the MSR-INRIA Joint Centre

Jean-Jacques Lévy

September 15, 2010

msr-inria.inria.fr
Plan

1. Context

2. Track A
   - Math. Components
   - Security
   - TLA+

3. Track B
   - DDMF
   - ReActivity
   - Adaptative search
   - Image & video mining
Long cooperation among researchers
Organization

a rather complex system

- 7 research projects (in two tracks)
- 20 resident researchers
- non permanent researchers funded by the Joint Centre
- permanent researchers paid by INRIA or MSR
- operational support by INRIA Saclay
- 1 system manager (Guillaume Rousse, INRIA Saclay)
- 1 administrative assistant (Martine Thirion, Joint Centre)
- 1 deputy director (Pierre-Louis Xech, MS France)
- active support from MS France
People
PhD Students

• Francois GARILLOT
• Sidi OULD BIHA
• Iona PASCA
• Roland ZUMKELLER
• Pierre-Malo DENIELOU
• Nataliya GUTS
• Jérémy PLANUL
• Santiago ZANELLA
• Alexandre BENOIT
• Marc MEZZAROBA
• Nathalie HENRY (+)
• Nicolas MASSON
• Arnaud SPIVAK
• Aurélien TABARD

• Alexandre ARBALAEZ
• Alvaro FIALHO
• Adrien GAIDON

Post Docs

• Stéphane LE ROUX
• Guillaume MELQUIOND (*)
• Assia MAHBOUBI (*)
• Ricardo CORIN (*)
• Gurvan LE GUERNIC
• Eugen ZALINESCU
• Tamara REZK (*)
• Kaustuv CHAUDURI (*)
• Stefan GERHOLD
• Fanny CHEVALIER
• Niklas ELMQVIST
• Catherine LEDONTAL
• Tomer MOSCOVICH
• Theophanis TSANDILAS
• Nikolaus HANSEN (*)
• Neva CHERNIAVSKY

_______________________________
(*) Now on permanent INRIA position, (+) on permanent MSR position
Localization
the plateau de Saclay
= long term investment
Canteen

- low quality
- hyper noisy
- long line
- often on strike
- closed during August

Bus 269-02

- super <= 10am
- chaotic after
- 06-07 less many

Campus

6mn

RER B

Paris = 30mn
Mathematical components

Georges Gonthier, MSRC
Assia Mahboubi, INRIA Saclay/LIX
Andrea Asperti, Bologna
Y. Bertot, L. Rideau, L. Théry, Sidi Ould Biha, Iona Pasca, INRIA Sophia

François Garillot, MSR-INRIA (PhD)
Guillaume Melquiond, MSR-INRIA (postdoc)
Stéphane le Roux, MSR-INRIA (postdoc)
Benjamin Werner, INRIA Saclay/LIX,
Roland Zumkeller, LIX (PhD)

Computational proofs

– computer assistance for long formal proofs.
– reflection of computations into Coq-logic: ssreflect.

4-color
Appel-Haken

finite groups
Feit-Thompson

Kepler
Hales
Goals and results

Computational proofs

– bring proof tools to mathematicians
– apply software engineering to proof construction
– prove landmark result
– build on 4-color theorem experience
– ssreflect proof language and Coq plugin
– combinatorial components
– linear algebra components
– group theory components
Driving goals

Four-colour Theorem

Finite Group Theory
The finite group challenge

The Classification of Finite Simple Group

Feit-Thompson

Frobenius groups
Thompson factorisation
character theory
linear representation
Galois theory
linear algebra
polynomials

Sylow theorems
Isomorphism theorems

|G| odd
G simple
G \cong \mathbb{Z}_p
Section R_props.

(* The ring axioms, and some useful basic corollaries. *)

Hypothesis mult1x : forall x, 1 * x = x.
Hypothesis mult0x : forall x : R, 0 * x = 0.
Hypothesis plus0x : forall x : R, 0 + x = x.
Hypothesis minusxx : forall x : R, x - x = 0.
Hypothesis plusA : forall x1 x2 x3 : R, x1 + (x2 + x3) = x1 + x2 + x3.
Hypothesis plusC : forall x1 x2 : R, x1 + x2 = x2 + x1.
Hypothesis multA : forall x1 x2 x3 : R, x1 * (x2 * x3) = x1 * x2 * x3.
Hypothesis multC : forall x1 x2 : R, x1 * x2 = x2 * x1.
Hypothesis distrR : forall x1 x2 x3 : R, (x1 + x2) * x3 = x1 * x3 + x2 * x3.

Lemma plusCA : forall x1 x2 x3 : R, x1 + (x2 + x3) = x2 + (x1 + x3).
Proof. move=> *; rewrite !plusA; congr (_ + _); exact: plusC. Qed.

Lemma multCA : forall x1 x2 x3 : R, x1 * (x2 * x3) = x2 * (x1 * x3).
Proof. move=> *; rewrite !multA; congr (_ * _); exact: multC. Qed.

Lemma distrL : forall x1 x2 x3 : R, x1 * (x2 + x3) = x1 * x2 + x1 * x3.
Proof. by move=> x1 x2 x3; rewrite !(multC x1) distrR. Qed.

Lemma oppK : involutive opp.
Proof.
by move=> x; rewrite ![x]plus0x -(minusxx (- x)) plusC plusA minusxx plus0x. Qed.

Lemma multm1x : forall x, -1 * x = -x.
Proof.
movex; rewrite ![x]plus0x -(minusxx x) ![x]mult1x plusC plusCA plusA.
by rewrite distrR minusxx mult0x plus0x. Qed.
rewrite isum0 ?plus0x // => i'; rewrite andT; move/negbE->; exact: mult0x.
Qed.

Lemma matrix_transpose_mul : forall m n p (A : M_(m, n)) (B : M_(n, p)),
\forall (A * m B) \forall \forall B * m \forall A,
\forall (A + m B) \forall \forall B * m \forall A.
Proof. split\rightarrow k i; apply: eq_isumR \rightarrow j _; exact: multC. Qed.

Lemma matrix_multx1 : forall m n (A : M_(m, n)), A * m \forall \forall m A.
Proof.
move\rightarrow m n A; apply: matrix_transpose_inj.
by rewrite matrix_transpose_mul matrix_transpose_unit matrix_multx1.
Qed.

Lemma matrix_distrR : forall m n p (A1 A2 : M_(m, n)) (B : M_(n, p)),
(A1 + m A2) * m B = A1 + m B + m A2 + m B.
Proof.
move\rightarrow m n p A1 A2 B; split\rightarrow i k _; rewrite -isum_plus.
by apply: eq_isumR \rightarrow j _; rewrite -distrR.
Qed.

Lemma matrix_distrL : forall m n p (A : M_(m, n)) (B1 B2 : M_(n, p)),
A * m (B1 + m B2) = A * m B1 + A * m B2.
Proof.
move\rightarrow m n p A B1 B2; apply: matrix_transpose_inj.
rewrite matrix_transpose_plus lmatrix_transpose_mul.
by rewrite -matrix_distrR -matrix_transpose_plus.
Qed.

Lemma matrix_multA : forall m n p q
(A : M_(m, n)) (B : M_(n, p)) (C : M_(p, q)),
A * m (B + m C) = A * m B + m C.
Proof.
move\rightarrow m n p q A B C; split\rightarrow i l _.
transitivity (\forall sum.(k) \forall sum.(c) (A i + j B j * k C k i)).
rewrite exchange_isum; apply: eq_isumR \rightarrow j _; rewrite isum_distrL.
by apply: eq_isumR \rightarrow k _; rewrite multA.
by rewrite -eq_isumR \rightarrow j _; rewrite isum_distrR.
Qed.

Lemma perm_matrixM : forall n (s t : S_(n)),
perm_matrix (s * t) = perm_matrix s * perm_matrix t.
Proof.
move\rightarrow n; split\rightarrow i j _; rewrite (isum01 (s i)) // set11 mult1x -permM.
rewrite isum0 \rightarrow [i j]; first by rewrite plusC plus0x.
by rewrite andT; move/negbE->; rewrite mult0x.
Qed.

Lemma matrix_trace_plus : forall n (A B : M_(n, n)), \forall (A + m B) = \forall A + \forall B.
Proof. by move\rightarrow n A B; rewrite -isum_plus. Qed.

Lemma matrix_trace_scale : forall n x (A : M_(n, n)), \forall (x * s m A) = x * \forall A.
Proof. by move\rightarrow *; rewrite isum_distrL. Qed.

-DOS-- determinant.v ?% (1190,48) (cog)
Lemma determinant1 : forall n, \det (unit_matrix n) = 1.
Proof.
move=> n; have:= @determinant_perm n 1%G; rewrite odd_perm1 => /= <-.
apply: determinant_extensional; symmetry; exact: perm_matrix1.
Qed.

Lemma determinant_scale : forall n x (A : M_(n)),
\det (x *sm A) = x ^ n * \det A.
Proof.
move=> n x A; rewrite isum_distrL; apply: eq_isumR => s _.
by rewrite multCA iprod_mult iprod_id card_ordinal.
Qed.

Lemma determinantM : forall n (A B : M_(n)), \det (A *m B) = \det A * \det B.
Proof.
move=> n A B; rewrite isum_distrR.
pose AB (f : F_(n)) (s : S_(n)) i := A i (f i) * B (f i) (s i).
transitivity (\sum_(f) \sum_(s : S_(n)) (-1) ^ s * \prod_(i) AB f s i).
rewrte exchange_isum; apply: eq_isumR => s _.
by rewrite -isum_distrl distr_iprodA_isumA.
rewrite (isumID (fun f => uniq (fval f)) plusC isum0 ?plus0x => /= [If Uf].
rewrte (reindex_isum (fun s => val (pval s))); last first.
  have s0 : S_(n) := 1%G; pose uf (f : F_(n)) := uniq (fval f).
pose pf f := if insub uf f is Some s then Perm s else s0.
exists pf => /= f Uf; rewrite /pf (insubT uf Uf) //; exact: eq_fun_of_perm.
apply: eq_isum => [sls _]; rewrite ?(valP (pval s)) // isum_distrL.
rewrte (reindex_isum (mulg s)); last first.
  by exists (mulg s^(-1)) => t; rewrite ?mulKgv ?mulKg.
apply: eq_isumR => t _; rewrite iprod_mult multiA multCA multA multA multCA multA.
Progress

- 2 x 200 pages: preliminary group theory
- 25% Feit-Thompson, book 1 (~200 pages)
- 0% Feit-Thompson, book 2 (~200 pages)

http://coqfinitgroup.gforge.inria.fr/progress.html
Secure Distributed Computations and their Proofs

Cédric Fournet, MSRC
Karthik Bhargavan, INRIA
Ricardo Corin, INRIA Rocq.
Pierre-Malo Deniélo, INRIA Rocq.
G. Barthe, B. Grégoire, S. Zanella, INRIA Sophia

James Leifer, INRIA Rocq.
Jean-Jacques Lévy, INRIA Rocq.
Tamara Rezk, INRIA Sophia
Francesco Zappa Nardelli, INRIA Rocq.
Nataliya Guts, MSR-INRIA (PhD)
Jérémy Planul, MSR-INRIA (intern)

Distributed computations + Security

– programming with secured communications
– certified compiler from high-level primitives to low-level crypto-protocols
– formal proofs of probabilistic protocols
Secure Distributed Computations and their Proofs

- Secure Implementations for Typed Session Abstractions (v1 and v2)
- Cryptographic Enforcement of Information-Flow Security
- Secure Audit Logs
- Automated Verifications of Protocol Implementations (TLS)
- CertiCrypt: Formal Proofs for Computational Cryptography
Automated Verifications of an Implementation for TLS

- Firefox + Apache
- certified client CClient + certified server CServer
- Test functional features of Firefox + CServer and CClient + Apache
- Prove security property of CC + CS
- by translation to CryptoVerif [Bruno Blanchet]
- automatic translation from Caml + assertion to CryptoVerif (fs2cv)
Tools for formal proofs

Damien Doligez, INRIA Rocq.
Kaustuv Chaudhury, MSR-INRIA (postdoc)
Leslie Lamport, MSRSV
Stephan Merz, INRIA Lorraine

Natural proofs

– first-order set theory + temporal logic
– specification/verification of concurrent programs.
– tools for automatic theorem proving

TLA+

tools for proofs

Zenon
EXTENDS Naturals

(* First some general logical axioms pulled from the trusted base *)

(* The following is a specific instance of a theorem provable by the Peano axioms *)
THEOREM TwoIsNotOne ==
  2 ≠ 1
PROOF OMITTED

THEOREM NegElim ==
  ASSUME
  NEW CONSTANT A,
  A, ¬A
  PROVE
  FALSE
PROOF OMITTED

THEOREM ImplIntro ==
  ASSUME
  NEW CONSTANT A, NEW CONSTANT B,
  ASSUME A PROVE B
  PROVE A → B
PROOF OMITTED
(* The main definitions and lemmas (proofs omitted) *)

\text{Divides}(d, n) ==
\begin{align*}
& \forall d \in \text{Nat} \\
& \forall n \in \text{Nat} \\
& \exists q \in \text{Nat} : n = d \times q
\end{align*}

\text{THEOREM DivLemma ==}
\begin{align*}
& \forall d, n \in \text{Nat} : \text{Divides}(d, n) \Rightarrow \exists r \in \text{Nat} : n = r \times d
\end{align*}
\text{PROOF OMITTED}

\text{Prime}(x) ==
\begin{align*}
& \forall x \in \text{Nat} \\
& \forall \forall d \in \text{Nat} : \text{Divides}(d, x) \Rightarrow \bigvee d = 1 \\
& \bigvee d = x
\end{align*}

\text{PrimeNat} = \{x \in \text{Nat} : \text{Prime}(x)\}

\text{THEOREM TwoIsPrime ==} 2 \in \text{PrimeNat}
\text{PROOF OMITTED}

\text{THEOREM SquareLemma ==}
\begin{align*}
& \forall p \in \text{PrimeNat}, x \in \text{Nat} : \\
& \text{Divides}(p, x^2) \Rightarrow \text{Divides}(p, x)
\end{align*}
\text{PROOF OMITTED}
(**
* Main theorem: there is no irreducible rational number x/y whose
* square is 2.
*)

THEOREM SqrtTwoIrrational ==
\forall x, y \in \mathbb{N} : \text{Coprime}(x, y) \Rightarrow x^2 \neq 2 \cdot y^2

PROOF <1>1. ASSUME
    NEW x \in \mathbb{N},
    NEW y \in \mathbb{N},
    coprimality :: \text{Coprime}(x, y),
    main :: x^2 = 2 \cdot y^2

PROVE
    FALSE

PROOF <2>1. Divides(2, x)
    PROOF <3>1. Divides(2, x^2)
        BY <1>1
        <3>2. QED
            BY <3>1, TwoIsPrime, SquareLemma
    <2>2. Divides(2, y)
        PROOF <3>1. PICK r \in \mathbb{N} : x = 2 \cdot r
            BY <2>1, DivLemma
            <3>2. x^2 = 2 \cdot (2 \cdot r^2)
                BY <3>1
            <3>3. 2 \cdot y^2 = 2 \cdot (2 \cdot r^2)
                BY <1>1\text{main}, <3>2
            <3>4. y^2 = 2 \cdot r^2
1. \text{QED}
   BY \text{<1, ImplIntro, ForallIntro}

2. \text{QED}
   BY \text{<1, TwoIsPrime, SquareLemma}

2. \text{Divides}(2, y)
   \text{PROOF}
   \text{<1. PICK } r \in \text{Nat} : x = 2 \ast r
       BY \text{<2}, \text{DivLemma}
   \text{<2. } x^2 = 2 \ast (2 \ast r^2)
       BY \text{<3, 1}
   \text{<3. } 2 \ast y^2 = 2 \ast (2 \ast r^2)
       BY \text{<1}, \text{main, <2, 2}
   \text{<4. } y^2 = 2 \ast r^2
       BY \text{<3, 2, LeftCancellationLemma}
   \text{<5. QED}
       BY \text{<3, 3, TwoIsPrime, SquareLemma}

3. \text{~(Divides}(2, y))
   \text{PROOF}
   \text{<1. } \forall d \in \text{Nat} : (\text{Divides}(d, x) \land \text{Divides}(d, y)) \Rightarrow d = 1
       BY \text{<1}, \text{coprimality}
   \text{<2. } 2 = 1
       BY \text{<2}, \text{<2, 2, <3, 1}
   \text{<3. QED}
       BY \text{<3}, \text{TwoIsNotOne}
   \text{<4. QED}
       BY \text{<2, <3, NegElim}

   \text{<1. QED}
       BY \text{<1, ImplIntro, ForallIntro}
MODULE Peterson

Not(i) == IF i = 0 THEN 1 ELSE 0

********
--algorithm Peterson {
    variables flag = [i \in \{0, 1\} \mapsto FALSE], turn = 0;
    process (proc \in \{0,1\}) {
        a0: while (TRUE) {
            a1: flag[self] := TRUE;
            a2: turn := Not(self);
            a3a: if (flag[Not(self)]) {goto a3b} else {goto cs} ;
            a3b: if (turn = Not(self)) {goto a3a} else {goto cs} ;
            cs: skip; \* critical section
            a4: flag[self] := FALSE;
        } \* end while
    } \* end process
} ********

AXIOM Arithmetic == 0 # 1

\* BEGIN TRANSLATION
VARIABLES flag, turn, pc

vars == <flag, turn, pc>

ProcSet == ({0,1})

Init == (* Global variables *)
    \forall flag = [i \in \{0, 1\} \mapsto FALSE]
    \forall turn = 0
    \forall pc = [self \in ProcSet \mapsto CASE self \in \{0,1\} \mapsto "a0"]

a0(self) == \forall pc[self] = "a0"
    \forall pc' = [pc \except ![self] = "a1"]
    \forall UNCHANGED <flag, turn>
a0(self) == /
\ pc[self] = "a0"
\ pc' = [pc EXCEPT ![self] = "a1"]
\ UNCHANGED << flag, turn >>

a1(self) == /
\ pc[self] = "a1"
\ flag' = [flag EXCEPT ![self] = TRUE]
\ pc' = [pc EXCEPT ![self] = "a2"]
\ UNCHANGED turn

a2(self) == /
\ pc[self] = "a2"
\ turn' = Not(self)
\ pc' = [pc EXCEPT ![self] = "a3a"]
\ UNCHANGED flag

a3a(self) == /
\ pc[self] = "a3a"
\ IF flag[Not(self)]
\ THEN /
\ pc' = [pc EXCEPT ![self] = "a3b"]
\ ELSE /
\ pc' = [pc EXCEPT ![self] = "cs"]
\ UNCHANGED << flag, turn >>

a3b(self) == /
\ pc[self] = "a3b"
\ IF turn = Not(self)
\ THEN /
\ pc' = [pc EXCEPT ![self] = "a3a"]
\ ELSE /
\ pc' = [pc EXCEPT ![self] = "cs"]
\ UNCHANGED << flag, turn >>

cs(self) == /
\ pc[self] = "cs"
\ TRUE
\ pc' = [pc EXCEPT ![self] = "a4"]
\ UNCHANGED << flag, turn >>

a4(self) == /
\ pc[self] = "a4"
\ flag' = [flag EXCEPT ![self] = FALSE]
\ pc' = [pc EXCEPT ![self] = "a0"]
## Logics in track A

<table>
<thead>
<tr>
<th>Math. components</th>
<th>Coq</th>
<th>higher-order + reflection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>PV/CV</td>
<td>applied pi-calculus + stochastic</td>
</tr>
<tr>
<td>Spec. / Verif.</td>
<td>TLA+</td>
<td>1st order + ZF + temporal</td>
</tr>
</tbody>
</table>
Track B

Computational Sciences
Scientific Information Interaction
Dynamic dictionary of math functions

Bruno Salvy, INRIA Rocq.,
Alin Bostan, INRIA Rocq.,
Frédéric Chyzak, INRIA Rocq.

Henry Cohn, [Theory Group] MSRR
Alexandre Benoît, MSR-INRIA (intern)
Marc Mezzarobba, MSR-INRIA (intern)

Computer Algebra and Web for useful functions,

– dynamic tables of their properties.
– generation of programs to compute them.

Maple 11
Among the most cited documents in the scientific literature.

Thousands of useful mathematical formulas, computed, compiled and edited by hand.

Started between 60 and 30 years ago.
9. Bessel Functions of Integer Order

Mathematical Properties

Notation

The tables in this chapter are for Bessel functions of integer order; the text treats general orders. The conventions used are:

- \( z = x + iy; \) x, y real.
- \( n \) is a positive integer or zero.
- \( \nu \) are unrestricted except where otherwise indicated; \( \nu \) is supposed real in the sections devoted to Kelvin functions 9.9, 9.10, and 9.11.

The notation used for the Bessel functions is that of Watson [11.6] and the British Association and Royal Society Mathematical Tables. The function \( Y_n(z) \) is often denoted \( N_n(z) \) by physicists and European workers.

Other notations are those of:

- Aldis, Airey:
  - \( G_n(x) \) for \(-\frac{1}{\pi}Y_n(x), K_n(x) \) for \((-\nu)K_n(x)\).
- Clifford:
  - \( C_n(x) \) for \((-\nu)J_n(\nu x)\).
- Gray, Mathews and MacRobert [9.9]:
  - \( Y_n(x) \) for \( \frac{1}{\pi}Y_n(x) + (2\pi n - 2\nu)J_n(x) \).
  - \( \bar{Y}_n(x) \) for \( \nu J_n(\nu x) \).
  - \( G_n(x) \) for \( \frac{1}{\pi}H_n^{(1)}(x) \).
- Jahnke, Emde and Lösch [9.32]:
  - \( \Lambda_n(x) \) for \( (\nu+1)J_n(x) \).
- Jeffreys:
  - \( H_n(x) \) for \( H_n^{(1)}(x) \).
  - \( H_n(x) \) for \( H_n^{(2)}(x) \).
  - \( K_n(x) \) for \( 2\pi x K_n(x) \).
- Heine:
  - \( \bar{J}_n(x) \) for \(-\frac{1}{\pi}Y_n(x)\).
- Neumann:
  - \( \Phi_n(x) \) for \( \frac{1}{\pi}Y_n(x) + (2\pi n - 2\nu)J_n(x) \).
- Whitaker and Watson [9.18]:
  - \( K_n(x) \) for \( \cos(\nu x)K_n(x) \).

Bessel Functions \( J \) and \( Y \)

9.1. Definitions and Elementary Properties

Differential Equation

\[
\frac{d^2w}{dz^2} + \frac{1}{z} \frac{dw}{dz} + \left( \nu^2 - \frac{1}{z^2} \right) w = 0
\]

Solutions are the Bessel functions of the first kind \( J_n(z) \), of the second kind \( Y_n(z) \) (also called Weber's function) and of the third kind \( H_\nu^{(1)}(z) \), \( H_\nu^{(2)}(z) \) (also called the Hankel functions). Each is a regular (holomorphic) function of \( z \) throughout the \( z \)-plane cut along the negative real axis, and for fixed \( z(\neq 0) \) each is an entire (integral) function of \( \nu \). When \( \nu = \pm n \), \( J_n(z) \) has no branch point and is an entire (integral) function of \( z \).

Important features of the various solutions are as follows: \( J_n(z) \) for any bounded range of \( \nu \) is bounded as \( z \to 0 \). \( J_n(z) \) and \( J_{-\nu}(z) \) are linearly independent except when \( \nu = n \) is an integer. \( J_n(z) \) and \( Y_n(z) \) are linearly independent for all values of \( \nu \).

\( H_\nu^{(1)}(z) \) tends to zero as \( |z| \to \infty \) in the sector \( 0 < \arg z < \pi \); \( H_\nu^{(2)}(z) \) tends to zero as \( |z| \to \infty \) in the sector \( -\pi < \arg z < 0 \). For all values of \( \nu, H_\nu^{(1)}(z) \) and \( H_\nu^{(2)}(z) \) are linearly independent.

Relations Between Solutions

9.1.2

\[
Y_n(z) = \frac{J_n(z) \cos(\nu \theta) - J_{-\nu}(z) \sin(\nu \theta)}{\sin(\nu \theta)}
\]

The right of this equation is replaced by its limiting value if \( \nu \) is an integer or zero.

9.1.3

\[
H_\nu^{(1)}(z) = J_\nu(z) + iY_\nu(z)
\]

9.1.4

\[
H_\nu^{(2)}(z) = J_\nu(z) - iY_\nu(z)
\]

9.1.5

\[
J_n(z) = (-1)^n J_{-n}(z) \quad Y_n(z) = (-1)^n Y_{-n}(z)
\]

9.1.6

\[
\frac{d}{dz}H_\nu^{(1)}(z) = \frac{d}{dz}H_\nu^{(2)}(z) = \frac{d}{dz}H_\nu^{(1)}(z) = -\nu\sin(\nu \theta)H_\nu^{(1)}(z)
\]
Limiting Forms for Small Arguments

When $x$ is fixed and $z \to 0$

9.1.17 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.18 $Y_0(x) = -i \left( \frac{1}{x} \right)^{1/2} \left( \frac{1}{x} \right)^{1/2}$

9.1.19 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.20 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.21 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.22 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.23 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.24 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.25 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

9.1.26 $J_0(x) = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta = \frac{1}{x} \int_0^\infty \cos \left( \frac{x}{\theta} \right) d\theta$

In the last integral the path of integration must lie to the left of the points $t=0,1,2,\ldots$
9.1.72
\[ \lim_{\lambda \to 0}\left(\lambda \int_{-\infty}^{\infty} \frac{e^{-\lambda x}}{\sqrt{2\pi}} dx\right) = -\frac{1}{\sqrt{\pi}} \]
\[ Y_n(x) (x > 0) \]

For \( P_\lambda^m \) and \( Q_\lambda^m \), see chapter 8.

9.1.73
Continued Fractions
\[ \frac{J_\lambda(z)}{J_{\lambda+1}(z)} = \frac{1}{z} + \frac{1}{z} + \frac{1}{z} + \frac{1}{z} + \cdots \]
9.1.74
Multiplication Theorem
\[ J_m(x \pm y) = \frac{1}{\sqrt{\pi}} \sum_{n=-\infty}^{\infty} \frac{(-1)^n}{n!} (x^2 - y^2)_{n} \cos (n x) \]
\[ \Phi = J \]
If \( \Phi = J \) and the upper signs are taken, the restriction on \( \lambda \) is unnecessary.

9.1.75
Neumann's Addition Theorem
\[ \Phi_n(x \pm y) = \sum_{m=-\infty}^{\infty} \Phi_{n+m}(y) J_m(x) \]
\[ (|x| < |y|) \]

9.1.76
Degenerate Form (\( x = 0 \))
\[ J_m(x) = 1 + \frac{1}{2} \sum_{k=1}^{\infty} \frac{1}{k^2} J_k(x) \]
9.1.77
Jost's
\[ J_n(x) = \sum_{m=-\infty}^{\infty} \Phi_{n+m}(y) J_m(x) \]
\[ (n \geq 1) \]
9.1.78
Gegenbauer's
\[ \Phi_n(x) \]
\[ \sin \theta = \frac{1}{\sqrt{\pi}} \sum_{m=-\infty}^{\infty} \Phi_{n+m}(y) \cos (m \theta) \]
\[ (|x| < |y|) \]

9.1.79
Gegenbauer's
\[ \Phi_n(x) \]
\[ \Phi_n(x) = 2^n T_n(x) \sum_{m=0}^{n} \frac{(-1)^m}{m!} T_m(x) J_m(x) \]
\[ C_n^{(1)}(\cos \alpha) \]
\[ (\nu \neq 0, 1, 2, \ldots, |\nu^b| < |\nu|) \]

9.1.80
In 9.1.79 and 9.1.80,
\[ w = \sqrt{1 + \frac{a^2}{x^2} - 2 \nu a x} \]
\[ u = 0 \cos \alpha = u \cos x, v \sin \alpha = v \sin x \]
the branches being chosen so that \( w \to a \) and \( v \to 0 \) as \( \nu \to 0 \). \( C_n^{(1)}(\cos \alpha) \) is Gegenbauer's polynomial

9.1.81
Gegenbauer's addition theorem.

9.1.82
Neumann's Expansion of an Arbitrary Function in a Series of Bessel Functions
\[ f(x) = a_0 J_0(x) + 2 \sum_{k=1}^{\infty} a_k J_k(x) \]
\[ (|x| < \infty) \]
where \( c \) is the distance of the nearest singularity of \( f(x) \) from \( x = 0 \).

9.1.83
and \( O_\nu(x) \) is Neumann's polynomial. The latter is defined by the generating function
\[ \frac{1}{\cosh \theta} \]

9.1.84
\[ \sinh \theta = \frac{1}{\sqrt{\pi}} \sum_{m=-\infty}^{\infty} \Phi_{n+m}(y) \sin (m \theta) \]
\[ (|x| < |y|) \]
\[ O_\nu(x) \]
\[ (\nu \neq 0, 1, 2, \ldots) \]
\[ O_\nu(x) \]

9.1.86
The more general form of expansion
\[ f(x) = a_0 J_0(x) + 2 \sum_{k=1}^{\infty} a_k J_k(x) \]
\[ (\nu \neq 0, 1, 2, \ldots, |\nu^b| < |\nu|) \]

9.2.126
Laplace Transforms
\[ f(t) \]
\[ F(s) \]
\[ \int_{0}^{\infty} \exp(-st) \Phi(t) \]
\[ \Phi(t) \]
\[ \int_{0}^{\infty} \exp(-st) \]

29.4.1
For the definition of the Laplace-\( \lambda \)-functions see [29.7]. In practice, Laplace-\( \lambda \)-functions are often written as ordinary Laplace transforms involving Dirac's delta function \( \delta(t) \).

9. Bessel Functions of Integer Order

Mathematical Properties

is chapter are for Bessel func-

tions used are:


te or zero.

icted except where otherwise

ised real in the sections devoted

9.9, 9.10, and 9.11.

ed for the Bessel functions is

15] and the British Association

Mathematical Tables. The

ten denoted $N_{\nu}(z)$ by physicists

are those of:

Bessel Functions $J$ and $Y$

9.1. Definitions and Elementary Properties

Differential Equation

9.1.1

$z^2 \frac{d^2w}{dz^2} + z \frac{dw}{dz} + (z^2 - \nu^2)w = 0$

Solutions are the Bessel functions of the first kind

$J_{\pm \nu}(z)$, of the second kind $Y_{\nu}(z)$ (also called

Weber’s function) and of the third kind $H^{(1)}_{\nu}(z)$, $H^{(2)}_{\nu}(z)$

(also called the Hankel functions). Each is a

regular (holomorphic) function of $z$ throughout

the $z$-plane cut along the negative real axis, and

for fixed $z(\neq 0)$ each is an entire (integral) func-
tion of $\nu$. When $\nu = \pm n$, $J_{\nu}(z)$ has no branch point

and is an entire (integral) function of $z$. 
Dynamic dictionary of math functions

Computer algebra:

- **classic**: polynomial to represent their roots + following tools: euclidian division, Euclid algorithm, Gröbner bases.

- **modern**: linear differential equation as data structures to represent their solutions [SaZi94, ChSa98, Chyzak00, MeSa03, Salvy05] with same tools as classical case but non-commutative.

- **prototype** ESF at http://algo.inria.fr/esf (65% of Abramowitz-Stegun)

- **todo**: interactivity, integral transforms, parametric integrals.
Welcome to this interactive site on Mathematical Functions, with properties, truncated expansions, numerical evaluations, plots, and more. The functions currently presented are elementary functions and special functions of a single variable. More functions — special functions with parameters, orthogonal polynomials, sequences — will be added with the project advances.

Select a special function from the list

- Help on selecting and configuring the mathematical rendering
- DDMF developers list
- Motivation of the project
- List of related projects
- Release history

The DDMF project (2008–2010) is hosted and supported by the Microsoft Research – INRIA Joint Centre.

http://ddmf.msr-inria.inria.fr

Powered by DynaMoW.

Bruno Salvy
The Dynamic Dictionary of Mathematical Functions
ReActivity

Wendy Mackay, INRIA Saclay, J.-D. Fekete, INRIA Saclay, Mary Czerwinski, MSRR, George Robertson, MSRR

Michel Beaudouin-Lafon, Paris 11, Olivier Chapuis, CNRS, Pierre Dragicevic, INRIA Saclay, Emmanuel Pietriga, INRIA Saclay, Aurélien Tabard, Paris 11 (PhD)

Logs of experiments for biologists, historians, other scientists

– mixed inputs from lab notebooks and computers,
– interactive visualization of scientific activity,
– support for managing scientific workflow.
ReActivity

Programme:

– Log platform and infrastructure for data collection and aggregation
  ◦ common format & share experiences,
  ◦ apply our own visualisation tools to the logged data
– Visualisation and instrumentation of scientific data logs,
  ◦ Visualisation of scaled to month-long or longer logs,
  ◦ strategies of interaction and navigation for meaningful sampling of data
– Mining of desktop data and interactions with visualised activities
  ◦ Design highly interactive tools for scientists to understand and interact with their past activities
  ◦ Create high-level interactive reflexive views that can be manipulated and reused)

Update:

– interactive wall and collaborative workflow
Adaptive Combinatorial Search for E-science

Parallel constraint programming and optimization for very large scientific data

– improve the usability of Combinatorial Search algorithms.
– automate the fine tuning of solver parameters.
– parallel solver: “disolver”
Adaptive Combinatorial Search for E-science

- **constraint programming**: learn instance-dependent variable ordering
- **evolutionary algorithms**: use multi-armed bandit algorithms and extreme values statistics
- **continuous search spaces**: use local curvature
Image and video mining for science and humanities

Jean Ponce, ENS
Andrew Blake, MSRC
Patrick Pérez, INRIA Rennes
Cordelia Schmid, INRIA Grenoble

Computer vision and Machine learning for:

- *sociology*: human activity modeling and recognition in video archives
- *archaeology and cultural heritage preservation*: 3D object modeling and recognition from historical paintings and photographs
- environmental *sciences*: change detection in dynamic satellite imagery
## Sciences in track B

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDMF</td>
<td>computer algebra</td>
<td>hard sciences</td>
</tr>
<tr>
<td>Adapt. search</td>
<td>constraints, machine learning</td>
<td>hard sciences, biology</td>
</tr>
<tr>
<td>Reactivity</td>
<td>chi + visualisation</td>
<td>soft sciences, biology</td>
</tr>
<tr>
<td>I.V. mining</td>
<td>computer vision</td>
<td>humanities, environment</td>
</tr>
</tbody>
</table>
Future
Future

- 30 resident researchers
- tight links with French academia (phD, post-doc)
- develop useful research for scientific community
- provide public tools (BSD-like license)
- become a new and attractive pole in CS research
- and source of spin off companies