System Description: cardT^AP: The First Theorem Prover on a Smart Card

Rajeev Goré[1], Joachim Posegga[2], Andrew Slater[1] , Harald Vogt[2]

[1] Automated Reasoning Project, Australian National University, 0200, Canberra
[2] Deutsche Telekom AG, Technologiezentrum, IT Security, D-64276 Darmstadt

Abstract. We present the first implementation of a theorem prover running on a smart card. The prover is written in Java and implements a dual tableau calculus. Due to the limited resources available on current smart cards, the prover is restricted to propositional classical logic. It can be easily extended to full first-order logic.
The potential applications for our prover lie within the context of security related functions based on trusted devices such as smart cards.

1 Smart Cards: the Secure PC of Tomorrow

Smart cards are currently evolving into one of the most exciting and most significant technologies of the information society. Current smart cards on the market are in fact small computers consisting of a processor, ROM and RAM, an operating system, a file system, etc. Although their resources are still quite restricted, continuous advances in chip manufacturing will soon enable to market smart cards with 32 bit processors and up to 128 KByte of memory. Manufacturers are also thinking about integrating small keyboards and LCD displays on these plastic cards. Thus, the next generation of smart cards will be as powerful as PCs were a few years ago.

The evolution of smart card technology resembles the development of computer technology over the last 20 years: the separation of "physics" and "logic". While early computers had to be programmed in machine language because each bit of memory and each instruction cycle was valuable, the increase of resources and processing power made it affordable to trade resources for higher level programming concepts and languages. This separation of software and hardware was the basis for the spread of computers into everyday life during this decade.

The same phenomenon is about to take place in smart card technology: as resources and processing power increase, it will become affordable to neglect the optimal use of the card processor and memory. The most promising move in this direction are Java smart cards, where a Java virtual machine is implemented inside the card. The software determining the function of the card is no longer tied to the particular card, but multiple applications can be loaded onto, and removed from, the card as desired.

The primary purpose of smart cards will continue to be security-related applications since they will serve as a trusted device for their owner. The most important applications to date are of a cryptographic nature like authentication and encryption, eg for electronic cash. Today's smart cards are used essentially as simple authentication devices in these contexts. Future smart cards applications, however, will take advantage of the fact that more complex cards will be able to carry out more complex operations; so that the smart card of the future will be a secure, personal computer.

Current smart cards have security-related applications hard-wired onto them. Future smart cards will serve multiple purposes and will be adaptable by downloading one or more applications. Interactions between such applications, and between the card and the outside world therefore become non-trivial. Formal logic is not only well-suited for modelling such complex interactions but is also ideal for describing a given security model. Consequently, a trusted, secure, personal device should be able to perform logical reasoning to ensure that the card complies to its owner's security model. A concrete example is the use of formal logic in the context of proof-carrying code [1].

Here we outline the first successful implementation of a theorem prover on a Java smart card.

2 Implementation Details

cardT^AP is a theorem prover for propositional logic that uses a dual tableaux method based on leanT^AP [2]. cardT^AP was specifically designed to reside on a smart card; the program executable size is less that 2 KByte, and the stack usage, heap space and allocated memory is minimal. To achieve this, cardT^AP naively simulates Prolog's run time stack and backtracking environment using recomputation. The trade off is efficiency: some work must be repeated since we cannot save all of the prover's previous states.[1] The theorem prover resides on the smart card as a Java applet which can down-load a formula from a card reader and determine its theoremhood.

Due to space constraints we only allow formulæ in Negated Normal Form using Reverse Polish Notation. The current prover is limited by statically defined restrictions on the length and complexity of the formula determined by the limited memory resources of the smart card [3]. Specifically, a formula can contain up to 26 distinct propositional variables, at most 20 disjunctions, and at most 20 nested conjunctions, with a total length of 126 symbols. Future cards with greater resources will be less restrictive.

Formulæ are written to an EEPROM file; excess EEPROM space is used as virtual memory. The efficiency of accessing the formula is somewhat enhanced by using a smaller buffer in local memory as a small "window" into the formula.

The prover traverses each path from the root of the proof tree to some leaf. If every leaf is closed then the formula is unsatisfiable. Typically a dual tableaux theorem prover is capable of remembering or copying information about some

[1] Although backtracking is not necessary for propositional classical logic, cardT^AP has been implemented for extensions to other logics.

point in the proof tree before it takes a branch at some conjunction. By doing so it can efficiently return to that branching point and traverse the alternate path. cardT^AP does not have enough memory space to arbitrarily store a "state" for some branch. As an alternative, cardT^AP stores a simple binary map of the paths taken to implement a depth first traversal of the proof tree, but is thus required to return to the root to traverse its next path. In doing so it reaccumulates the state information it had previously acquired at the last branching point in the proof tree. Disjunctions are also mapped so that if the prover reaches a leaf node leading from some conjunction, it may look in the disjunction map to determine whether there is a disjunct to process that may result in the current path closing. If a disjunct is available then that subformula may be immediately processed as if it were attached to the open node. Each path, from root to leaf, generates its own disjunction map as the path is traversed. Each path also generates state information regarding the variables in the formula as they are identified.

3 Experimental Results

We successfully ran cardT^AP on a smart card provided by Schlumberger [3] implementing JavaCard API V1.0 [4]. This card handles applications of up to 2.8K and offers approximately 200 bytes of main memory during run time. Our test formulæ consisted of 17 theorems of propositional logic [5] converted into negated normal form and into reverse polish notation. We also tested some non-theorems, obtained by mutating some of these 17 theorems.

Each formula was loaded onto the card individually and tested using the proof procedure described above. The interaction was performed through *LoadSolo*, a simple tool for communicating with the card, which came with the Cyberflex Development Kit [3]. cardT^AP returned an answer code indicating whether or not the formula was a theorem. All measurements were made by hand: each theorem was proved 3 times, the fastest and the slowest times were discarded. The following run-times include communication overhead:

Pelletier's 17 theorems:		
P1 21.9 s	P2 6.9 s	P3 2.0 s
P4 22.1 s	P5 8.7 s	P6 1.7 s
P7 1.7 s	P8 3.2 s	P9 27.6 s
P10 1:33 min	P11 7.0 s	P12 -
P13 1:50 min	P14 2:40 min	P15 22.1 s
P16 2.0 s	P17 -	

Non-theorems:		
N1	&-pp	2.0 s
N2	+pp	2.4 s
N3	p	1.8 s
N4	-p	1.8 s
N5	+&&p-q&-qp&+-pq+qp	7.2 s
N6	+&+p-q&-qp&+-pq+qp	25.3 s
N7	+&pp&p-p	2.7 s
N8	++-pq+q-p	5.0 s
N9	+p&+-pqp	6.7 s
N10	+&pq+&-p-q+&-p-q&p-q	7.3 s

Timing constraints enforced by the card due to the corresponding ISO-Standards either raised an exception or garbled communication during some of the longer computations.[2] These problems could be partially solved by interspersing commands which send data from the card to the reader. These modifications are sufficient for proving the shorter theorems, theorems P10 and P13, but for the larger ones, like theorem P14, additional modifications had to be made. All modifications concern only additional commands for communication. Theorems P12 and P17 could not be proved with any version of cardT^AP.

3.1 Conclusion and Outlook

The current version of cardT^AP is a propositional logic theorem prover written in Java. The methodology is essentially the same as that of the Prolog first order logic theorem prover leanT^AP. With greater available resources on smart cards, an extension of cardT^AP to first-order logic is straightforward. The simplicity of the Java code is a direct result of the tableau methodology which nicely partitions the problem into multiple branches, each of which can be explored using the limited resources individually. In contrast, "global" procedures such as resolution would not have been as well suited since they accumulate information rather than partitioning it.

We are currently experimenting with prototypes of new smart cards, implementing Java Card V2.0. These cards offer a 32-Bit RISC processor and up to 16K space in the EEPROM. First tests indicate that our prover runs about one order of magnitude faster in these cards, and it might in fact be realistic to extend the prover to first order logic.

References

1. G Necula and P Lee. Proof carrying code. Technical Report CMU-CS-96-165, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, September 1996.
2. Bernhard Beckert and Joachim Posegga. leanT^AP: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.
3. Schlumberger Inc. Cyberflex. http://www.cyberflex.austin.et.slb.com, 1997.
4. JavaSoft Inc. Javacard API. http://www.javasoft.com/products/javacard/, 1997.
5. Francis J. Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2:191–216, 1986.

[2] The cardT^AP program itself has been independently run in a simulation environment where these problems did not surface.