Some uses of Caml in industry

Xavier Leroy

INRIA Paris-Rocquencourt

CUFP 2007



X. Leroy (INRIA)

Some uses of Caml in industry

CUFP 2007 1 / 34

∃ >



2 Perceived needs; the Caml consortium experiment

3 A quick look at the smart card industry



Static Driver Verifier (Microsoft)



Static verification of Windows kernel-mode drivers, detecting violations of the Windows Driver Model API and usage rules.

Sophisticated static analysis with model checking technology.

Distributed to developers as part of the Windows Driver Kit.

(日) (同) (三) (三)

We developed SLAM using INRIA's OCaml functional programming language. The expressiveness of this language and robustness of its implementation provided a great productivity boost.

MSR-TR-2004-08, T.Ball, B.Cook, V.Levin and S.K.Rajamani

IBM Migration Toolkit

IBM DB2 Migration ToolKit - printshop						
Application Project Tools Help						
ág - 90						
Specify Source Convert Refine Generate Data Transfer Scripts Deploy to DB2						
DB2 B- DB2Catalog						
B- 🖬 GAR	Property	Oracle	DB2			
- C Datatypes	Name	CLEAN SP FROM SPJOBREL	clean sp from spi1			
🖻 🦳 Tables						
B ACCOUNTS						
0 III HOLIDAYS						
B B JOBS	Search / Next		Go to Line Number			
U JOBSTATUS	CREATE TRIDO	ER clean on from soil AFTER DELET	E ON SALESPEOPLE			
U OLD_JOBS	REFERENCING	OLD AS OLD FOR EACH ROW MODE D	828QL			
S ALESPEUPLE	REGIN ATOMIC					
E Coorners	DEGINATOMIC					
E D Foreign Key:	delete from s	p_per_job where sp_id = :old.id;				
B 🦳 Trippers	DELETE FROM	ISP_PER_JOB				
tlean_s	WHERE SP_ID = OLD JD;					
B- B SP_PER_JOB	B B SP_PER_JOB ENDI					
- 🗁 Views						
Procedures/Functio						
Oracle DB2						
Report Messages Number of lines: 11						
For Help, press F1						
CIL2R			* * 8			

Static analysis and conversion of database schema, e.g. between DB2 and Oracle.

Compiler-like technology, generating migration scripts.

Distributed as part of IBM's Migration Toolkit.



A domain-specific language, inspired by the synchronous language Esterel, to program assembly-line automata and robots.

Developed by the Athys start-up, then integrated in the Delmia computer-aided manufacturing environment of Dassault Systèmes. FL/reFLect: an ML-like functional language with BDDs and integrated model checking capabilities.

Used at Intel for high-level modeling and verification of circuits.

Part of the Forte environment for hardware verification.

Reimplementation in progress as a front-end for the OCaml compiler, reusing the back-end.

Modeling Language for Finance (LexiFi)

System System Life System System Life Mark Inde Life Inde State Scenarios Mark Inde Life Inde State Scenarios Mark Inde Life Inde Scenarios Local Scenario Media Life Inde Scenarios Media Life Indo Scenarios Med	🚟 LexiFi Express	\mathbf{X}			
New Tede (Bote), Teden (Match Dala) Someon) Addreek Version Contract Definition (Price Fauer Error & Exercise Fauer Cath Flow Instancer Paumiers (Anaples) Heb Paddree Version (Price Fauer Error & Exercise Fauer Cath Flow Instancer Paumiers (Anaples) Heb Paddree Version (Price Fauer Error & Exercise Fauer Cath Flow Instancer Paumiers (Anaples) Heb Paddree Version (Price Fauer Error & Exercise Fauer Cath Flow Instancer Paumiers (Anaples) Heb Paddree Version (Price Fauer Error & Exercise Fauer Cath Flow Instancer Paumiers (Anaples) Heb Paddree Version (Price Fauer Error & Exercise Fauer Cath Heb Paddree Version (Price Fauer Error & Exercise (Error	System Admin Options Help				
Automative Vision Fause Farge & Escoles Fause Cath Row Instancer Pauseter Angles: Heb Add Hew Angles Static Angles Und Hew Angles Static Angles Vision	New Trade Blotter Trades Market Data Scenarios				
Local Scenario P Import Local Scenario V International Status V Objectness V O	Autocatable Wont of Contract Definition Pricing Future Fierings & Exercises Future Cesh Flows Instrument Parameters Analytics Heb Add New Analytics Static Analysis	_			
Import Local Scenario Underiva Formatio Scenario Scenario Scenario Scenario Underiva Stati Date Endorma Scenario Underiva Stati Date Endorma Scenario Underiva Stati Date Endorma Scenario Underiva Stati Date Calcular Calcular Endorma Scenario Tecneral Analysis Calcular Endorma Scenario Tecneral Analysis Calcular Endorma Scenario Calcular Calcular Endorma Scenario Scenario Scenario Scenario Scenario Endorma Scenario Scenario Scenario Endorma Scenario Scenario Scenario Scenario Scenario Scenario Scenario Scenario Scenario Scenario	Local Scenario				
Underward Stat Date End Date Lower Board Uger Board Wate, 22 19801 01 2100 01 2000 1000 Wate, 22 19801 02 2100 01 2000 1000 Calcular Calcular Calcular Calcular Internal Analysis Filter all Analysis Calcular (Line) Calcular (Line) Internal Analysis Filter all Analysis Calcular (Line) Calcular (Line) Internal Analysis Filter all Analysis Calcular (Line) Calcular (Line) Internal Analysis Filter all Analysis Calcular (Line) Calcular (Line) <	Import Local Scenario				
V Unit entropy was 1960-01 2000-01 2000 4000 NAA., 228 1960-10 2000 1000 1000 Colorida Colorida 2000 1000 1000 1000 Feed International strapp Colorida 10000 10000 10000 10000 100000 100000 100000 100000 100000 100000 100000 1000000 1000000 1000000 1000000 1000000 1000000 1000000 10000000 10000000 1000000000 10000000 1000000000000000000000000000000000000	Indebino Stat Date End Date Lower Round Linner Round				
Cdoube Cdoube 0000 InternalAndynit Cdoube 0000 Presultadynit Franciski Step: Aggregated Cashflow 00000 Presultadynit Franciski Step: Aggregated Cashflow 00000 Presultadynamic Franciski Step: Aggregated Cashflow 00000 Presultadynamic Franciski Step: Aggregated Cashflow 00000 Presultadynamic Franciski Step: Aggregated Cashflow 000000 Presultadynamic Franciski Step: Aggre	DJ Eurostows 50 1980-01-01 2100-01-01 3000 4000	- 1			
Cockete Cockete Internal Analytic Internal Analy	Nikkei_225 1980-01-01 2100-01-01 12000 18000				
Catalate Education E					
Internal-Andrein Recall (Internal-bit Strg: Appropriate Cartificien) Org. centre: a-root()	Calculate Calculate				
Prevail [International Step] Aggregated California one_contentinvorg(endpointinvorg) endpoint_invorg(endpoint_invorg(endpoint_invorg) endpoint_invorg(endpoint_invorg) endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint_invorg(endpoint	Interval Analysis				
Immediate Control (March 10, March 10,	Result Intermediate Step Aggregated Cashflows				

A domain-specific language for formal specification and pricing of complex financial products.

A mild extension of OCaml + a library of financial products + GUIs and interfaces with Excel, etc. A Wall Street trading firm, which develops in-house a lot of software for financial quantitative research, mostly in Caml.

Aggressive hiring of Caml programming talents.

Organizers and sponsors of the "OCaml summer of code" initiative.



A set of systems administration tools for virtualization solutions based on the Xen open-source hypervisor.

Image: Image:

The Astrée static analyzer (ENS)





A static program analyzer for critical embedded software, based on abstract interpretation.

Proves the absence of a large class of run-time errors, including memory violations and integer and floating-point overflows.

Used by Airbus to verify the fly-by-wire software for the A340 and A380. No false alarms! The majority of Caml industrial applications revolve around programming language technologies:

- Domain-specific languages.
- Static analysis, program verification.
- Compilation, interpretation.

Occasional in-roads in systems programming, where scripting languages are typically used: XenSource, early Linspire (\rightarrow Haskell).

More unconventional uses of Caml are to be found in academic projects, especially in network protocols (Ensemble groupware, Unison synchronization, MLDonkey and Peerple P2P applications, ...).



2 Perceived needs; the Caml consortium experiment

3 A quick look at the smart card industry

4 Conclusions

.∃ >

Crucially important:

- Windows support.
- Linux/BSD support.
- x86 64-bit support.
- Stability, stability, more stability.
- Foreign-function interface.
- For some uses (e.g. static analysis): execution speed, moderate memory requirements.

Unimportant:

- GUI toolkits.
- Rich (as in Perl-rich) libraries in general.
- Yet another integrated development environment.

The age-old question: "what if (Xavier \mid Simon $\mid \dots)$ gets run over by a bus?"

Frequently asked question in the 1990's, less so nowadays.

Being open source software helps.

Having been around longer than Java helps.

Could some official commitment from a reputable institution help?

An attempt to (lightly) formalize our relations with industrial users.

Initial goals:

- A place for serious industrial users to meet and discuss.
- Collect funds to pay for a full-time programmer at INRIA.
- Yearly meetings to decide on new directions and developments, esp. what this programmer should work on.
- Low membership fees.
- Expected about 20 members.

Inspired by the Python consortium, itself inspired by the Web consortium.

What we ended up with:

- Only 6 members today, not enough to fund a full-time programmer.
- Very few requests for specific developments.
- The most tangible benefit for members is that they benefit from more liberal licensing conditions:
 - Non-members: a somewhat restrictive open-source license
 - Members: a "free for all uses" license.

	Licensing	General
	conditions	sponsorship
Dassault Aviation		~
Dassault Systèmes	✓	
Intel		~
LexiFi	✓	
Microsoft	~	
XenSource	?	?

メロト メポト メヨト メヨト

Dual licensing is a good solution (except for integrating external contributions).

Not much interest in identifying developments useful for several members, and sharing the costs.

It's hard to have something to sell.

Examples of industrial uses of Caml

Perceived needs; the Caml consortium experiment

3 A quick look at the smart card industry

4 Conclusions

.∃ >



A tiny computer, usable as a security token:

- Low resources, inexpensive.
- Highly secure against software and hardware attacks.

10 years ago:

- Closed, propietary architectures.
- Programs developed by card manufacturers.
- Written in assembler or C.

Nowadays:

- Standardized, mostly open software architectures: MultOS, Java Card.
- Programming no longer restricted to manufacturers.
- The Java Card subset of Java.

Founded in 1999 with the intention of becoming an independent provider of software and solutions for smart cards and other secure embedded systems.

Security evaluation and consulting (\rightarrow Trusted Labs company):

- Common Criteria analyses;
- formal methods and verification;
- testing, testing tools.

High-security software components

- for smart cards;
- for card readers and terminals;
- for mobile phones.

- Low-level system components: management of persistent memory; cryptographic libraries.
- Java Card: virtual machine, run-time environment, APIs.
- The Global Platform protocols: secure communication channels, key management.
- Applications: EMV payment, ...

- For on-card, embedded code: C, occasional bits of assembler; Java Card
- For off-card code, e.g. development and verification tools: mostly Java; one product written in Caml (test generation and administration).

"Just like" Java, except:

- Fewer numerical types: no float, no double, no long; int is optional
 - \rightarrow compute with short.
- Objects allocated in persistent memory
 - \rightarrow objects as storage; transactions.
- No garbage collection

 \rightarrow allocate all needed space at installation-time; work in-place; little object-orientation.

```
private void credit(APDU apdu) {
    if ( ! pin.isValidated())
        ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);
    byte[] buffer = apdu.getBuffer();
    byte numBytes = buffer[IS07816.OFFSET_LC];
    byte byteRead = (byte)(apdu.setIncomingAndReceive());
    if (byteRead != 1) ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
    byte creditAmount = buffer[IS07816.OFFSET_CDATA];
    if ( (creditAmount > MAX_TRANSACTION_AMOUNT) || ( creditAmount < 0 ) )
        ISOException.throwIt(SW_INVALID_TRANSACTION_AMOUNT);
    if ( (balance + creditAmount) > MAX BALANCE )
        ISOException.throwIt(SW_EXCEED_MAXIMUM_BALANCE);
    balance = (short)(balance + creditAmount);
}
```

F.P. on-card:

- Not enough resources.
- "Java Card-ization" impossible for F.P.L.

F.P. on card terminals:

- Technically feasible.
- Which applications?
- F.P. in development environments:
 - A need for custom development tools.
 - A classic match for F.P.

C compilation and pre/post-optimization:

- Low quality of vendor-supplied C compilers.
- Temperamental 8-bit target architectures (e.g. 8051).

Verification and transformation on Java Card VM bytecode:

- Optimizing bytecode for size.
- Verification of API conformance, security properties, ...

Domain-specific languages and C / Java Card code generators:

• The example of APDU parsing

Application Protocol Data Units (APDUs)

The VERIFY command message is coded according to Table I - 24:

Code	Value
CLA	.00.
INS	20
P1	.00.
P2	Qualifier of the reference data (see Table I-33)
Lc	var.
Data	Transaction PIN Data
Le	Not present

Table I - 24- VERIFY Command Message

Table I-33 defines the qualifier of the reference data (P2):

b8	b7	B6	b5	b4	b3	B2	b1	Meaning
0	0	0	0	0	0	0	0	As defined in ISO/IEC 7816-41
1	0	0	0	0	0	0	0	Plaintext PIN, format as defined below
1	0	0	0	0	X	X	X	RFU for this specification
1	0	0	0	1	0	0	0	Enciphered PIN, format as defined in
								Book 2
1	0	0	0	1	0	х	x	RFU for this specification
1	0	0	0	1	1	x	x	RFU for the individual payment
								systems
1	0	0	1	х	x	x	x	RFU for the issuer

Table I - 25- VERIFY Command qualifier of reference data (P2)

Low-level, packet-based communication protocols between card and reader.

- Poorly specified protocols.
- Vulnerable to deadlocks.
- Hand-written parsers.
- Bugs in parsers can be security risks.

No APDU parser generator that I know of.

No formal language to specify APDU protocols that I know of.

Examples of industrial uses of Caml

Perceived needs; the Caml consortium experiment

3 A quick look at the smart card industry

4 Conclusions

3 ×

Caml in particular and F.P. in general are quite successful in the area of programs that manipulate programs (program verification and transformation, \dots)

This area is industrially relevant.

This is a niche, but an active, technologically important niche.

Caml in particular and F.P. in general are quite successful in the area of programs that manipulate programs (program verification and transformation, \dots)

This area is industrially relevant.

This is a niche, but an active, technologically important niche.

Caml in particular and F.P. in general are quite successful in the area of programs that manipulate programs (program verification and transformation, ...)

This area is industrially relevant.

This is a niche, but an active, technologically important niche.

Caml in particular and F.P. in general are quite successful in the area of programs that manipulate programs (program verification and transformation, ...)

This area is industrially relevant.

This is a niche, but an active, technologically important niche.