Automated Theorem Proving

Georg Moser

Institute of Computer Science @ UIBK

Winter 2015

Summary of Last Lecture

Definition

$$\frac{C \vee A \quad D \vee \neg B}{(C \vee D)\sigma}\ \mathsf{ORe} \qquad \frac{C \vee A \vee B}{(C \vee A)\sigma}\ \mathsf{OFc}$$

$$\frac{C \vee s = t \quad D \vee \neg A[s']}{(C \vee D \vee \neg A[t])\sigma}\ \mathsf{OPm(L)} \qquad \frac{C \vee s = t \quad D \vee A[s']}{(C \vee D \vee A[t])\sigma}\ \mathsf{OPm(R)}$$

$$\frac{C \vee s = t \quad D \vee u[s'] \neq v}{(C \vee D \vee u[t] \neq v)\sigma}\ \mathsf{SpL} \qquad \frac{C \vee s = t \quad D \vee u[s'] = v}{(C \vee D \vee u[t] = v)\sigma}\ \mathsf{SpR}$$

$$\frac{C \vee s \neq t}{C\sigma}\ \mathsf{ERR} \qquad \frac{C \vee u = v \vee s = t}{(C \vee v \neq t \vee u = t)\sigma}\ \mathsf{EFc}$$

- ORe and OFc are ordered resolution and ordered factoring
- OPm(L), OPm(R), SpL, SpR stands for ordered paramodulation and superpostion (left or right)
- ERR means equality resolution and EFc means equality factoring

Outline of the Lecture

Early Approaches in Automated Reasoning

Herbrand's theorem for dummies, Gilmore's prover, method of Davis and Putnam

Starting Points

resolution, tableau provers, Skolemisation, redundancy and deletion

Automated Reasoning with Equality

ordered resolution, paramodulation, ordered completion and proof orders, superposition

Applications of Automated Reasoning

Neuman-Stubblebinde Key Exchange Protocol, Robbins problem

Neuman-Stubblebine Key Exchange Protocol

Description

- Neuman-Stubblebine key exchange protocol aims to establish a secure key between two agents that already share secure keys with a trusted third party
- principals: Alice, Bob, Server

Conventions

A, B, T: identifiers of Alice, Bob, Server

$\mathsf{N_a}$, $\mathsf{N_b}$: nonce created by Alice, Bob

Time: time span of key $\mathsf{K_{ab}}$

$\mathsf{E}_{key}(message)$: encryption of *message* using *key*

$\mathsf{K_{at}}$: key between A and T

$\mathsf{K_{bt}}$: key between B and T

$\mathsf{K_{ab}}$: key between A and B

Definition

we write

$$\mathsf{A} \longrightarrow \mathsf{B}\colon M \qquad \text{Alice sends Bob message } M$$

The Protocol

1. $A \longrightarrow B: A, N_a$
   Alice sends to Bob
   - her identifier
   - a freshly generated nonce
2. $B \longrightarrow T: B, E_{K_{bt}}(A, N_a, Time), N_b$
   Bob encrypts the triple $(A, N_a, Time)$ and sends to Server
   - his identity
   - encryption of $(A, N_a, Time)$
   - new nonce
3. $T \longrightarrow A: E_{K_{at}}(B, N_a, K_{ab}, Time), E_{K_{bt}}(A, K_{ab}, Time), N_b$
   Server generates K_{ab} and sends to Alice
   - encryption of K_{ab} with key for Alice
   - encryption of K_{ab} with key for Bob
   - N_b
4. $A \longrightarrow B: E_{K_{bt}}(A, K_{ab}, Time), E_{K_{ab}}(N_b)$
   Alice encrypts Bob's nonce with K_{ab} and forwards part of message

The Attack

Assumptions

1. intruder can intercept and record all sent messages
2. intruder can send messages and can forge the sender of a message
3. intruder can encrypt messages, when he finds out a key
4. intruder has no access to information private to Alice, Bob, or Server the server.
5. intruder cannot break any secure key

still Intruder (denoted I) can break the protocol

1. $I(A) \longrightarrow B: A, N_a$
2. $B \longrightarrow I(T): B, E_{K_{bt}}(A, N_a, Time), N_b$.
3. $I(A) \longrightarrow B: E_{K_{bt}}(A, N_a, Time), E_{N_a}(N_b)$.

the problem is that keys and nonces can be confused

$$E_{K_{bt}}(A, K_{ab}, Time) \quad \text{and} \quad E_{K_{bt}}(A, N_a, Time)$$

Formalisation in First-Order

Definition

definition of the language $\mathcal{L}$ of the formalisation

1. individual constants: a, b, t, na, at, bt
   - a, b, t are to be interpreted as the identifiers A, B, and T
   - constant na refers to Alics's nonce
   - at (bt) represents the key K_{at} (K_{bt})
2. function constants: nb, tb, kt, key, sent, pair, triple, encr, quadr
   - nb, tb, kt are unary; key, pair, encr are binary; sent, triple are ternary, and quadr is 4-ary
   - nb, tb compute Bob's fresh nonce and the time-stamp Time
   - kt computes of the new key
   - the other constants act as containers as the formalisation is based on unary predictes

Definition (Definition (cont'd))

4. predicate constants: Ak, Bk, Tk, P, M, Fresh, Nonce, $Store_a$, $Store_b$
   - Ak, Bk, Tk assert together with key existence of keys
   - P represents principals
   - M represents messages using the function sent
   - Fresh asserts that Bob is only interested in fresh nonces
   - Nonce denotes that its argument is a nonce
   - $Store_a$, $Store_b$ denote information that is in the store of Alice or Bob

Notation

we indicate the type of a bound variable in its name as subscript
the bound variable x_{na} indicates that this variable plays the role of the nonce N_a

Formalisation of Protocol

$A \longrightarrow B\colon A, N_a$

$1\colon \mathsf{Ak}(\mathsf{key}(\mathsf{at},\mathsf{t}))$

$2\colon \mathsf{P}(\mathsf{a})$

$3\colon \mathsf{M}(\mathsf{sent}(\mathsf{a},\mathsf{b},\mathsf{pair}(\mathsf{a},\mathsf{na}))) \wedge \mathsf{Store_a}(\mathsf{pair}(\mathsf{b},\mathsf{na}))$

$B \longrightarrow T\colon B, E_{K_{bt}}(A, N_a, Time), N_b$

$4\colon \mathsf{Bk}(\mathsf{key}(\mathsf{bt},\mathsf{t}))$

$5\colon \mathsf{P}(\mathsf{b})$

$6\colon \mathsf{Fresh}(\mathsf{na})$

$7\colon \forall x_a\ x_{na}\,(\mathsf{M}(\mathsf{sent}(x_a,\mathsf{b},\mathsf{pair}(x_a,x_{na}))) \wedge \mathsf{Fresh}(x_{na}) \rightarrow$
$\rightarrow \mathsf{Store_b}(\mathsf{pair}(x_a,x_{na})) \wedge \mathsf{M}(\mathsf{sent}(\mathsf{b},\mathsf{t},$
$\mathsf{triple}(\mathsf{b},\mathsf{nb}(x_{na}),\mathsf{encr}(\mathsf{triple}(x_a,x_{na},\mathsf{tb}(x_{na})),\mathsf{bt})))))$

$T \longrightarrow A\colon E_{K_{at}}(B, N_a, K_{ab}, Time), E_{K_{bt}}(A, K_{ab}, Time), N_b$

$8\colon \mathsf{Tk}(\mathsf{key}(\mathsf{at},\mathsf{a})) \wedge \mathsf{Tk}(\mathsf{key}(\mathsf{bt},\mathsf{b}))$

$9\colon \mathsf{P}(\mathsf{t})$

$10\colon \forall x_b \forall x_{nb} \forall x_a \forall x_{na} \forall x_{time} \forall x_{bt} \forall x_{at}$
$(\mathsf{M}(\mathsf{sent}(x_b,\mathsf{t},\mathsf{triple}(x_b,x_{nb},\mathsf{encr}(\mathsf{triple}(x_a,x_{na},x_{time}),x_{bt})))) \wedge$
$\wedge \mathsf{Tk}(\mathsf{key}(x_{at},x_a)) \wedge \mathsf{Tk}(\mathsf{key}(x_{bt},x_b)) \wedge \mathsf{Nonce}(x_{na}) \rightarrow \mathsf{M}(\mathsf{sent}(\mathsf{t},x_a,$
$\mathsf{triple}(\mathsf{encr}(\mathsf{quadr}(x_b,x_{na},\mathsf{kt}(x_{na}),x_{time}),x_{at}),$
$\mathsf{encr}(\mathsf{triple}(x_a,\mathsf{kt}(x_{na}),x_{time}),x_{bt}),x_{nb}))))$

$11\colon \mathsf{Nonce}(\mathsf{na})$

$12\colon \forall x \neg \mathsf{Nonce}(\mathsf{kt}(x))$

$13\colon \forall x\,(\mathsf{Nonce}(\mathsf{tb}(x)) \wedge \mathsf{Nonce}(\mathsf{nb}(x)))$

Remark

formulas 11–13 are not part of the protocol, but prevents that the intruder can generate arbitrarily many keys

$A \longrightarrow B\colon E_{K_{bt}}(A, K_{ab}, Time), E_{K_{ab}}(N_b)$

$14\colon \forall x_{nb} \forall x_k \forall x_m \forall x_b \forall x_{na} \forall x_{time}$
$((\mathsf{M}(\mathsf{sent}(\mathsf{t},\mathsf{a},\mathsf{triple}(\mathsf{encr}(\mathsf{quadr}(x_b,x_{na},x_k,x_{time}),\mathsf{at}),x_m,x_{nb}))) \wedge$
$\wedge \mathsf{Store_a}(\mathsf{pair}(x_b,x_{na}))) \rightarrow$
$\rightarrow \mathsf{M}(\mathsf{sent}(\mathsf{a},x_b,\mathsf{pair}(x_m,\mathsf{encr}(x_{nb},x_k)))) \wedge \mathsf{Ak}(\mathsf{key}(x_k,x_b)))$

$15\colon \forall x_k \forall x_a \forall x_{na}$
$((\mathsf{M}(\mathsf{sent}(x_a,\mathsf{b},\mathsf{pair}(\mathsf{encr}(\mathsf{triple}(x_a,x_k,\mathsf{tb}(x_{na})),\mathsf{bt}),$
$\mathsf{encr}(\mathsf{nb}(x_{na}),x_k)))) \wedge$
$\wedge \mathsf{Store_b}(\mathsf{pair}(x_a,x_{na}))) \rightarrow \mathsf{Bk}(\mathsf{key}(x_k,x_a)))$

Fact

SPASS verifies that the protocol terminates in less than a millisecond

$$\mathcal{G} \models \exists x(\mathsf{Ak}(\mathsf{key}(x,\mathsf{a})) \wedge \mathsf{Bk}(\mathsf{key}(x,\mathsf{b})))$$

Formalisation of the Intruder

extend $\mathcal{L}$ by predicate constants Ik and Im

Behaviour of Intruder

$16\colon \forall x_a\ x_b\ x_m\,(\mathsf{M}(\mathsf{sent}(x_a,x_b,x_m)) \rightarrow \mathsf{Im}(x_m))$

$17\colon \forall u\ v\,(\mathsf{Im}(\mathsf{pair}(u,v)) \rightarrow \mathsf{Im}(u) \wedge \mathsf{Im}(v))$

⋮

$20\colon \forall u\ v\,(\mathsf{Im}(u) \wedge \mathsf{Im}(v) \rightarrow \mathsf{Im}(\mathsf{pair}(u,v)))$

⋮

$23\colon \forall x\ y\ u\,((\mathsf{P}(x) \wedge \mathsf{P}(y) \wedge \mathsf{Im}(u)) \rightarrow \mathsf{M}(\mathsf{sent}(x,y,u)))$

$24\colon \forall u\ v\,((\mathsf{Im}(u) \wedge \mathsf{P}(v)) \rightarrow \mathsf{Ik}(\mathsf{key}(u,v)))$

$25\colon \forall u\ v\ w\,((\mathsf{Im}(u) \wedge \mathsf{Ik}(\mathsf{key}(v,w) \wedge \mathsf{P}(w)) \rightarrow \mathsf{Im}(\mathsf{encr}(u,v)))$

Fact

$\mathcal{H}$ extends $\mathcal{G}$ by 16–25

SPASS shows that the protocol insecure in less than a millisecond

$$\mathcal{H} \models \exists x(\mathsf{Ik}(\mathsf{key}(x,\mathsf{b})) \wedge \mathsf{Bk}(\mathsf{key}(x,\mathsf{a})))$$

Definition

$\mathcal{B} = \langle B; +, \cdot, \sim, 0, 1\rangle$ is a Boolean algebra if

1. $\langle B; +, 0\rangle$ and $\langle B; \cdot, 1\rangle$ are commutative monoids
2. $\forall\ a, b, c \in B$:
$$a \cdot (b + c) = (a \cdot b) + (a \cdot c) \qquad a + (b \cdot c) = (a + b) \cdot (a + c)$$
3. $\forall\ a \in B$: $a + \sim a = 1$ and $a \cdot \sim a = 0$

$\sim a$ is called complement (or negation) of a

Definition

consider the following axioms:

$$\begin{aligned} x + y &= y + x && \text{commutativity} \\ (x + y) + z &= x + (y + z) && \text{associativity} \\ \mathsf{n}(\mathsf{n}(x) + y) + \mathsf{n}(\mathsf{n}(x) + \mathsf{n}(y)) &= x && \text{Huntington equation} \end{aligned}$$

the operation $\mathsf{n}(\cdot)$ is just complement

Theorem

the provided axioms form a minimal axiomatisation of Boolean algebras, that is all axioms are independent from each other

Example

recall $x \cdot y = \sim(\sim x + \sim y)$, thus

$$\sim(\sim x + y) + \sim(\sim x + \sim y) = x \cdot \sim y + x \cdot y = x \cdot (\sim y + y) = x$$

Definition

Robbins equation:

$$\sim(\sim(x + y) + \sim(x + \sim y)) = x \qquad \text{(R)}$$

Example

$$\sim(\sim(x + y) + \sim(x + \sim y)) = (x + y) \cdot (x + \sim y) = x + (y \sim y) = x$$

Robbins Question

Question ①

Does Huntington's equation follow from (i) commutativity (ii) associativity and (iii) Robbins equation?

Answer

McCune (or better EQP) says yes

Definition

a Robbins algebra is an algrebra satisfying (i) commutativity (ii) associativity and (iii) Robbins equation

Question ②

Is any Robbins algebra a Boolean algebra?

Auxiliary Lemmas

Lemma

a Robbins algebra satisfying $\exists x(x + x = x)$ is a Boolean algebra

Proof (Sketch).

automatically provable by EQP in about 5 seconds ∎

Lemma

a Robbins algebra satisfying $\exists x \exists y(x + y = x)$ is a Boolean algebra

Proof (Sketch).

1. originally the lemma was manually proven by Steve Winker
2. based on the above lemma, EQP can find a proof in about 40 minutes ∎

Lemma

a Robbins algebra satisfying $\exists x \exists y(\sim(x+y) = \sim x)$ *is a Boolean algebra*

Proof (Sketch).

originally the lemma was manually proven by Steve Winker ■

Lemma

all Robbin algebras satisfy $\exists x \exists y(x + y = x)$

Proof (Sketch).

by EQP, dedicated (incomplete) heuristics are essential ■

Theorem

commutativity, associativity, and Robinns equation minimally axiomatise Boolean algebra

Proof (of First and Last Lemma).

$n(n(n(x) + y) + n(x + y)) = y$ 7, (R)

$n(n(n(x + y) + n(x) + y) + y) = n(x + y)$ 10, [7 → 7]

$n(n(n(n(x) + y) + x + y) + y) = n(n(x) + y)$ 11, [7 → 7]

$n(n(n(n(x) + y) + x + 2y) + n(n(x) + y)) = y$ 29, [11 → 7]

$n(n(n(n(n(x) + y) + x + 2y) + n(n(x) + y) + z) + {}$
$+ n(y + z)) = z$ 54, [29 → 7]

$n(n(n(n(n(x) + y) + x + 2y) + n(n(x) + y) + {}$
$+ n(y + z) + z) + z) = n(y + z)$ 217, [54 → 7]

$n(n(n(n(n(n(x) + y) + x + 2y) + n(n(x) + y) + {}$
$+ n(y + z) + z) + z + u) + n(n(y + z) + u)) = u$ 674, [217 → 7]

$n(n(n(n(3x) + x) + n(3x)) + n(n(n(3x) + x) + 5x)) = {}$
$= n(n(3x) + x)$ 6736, [10 → 674]

Proof.

$n(n(n(3x) + x) + 5x) = n(3x)$ 8855, [6736 → 7]

$n(n(n(n(3x) + x) + n(3x) + 2x)) = n(n(3x) + x) + 2x$ 8865, [8855 → 7]

$n(n(n(3x) + x) + n(3x)) = x$ 8866, [8855 → 7]

$n(n(n(n(3x) + x) + n(3x) + y) + n(x + y)) = y$ 8870, [8866 → 7]

$n(n(3x) + x) + 2x = 2x$ 8871, [8865]

- last line asserts: $\exists x \exists y(x + y = x)$
- also derived: $\exists x \exists y(\sim(x + y) = \sim x)$

■

Remarks

- SPASS could not find proof in 12 hours
- mkbtt cannot parse the problem ☺

Equational Prover EQP

Definition

- EQP is restricted to equational logic and performs AC unification and matching
- based on basic superposition, that is, paramodulation into substitution parts of terms are forbidded
- incomplete heuristics

Definition

- AC unifiers are found by finding a basis of a linear Diophantine equation
- the complete set of unifiers is given as linear combinations of (members of) the basis

Definition

- a subset yields potential unifier if all unification conditions except unification of subterms are fulfilled
- the super-0 strategy restricts the number of AC unifiers by ignoring supersets if a potential unifier is found

NB: the super-0 strategy yields incompleteness

Definition

for AC matching a dedicated algorithm based on backtracking is used

Definitions

- the weight of a pair of equations be the sum of the size of its members
- the age of a pair is the sum of the ages of its members

Definition

a pairing algorithm used to select the next equation:

1. either the lightest or the oldest pair (not yet selected) is chosen
2. pair selection ratio specifies the ratio $\frac{\mathit{lightest}}{\mathit{oldest}}$
3. default ratio is $\frac{1}{0}$

Use of EQP

- successful attack took place over the course of five weeks
- the following search parameters were varied
  1. limit on the size of retained equations
  2. with or without super-0 heuristics
  3. with or without basic restriction
  4. pair selection ratio $\frac{1}{0}$ or $\frac{1}{1}$
- subsequent experiments searched for shorter proofs
- yielded direct proof without the use of Winker's lemmas

Thank You for Your Attention!