the MSR-INRIA Joint Centre

Jean-Jacques Lévy
June 2, 2008
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)
• 12 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
- Alexandro ARBALAEZ
- Alvaro FIALHO
- Francois GARILLOT
- Sidi OULD BIHA
- Iona PASCA
- Arnaud SPIVAK
- Nicolas MASSON
- Nathalie HENRY
- Nataliya GUTS
- Santiago ZANELLA

### Post Docs
- Stéphane LEROUX
- Guillaume MELQUIOND
- Roland ZUMKELLER
- Assia MAHBOUBI (*)
- Aurélien TABARD
- Catherine LEDONTAL
- Niklas ELMQVIST
- Gurvan LE GUERNIC
- Eugen ZALINESCU
- Ricardo CORIN (*)
- Tamara REZK (*)

### Interns
- Jorge Martin PEREZ–ZERPA
- Sébastien MIGNOT
- Fabien TEYTAUD
- Alexandre BENOIT
- Pratik PODDAR
- Sean McLAUGHLIN
- Etienne MIRET
- Enrico TASSI
- Fei Li
- Yoann COLDFLY
- Jérémy PLANUL

(*) Now on permanent INRIA position
Track B

Computational Sciences
Scientific Information Interaction
Dynamic dictionary of math functions

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

Computer Algebra and Web for useful functions,

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

Maple 11
9. Bessel Functions of Integer Order

Mathematical Properties

Bessel Functions J and Y

9.1. Definitions and Elementary Properties

Differential Equation

9.1.1

\[ z^2 \frac{d^2 y}{dz^2} + z \frac{dy}{dz} + (z^2 - \nu^2)y = 0 \]

Solutions are the Bessel functions of the first kind \( J_\nu(z) \), 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) 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: \( J_\nu(z) \) is bounded as \( z \to 0 \) in any bounded range of \( \nu \), \( J_\nu(z) \) and \( J_{\nu \pm 1}(z) \) are linearly independent except when \( \nu \) is an integer. \( J_{\nu \pm 1}(z) \) and \( Y_{\nu \pm 1}(z) \) are linearly independent for all values of \( \nu \).

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

Relations Between Solutions

9.1.2

\[ Y_\nu(z) = \frac{J_\nu(z) \cos(\nu \pi) - J_{1-\nu}(z)}{\sin(\nu \pi)} \]

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

9.1.3

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

9.1.4

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

9.1.5

\[ J_\nu(z) = (z / \nu)^{1/2} J_{1/2}(z) \]

9.1.6

\[ H^{(1)}\nu(z) = z^{-1/2} H_{1/2}^{(1)}(z) \]

\[ H^{(2)}\nu(z) = -z^{-1/2} H_{1/2}^{(2)}(z) \]

9.1.18 Integral Representations

\[ J_\nu(x) = \frac{1}{\pi} \int_0^\infty \cos(x \sin \theta) d\theta \cdot \int_0^{\pi/2} \cos(x \cos \theta) d\theta \]

9.1.19

\[ Y_\nu(x) = \frac{4}{\pi} \int_0^\infty \cos(x \cos \theta) \left( \gamma + \ln(2x \sin \theta) \right) d\theta \]

9.1.20

\[ J_\nu(x) = \frac{1}{\pi} \int_{\pi/2}^{3\pi/2} \cos(x \cos \theta) \sin^\nu \theta d\theta = \frac{2^{\nu+1}}{(\nu+1)!} \left( \frac{x}{2} \right)^{\nu+1} \int_0^1 (1-t^2)^{\nu+1} \cos(x \sqrt{1-t^2}) dt \quad (\nu > -\frac{1}{2}) \]

9.1.21

\[ J_\nu(x) = \frac{1}{\pi} \int_0^\infty \cos(x \sin \theta - \nu \theta) d\theta \]

\[ = \frac{1}{\pi} \int_0^\infty e^{i \nu \sin \theta - x \sin \theta} d\theta \quad (\nu \in \mathbb{R}, \theta \neq 0) \]

9.1.22

[Formulas for Derivatives]

\[ J_\nu(x) = \frac{1}{\pi} \int_0^\infty \cos(x \sin \theta - \nu \theta) d\theta \]

\[ = \frac{1}{\pi} \int_0^\infty e^{i \nu \sin \theta - x \sin \theta} d\theta \quad (\nu \in \mathbb{R}, \theta \neq 0) \]

9.1.23

\[ J_\nu(x) = \frac{1}{\pi} \int_0^\infty \sin(x \cosh t) dt \quad (x > 0) \]

\[ Y_\nu(x) = \frac{1}{\pi} \int_0^\infty \cosh(x \sinh t) dt \quad (x > 0) \]

9.1.24

\[ J_\nu(x) = \frac{2^{\nu} (3) \nu + 1}{\pi (\nu + 1)!} \int_0^\infty \cos(x \sin t) d\theta \cdot \int_0^{\pi/2} \cos(x \cos \theta) d\theta \quad (x > 0) \]

9.1.25

\[ H^{(1)}_\nu(x) = \frac{1}{\pi} \int_0^\infty \sin(x \sin t) \left( e^{\nu \sin t} - 1 \right) dt \quad (\nu > 0, x > 0) \]

9.1.26

\[ H^{(2)}_\nu(x) = \frac{1}{\pi} \int_0^\infty \sin(x \cos t) \left( e^{\nu \cos t} - 1 \right) dt \quad (\nu > 0, x > 0) \]

In the last integral the path of integration must lie to the left of the points \( t = 0, 1, 2, \ldots \).
9.1.72
\[ \lim (r!)Q_n^\alpha \left( \cos \frac{\pi}{2} \right) = -\frac{1}{\pi} Y_n^\alpha (x) \quad (x > 0) \]

For \( P_n^\alpha \) and \( Q_n^\alpha \), see chapter 8.

9.1.73
\[ \frac{J_n(x)}{J_n(0)} = \frac{1}{2\pi} \frac{1}{2i(n+1)x^{-1}} \frac{1}{2(n+2)x^{-1}} \frac{1}{2(n+3)x^{-1}} \cdots \]
\[ = \frac{1}{2^0} \frac{1}{2^1} \frac{1}{2^2} \frac{1}{2^3} \frac{1}{2^4} \cdots \]

Multiplication Theorem

9.1.74
\[ \psi_n(x) = \lambda^n \sum \frac{(\pi x)^{n-1}}{n!} \psi_{\text{real}}(x) \]
\[ (\lambda^2 - 1)(\pi^2 x^2) \]

If \( \psi = J \) and the upper signs are taken, the restriction on \( \lambda \) is unnecessary.

This theorem will furnish expansions of \( \psi_n(x) \) in terms of \( \psi_{\text{real}}(x) \).

Neumann's Addition Theorems

9.1.75
\[ \psi_n(x \pm \delta) = \sum \psi_{n_m}(x) J_m(\delta) \quad (|\delta| < |x|) \]

The restriction \(|\delta| < |x|\) is unnecessary when \( \psi = J \) and \( \nu \) is an integer or zero. Special cases are

9.1.76
\[ J_n(x) = \sum \frac{J_n(x)}{\nu!} \]

9.1.77
\[ 0 = \sum_{\nu=0}^\infty (-1)^\nu J_{\nu+1}(x) J_{\nu+1}(x) \quad (n \geq 1) \]

9.1.78
\[ J_{n}(x) = \sum_{\nu=0}^\infty J_{\nu+1}(x) J_{\nu+1}(x) + \sum_{\nu=0}^\infty (-1)^\nu J_{\nu+1}(x) J_{\nu+1}(x) \]

Grad's

9.1.79
\[ \psi_n(x) = \frac{\cos \pi x}{\pi} \sum \psi_{\nu}(x) J_{\nu}(x) \cos \left( \pi (\nu - \nu') \right) \]

Gegenbauer's

9.1.80
\[ \frac{d}{dx} \left( J \right) = 2J(x) \sum_{\nu=0}^\infty \frac{J_{\nu+k}(x) J_{\nu+k}(x)}{\nu!} \]
9. Bessel Functions of Integer Order

Mathematical Properties

Bessel Functions J and Y

9.1. Definitions and Elementary Properties

Differential Equation

\[
J_\nu(x) = \frac{d}{dx} \left( x^{\nu} J_\nu(x) \right) = x^{\nu} J_{\nu+1}(x)
\]

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

Important features of the various solutions are as follows: \( J_\nu(x) \) is bounded as \( x \to 0 \) in any bounded range of \( \nu \). \( J_\nu(x) \) and \( Y_\nu(x) \) are linearly independent except when \( \nu \) is an integer. \( J_\nu(x) \) and \( Y_\nu(x) \) are linearly independent for all values of \( \nu \).

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

Relations Between Solutions

9.1.2

\[
Y_\nu(x) = \frac{J_\nu(x) \cos(\nu x) - J_{\nu+1}(x)}{\sin(\nu x)}
\]

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

9.1.3

\[
H^{(1)}_\nu(x) = J_\nu(x) + iY_\nu(x)
\]

\[
H^{(2)}_\nu(x) = J_\nu(x) - iY_\nu(x)
\]

9.1.4

\[
H^{(1)}_\nu(x) = e^{i\nu x} J_\nu(x)
\]

\[
H^{(2)}_\nu(x) = e^{-i\nu x} J_\nu(x)
\]

\[
J_{-\nu}(x) = (-1)^\nu J_\nu(x)
\]

\[
Y_{-\nu}(x) = (-1)^\nu Y_\nu(x)
\]

\[
H^{(1)}_{-\nu}(x) = e^{-i\nu x} H^{(1)}_\nu(x)
\]

\[
H^{(2)}_{-\nu}(x) = e^{i\nu x} H^{(2)}_\nu(x)
\]
9. Bessel Functions of Integer Order

Mathematical Properties

Bessel Functions $J$ and $Y$

9.1. Definitions and Elementary Properties

Differential Equation

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

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) function of $\nu$. When $\nu = \pm n$, $J_{\nu}(z)$ has no branch point and is an entire (integral) function of $z$. 

is chapter are for Bessel func-
order; the text treats general
ations used are:

integer or zero.
icted except where otherwise
used real in the sections devoted
9.9, 9.10, and 9.11.
led for the Bessel functions is
15] and the British Association
Mathematical Tables. The
ten denoted $N_{\nu}(z)$ by physicists
are those of:
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.
ReActivity

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.

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

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DDMF</strong></td>
<td><strong>computer algebra</strong></td>
<td><strong>hard sciences</strong></td>
</tr>
<tr>
<td><strong>Adapt. search</strong></td>
<td><strong>constraints, machine learning</strong></td>
<td><strong>hard sciences, biology</strong></td>
</tr>
<tr>
<td><strong>Reactivity</strong></td>
<td><strong>chi + visualisation</strong></td>
<td><strong>soft sciences, biology</strong></td>
</tr>
<tr>
<td><strong>I.V. mining</strong></td>
<td><strong>computer vision</strong></td>
<td><strong>humanities, environment</strong></td>
</tr>
</tbody>
</table>
Track A
Software Security
Trustworthy Computing
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
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 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 ![2][x]plus0x -(minusxx (- x)) plusC plusA minusxx plus0x.
  Qed.

Lemma multm1x : forall x, -1 * x = -x.
  Proof.
  move=> x; rewrite ![ * x]plus0x -(minusxx x) ![1][x]mult1x plusC plusA.
  by rewrite !distrR minusxx mult0x plus0x.
  Qed.

Lemma mult_opp : forall x1 x2 : R, (- x1) * x2 = -(x1 * x2).
  Proof. by move=> *; rewrite !multm1x -multA multm1x. Qed.

Lemma opp_plus : forall x1 x2 : R, - (x1 + x2) = - x1 - x2.
  Proof.
  by move=> x1 x2; rewrite !multm1x multC distrR -!(multC -1) !multm1x.
  Qed.

Lemma RofSnE : forall n, RofSn n = n + 1.
  Proof. by elim=> /; rewrite plus0x. Qed.

Lemma Raddn : forall m n, (m + n)\%N = m + n : R.
  Proof.
  move=> m n; elim: m := ![m IHm]; first by rewrite plus0x.
  by rewrite !RofSnE IHm plusC plusA.
  Qed.

Lemma Rsubn : forall m n, m >= n -> (m - n)\%N = m - n : R.
  -(DOS)-- determinant.v 42% (709,42) (cog)
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 -{2}[x]plus0x -(minusxx (- x)) plusC plusA minusxx plus0x.
Qed.

Lemma multm1x : forall x, -1 * x = -x.
Proof.
move=> x; rewrite -[_ * x]plus0x -(minusxx x) -{1}[x]mult1x plusC plusCA plusA.
by rewrite -distrR minusxx mult0x plus0x.
Qed.
Lemma Rsubn : forall m n, m >= n => (m - n)\in \mathbb{N} = m - n : R.
Proof.
move=> m n; move/leq_add_sub=> Dm.
by rewrite \{-2\}Dm Raddn -plusA plusCA minusxx plusC plus0x.
Qed.

Lemma Rmuln : forall m n, (m * n)\in \mathbb{N} = m * n : R.
Proof.
move=> m n; elim: m => /= [\{IHm\}]; first by rewrite mult0x.
by rewrite Raddn RofSnE IHn distrR mult1x plusC.
Qed.

Lemma RexpSnE : forall x n, RexpSn x n = x ^ n * x.
Proof. by move=> x; elim=> /= [\{ \rightarrow \rightarrow \}; rewrite mult1x. Qed.

Lemma mult_exp : forall x1 x2 n, (x1 * x2) ^ n = x1 ^ n * x2 ^ n.
Proof. by move=> x1 x2; elim=> /= n IHn; rewrite !RexpSnE IHn -!multA (multCA x1).
Qed.

Lemma exp_addn : forall x n1 n2, x ^ (n1 + n2) = x ^ n1 * x ^ n2.
Proof. move=> x n1 n2; elim: n1 => /= [\{IHn\}]; first by rewrite mult1x.
by rewrite !RexpSnE IHn multC multCA multA.
Qed.

Lemma Rexpn : forall m n, (m ^ n)\in \mathbb{N} = m ^ n : R.
Proof. by move=> m; elim=> /= n IHn; rewrite Rmuln RexpSnE IHn multC. Qed.

Lemma exp0n : forall n, 0 < n => 0 ^ n = 0.
Proof. by move=> [\{IHn\}]; rewrite multC mult0x. Qed.

Lemma exp1n : forall n, 1 ^ n = 1.
Proof. by elim=> /= n IHn; rewrite RexpSnE IHn mult1x. Qed.

Lemma exp_muln : forall x n1 n2, x ^ (n1 * n2) = (x ^ n1) ^ n2.
Proof. move=> x n1 n2; rewrite multnC; elim: n2 => /= n2 IHn.
by rewrite !RexpSnE exp_addn IHn multC.
Qed.

Lemma sign_odd : forall n, (-1) ^ odd n = (-1) ^ n.
Proof. move=> n; rewrite \{-2\}[n]odd_double_half addnC double_mul2 exp_addn exp_muln.
by rewrite /= multtmx oppK expln mult1x.
Qed.

Lemma sign_addb : forall b1 b2, (-1) ^ (b1 (+) b2) = (-1) ^ b1 * (-1) ^ b2.

Lemma sign_permM : forall d (s t : permType d),
-\(d(S)\) determinant.v 45% (768,61) (c0a)
Lemma matrix_transpose_mul : forall m n p (A : M_(m, n)) (B : M_(n, p)),
     \forall (A *m B) = \forall B *m \forall A.
Proof. split=> k i; apply: eq_isumR => j _; exact: multC. Qed.

Lemma matrix_multx1 : forall m n (A : M_(m, n)), A *m \forall m = \forall A.
Proof. move=> 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=> m n p A1 A2 B; split=> i k /=; rewrite -isum_plus.
by apply: eq_isumR => 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 +m A *m B2.
Proof. move=> m n p A B1 B2; apply: matrix_transpose_inj.
rewrite matrix_transpose_plus !matrix_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) = \forall A *m B *m C.
Proof. move=> m n p q A B C; split=> i l /=.
transitivity \forall (k) \forall (l) ((A i j * B j k * C k l)).
    rewrite exchange_isum; apply: eq_isumR => j _; rewrite isum_distrL.
    by apply: eq_isumR => k _; rewrite multiA.
by apply: eq_isumR => j _; rewrite isum_distrR.
Qed.

Lemma perm_matrixM : forall n (s t : S_(n)),
     perm_matrix (s * t)%G = perm_matrix s *m perm_matrix t.
Proof. move=> n; split=> i j /=; rewrite (isumD1 (s i)) // set1 multx -permM.
rewrite isum0 => [ij']; first by rewrite plusc plus0x.
by rewrite andbT; move/\negbET->; rewrite mult0x.
Qed.

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

Lemma matrix_trace_scale : forall n x (A : M_(n)), \tr (x *m A) = x * \tr A.
Proof. by move=> x; rewrite isum_distrL. Qed.
Lemma determinant_multilinear : forall n (A B C : M_(n)) i0 b c,
    row i0 A = b * sm row i0 B =m c * sm row i0 C ->
    row' i0 B = m row' i0 A -> row' i0 C =m row' i0 A ->
    |\det A = b * \det B + c * \det C.
Proof.
move n A B C i0 b c ABC.
move/matrix_eq_rem_row CA; move/matrix_eq_rem_row CA.
rewrite isum_distrL isum_plus; apply eq_isumR => s _.
rewrite l!(multCA (_ ^ s)) -distr; congr (_ * _).
rewrite !(iprodD1 _ i0 (setA _)) (matrix_eq_row ABC) distrR lmultA.
by congr (_ * _ + _ * _); apply eq_iprodR => i;
rewite anbT => ?; rewrite ?BA ?CA.
Qed.

Lemma alternate_determinant : forall n (A : M_(n)) i1 i2,
    i1 !+ i2 -> A i1 -i1 A i2 -> |\det A = 0.
Proof.
move n A i1 i2 Di2 A12; pose r := I_(n(i).
pose t := transp i1 i2; pose tr s := (t * s)\times G.
move trK : involutive tr by move=> s; rewrite \tra multA transp2 mult1.
move Etr : forall s, odd_perm (tr s) = even_perm s.
    by move=> s; rewrite odd_perm odd_transp Di2.
rewrite (\det (\bra isumID (even_perm r) \bra) = set S1 := \bra isum (in _ _ _).
rewrite (\bra minusxx S1) _ cong (_ + _); rewrite (\bra S1 -isum opp).
rewrite (reindex_isum tr); last by exists tr.
symmetry; apply eq_isum => [s | s seven]; first by rewrite negbK Etr.
rewrite -multm multA tr seven (negbK seven) multmix; congr (_ * _).
rewrite (reindex_iprod tr); last by exists (t _ -> _) => i _; exact: transpK.
apply eq_iprodR => i _; rewrite permM /.
by case: transpP => // ->; rewrite A12.
Qed.

Lemma determinant_transpose : forall n (A : M_(n)), |\det (\^t A) = \det A.
Proof.
move n A; pose r := I_(n); pose ip p := permtype r := p^.-1.
rewrite (\det (\bra reindex_isum ip) = last first.
    by exists ip => s _; rewrite \tra invgK.
apply eq_isumR => s _; rewrite odd_perm /= (reindex_iprod s).
    by congr (_ * _); apply eq_iprodR => i _; rewrite permK.
by exists (s^-1 _ _ _ => i _; rewrite ?permK ?permK.
Qed.

Lemma determinant_perm : forall n s, |\det (\perm_matrix n s) = (-1) ^ s.
Proof.
move n s; rewrite (\det (\bra isumD1 s) =.
rewrite iprod1 => [i _]; last by rewrite /= set11.
rewrite isum0 => [!t Dst]; first by rewrite plusC plus0x multC multlx.
case (pickP (fun i => s t = t i)) => [! ist | Est].
    by rewrite (iprodD1 i) / multCA /= (negbK ist) mult0x.
move Dst; rewrite anbT; case eqP.
-205-- determinant.v 81% (12564,4) (Coq)
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 \times n * \det A.

Proof.
move=> n x A; rewrite isum_distr; 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 \( \sum f \) \( \sum s \) \( i \)^(-1) \( ^* \) \( \prod (i) A B f s i \).
rewrite exchange_isum; apply: eq_isumR => s _ .
by rewrite -isum_distr distr_iprodA_isumA.
rewrite \( \sum \) ID (fun f => uniq (fval f)) plusC isum0 ?plus0x => /\ [If UF].
rewrite (reindex_isum (fun s => val (pval s))) ; last first.
pose s0 : S_(n) := S0; pose uf (f : F_(n)) := uniq (fval f).
pose pf f := if isub uf f is some s then Perm s else \emptyset.
exists pf /= /= uf uf; rewrite /pf (isubT uf uf) //; exact: eq_fun_of_perm.
apply: eq_isum [\$S ]; rewrite ?(valP (pval s)) ; isum_distrL.
rewrite (reindex_isum (mulg s)) ; last first.
by exists (mulg s'-1) \( => \) t; rewrite ?mulkgv ?mulkg.
apply: eq_isumR \( \Rightarrow \) t; rewrite iprod_mult multCA multA multCA multA.
rewrite -sign_permM; congr (_ * _); rewrite (reindex_iprod s^1); last first.
by exists (s : _ _); rewrite ?permK ?permkv.
by apply: eq_iprodR => i _; rewrite permM permKv ?set1 // -[3][i](permKv s).
transitivity \( \det \) \( \matrix i j \) \( B (f i) (j) \) \( ^* \) \( \prod (i) A i (f i) \).
rewrite multCA iprod_distr; apply: eq_isumR => s _ .
by rewrite multCA iprod_mult.
suffices [il [if Ef12 Di12]] : exists il, exists s il, f il = f il &\& (il1 = il2).
by rewrite [alternate_determinant Di12] ?mul0x => /= j; rewrite Ef12.
pose ninj il1 il2 := f il1 = f il2 &\& (il1 \neq il2).
case: (pickP (fun i => == set0b (ninj i)) ) \( \Rightarrow \) [il1 ninj].
by case/set0Pm=> il2; case/andP; move/eP; exists il1; exists il2.
case/(perm_uniqP f) : uf \( \Rightarrow \) il1 il2; move/eP=> Df12; apply/eP.
by apply/idPm=> Di12; case/set0Pm: (ninj f il1) ; exists il2; apply/andP.
Qed.

(* And now, the Laplace formula. *)

Definition cofactor n (A : M_(n)) (i j : I_(n)) :=
(-1)^j (val i + val j) * \det (row', i (col', j) A).

(* Same bug as determinant Add Morphism cofactor with *)

-(DOS) determinant.v 85% (1284,0) (coa)
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_\(_n\))(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_\(_n\))(n), \det (A *m B) = \det A * \det B.
Proof.
move=> n A B; rewrite isum_distrR.
pose AB (f : F_\(_n\))(n) (s : S_\(_n\))(n) i := A i (f i) * B (f i) (s i).
transitivity (\sum_(f) \sum_(s : S_\(_n\))(n) (-1) ^ s * \prod_(i) AB f s i).
rewrite 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].
rewrite (reindex_isum (fun s => val (pval s))) last first.
have s0 : S_\(_n\)(n) := 1%G; pose uf (f : F_\(_n\))(n) := uniq (fval f).
pose pf := 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.
rewrite (reindex_isum (mulg s)); last first.
by exists (mulg s^\(-1\)) => t; rewrite ?mulKgv ?mulKg.
Secure Distributed Computations and their Proofs

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
- CertiCrypt: Formal Proofs for Computational Cryptography
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) ==
\forall d \in \text{Nat}
\forall n \in \text{Nat}
\forall \exists q \in \text{Nat} : n = d \times q

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

\text{Prime}(x) ==
\forall x \in \text{Nat}
\forall \forall d \in \text{Nat} : \text{Divides}(d, x) \Rightarrow \forall d = 1
\forall d = x

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

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

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

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

PROOF <1>1. ASSUME
    \begin{align*}
    & \text{NEW } x \in \text{Nat}, \\
    & \text{NEW } y \in \text{Nat}, \\
    & \text{coprimality} :: \text{Coprime}(x, y), \\
    & \text{main} :: x^2 = 2 * y^2
    \end{align*}

    PROVE
    \text{FALSE}

    PROOF <2>1. \text{Divides}(2, x)
        PROOF <3>1. \text{Divides}(2, x^2)
            \begin{align*}
            & \text{BY } <1> \text{1} \text{3} \\
            \end{align*}
        \text{QED}
        \begin{align*}
        & \text{BY } <3> \text{1}, \text{TwoIsPrime}, \text{SquareLemma} \\
        \end{align*}

    <2>2. \text{Divides}(2, y)
        PROOF <3>1. \text{PICK } r \in \text{Nat} : x = 2 * r
            \begin{align*}
            & \text{BY } <2> \text{1}, \text{DivLemma} \\
            \end{align*}
        \begin{align*}
        & <3>2. x^2 = 2 * (2 * r^2) \\
        \end{align*}
        \text{BY } <3> \text{1}
        \begin{align*}
        & <3>3. 2 * y^2 = 2 * (2 * r^2) \\
        \end{align*}
        \text{BY } <1> \text{1} \text{main}, <3> \text{2}
        \begin{align*}
        & <3>4. y^2 = 2 * r^2
        \end{align*}
<3-2. QED
   BY <3-1, TwoIsPrime, SquareLemma
<2-2. Divides(2, y)
   PROOF <3-1. PICK r \in Nat : x = 2 * r
      BY <2-1, DivLemma
      <3-2. x^2 = 2 * (2 * r^2)
         BY <3-1
      <3-3. 2 * y^2 = 2 * (2 * r^2)
         BY <1-1!main, <3-2
      <3-4. y^2 = 2 * r^2
         BY <3-2, LeftCancellationLemma
      <3-5. QED
         BY <3-3, TwoIsPrime, SquareLemma
<2-3. ~ (Divides(2, y))
   PROOF <3-1. \(\forall d \in \text{Nat} : (\text{Divides}(d, x) \land \text{Divides}(d, y)) \Rightarrow d = 1\)
      BY <1-1!coprimality
      <3-2. 2 = 1
         BY <2-1, <2-2, <3-1
      <3-3. QED
         BY <3-2, TwoIsNotOne
<2-4. QED
   BY <2-2, <2-3, NegElim
<1-2. QED
   BY <1-1, ImplIntro, ForallIntro
# Logics in track A

<table>
<thead>
<tr>
<th>Category</th>
<th>System</th>
<th>Logic Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math. components</td>
<td>Coq</td>
<td>higher-order + reflection</td>
</tr>
<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>
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