1111111

translation algorithm: copy each I_i k_i many times, giving $J_{\dots}$ with renaming of variables.

output of our algorithm = input to RT algorithm:

- (possibly different) initial adversary knowledge
- (J_1)
- (J_2)
- (J_3)
- ...
- (J_m)

Since k_i is represented in unary, this translation works in polynomial time. So we have a polynomial-time many-one reduction from our generalized problem to INSECURE. So, our generalized problem is also in NP.

more complex case: k_i are represented in binary translation still works, but not in polynomial time.

- good news: problem is still decidable
- bad news: we don't have a good upper bound (probably PSPACE-complete).

reductions easy problem: STRING-1: determine whether string ends with "1"

- unary case: GENERALIZED-INSECURE $\leq_m^p$ INSECURE, so GENERALIZED-INSECURE is in NP.
- also: STRING-1 $\leq_m^p$ INSECURE, by:
  - input: string
  - let P_1 be a secure protocol, P_2 an insecure protocol
  - if string ends with 1, return P_2
  - otherwise, return P_1
- also: INSECURE $\leq_m^p$ GENERALIZED-INSECURE (by setting all k_i to 1). So, GENERALIZED-INSECURE is NP-hard. Since it's also in NP, it is NP-complete.

Exercise 8

Task: Needham-Schroeder as Horn clauses

Task: Missing Proof

equations: LHS is more "complex", RHS is "simplification"

algorithm for normal form:

- **INPUT**: term t
- $v := t$
- while these is some v' with $v \to v'$, and $v \neq v'$:
  - $v := v'$
- **OUTPUT**: term v, which is normal form of t

equation: hash(decA($\hat{k}_B$)(encA(kB)(t))) = hash(t)
(we can apply equations inside function calls)

correctness proof

- algorithm terminates: because rewrite relation is terminating.
- algorithm output is well-defined: because of confluence
- output is normal form of input term t:
  - output is some normal form, because algorithm stopped.
  - output is normal form of t (and not some random other term), because $v \equiv_E t$: E-equivalence is maintained in every step.

Task: "Badly-Behaved" Equational Theories

- operators: a, b, c
- equations: a(x) = b(a(x)), a(x) = c(a(x))

bad behavior:

- a(t) -> b(a(t)) -> b(c(a(x))) -> ...
- a(t) -> c(a(t)) -> c(b(a(x))) -> ...

not terminating, because we can add as many b/c's inside as we like:

- a(t) -> b(a(t)) -> b(c(a(x))) -> b(c(c(a(x)))) -> b(c(c(b(a(x)))))
- a(t) -> c(a(t)) -> c(b(a(x))) -> c(b(a(c(x)))) -> c(b(a(c(b(x)))))

this is also **not** confluent, since we can only do "unification" inside, but the outmose operators will remain different.

alternative (not worse)

- operators: a, b, c
- equations: a(x) = b(a(x)), a(x) = c(a(x)), a(x)=d, a(x)=e