xAct

Efficient manipulation of tensor expressions

José M. Martín García

Laboratoire Univers et Théories, Meudon
&
Institut d'Astrophysique de Paris

LUTH, Meudon, 21 April 2009

Computers in science

Computers in science

- Human 10^{-2} flops vs. computer $10^{9} - 10^{15}$ flops.

Computers in science

- Human 10^{-2} flops vs. computer $10^{9} - 10^{15}$ flops.
- Numerics: Approximate solutions to continuous problems.

Computers in science

- Human 10^{-2} flops vs. computer $10^{9} - 10^{15}$ flops.
- Numerics: Approximate solutions to continuous problems.
- Computers are discrete-calculus machines!

Computers in science

- Human 10^{-2} flops vs. computer $10^9 - 10^{15}$ flops.
- Numerics: Approximate solutions to continuous problems.
- Computers are discrete-calculus machines!
- Computer algebra (CA): Exact solutions.

Computers in science

- Human 10^{-2} flops vs. computer $10^9 - 10^{15}$ flops.
- Numerics: Approximate solutions to continuous problems.
- Computers are discrete-calculus machines!
- Computer algebra (CA): Exact solutions.
- Our problem: Efficient Tensor Computer Algebra (TCA).

Summary

1. General purpose computer algebra (CA)
   - Focus: the canonicalizer.
2. Computer algebra for tensor calculus (TCA)
   - Focus: types of problems.
3. A *Mathematica* / C implementation: *xAct*
4. Case example: scalars of the Riemann tensor

1. Computer algebra (CA)

1. Computer algebra (CA)

- Definition:
  - Manipulation of symbols (even the program itself)
  - No truncation of information
  - In science: manipulation of symbolic mathematical expressions

1. Computer algebra (CA)

- Definition:
  - Manipulation of symbols (even the program itself)
  - No truncation of information
  - In science: manipulation of symbolic mathematical expressions

- Early history:
  - 1950: ALGAE (Los Alamos)
  - 1953: Kahrimanian, Nolan: differentiation systems
  - 1963: ALTRAN / ALPAK (Bell Labs)
  - 1960’s: LISP: recursion, conditionals, dynamical allocation of memory, garbage collection, etc.

1b. Computer algebra systems (CAS)

1b. Computer algebra systems (CAS)

- General purpose systems:

System	Language	Develop.	Release	Strengths

1b. Computer algebra systems (CAS)

- General purpose systems:

System	Language	Develop.	Release	Strengths
Reduce	LISP	?	1968	
Macsyma	LISP	?	1970	
Derive	LISP	1979	1988	

1b. Computer algebra systems (CAS)

- General purpose systems:

System	Language	Develop.	Release	Strengths
Reduce	LISP	?	1968	
Macsyma	LISP	?	1970	
Derive	LISP	1979	1988	
Maple	C	1979	1985	hashing routines
Mathematica	C	1986	1988	pattern matching

1b. Computer algebra systems (CAS)

- General purpose systems:

System	Language	Develop.	Release	Strengths
Reduce	LISP	?	1968	
Macsyma	LISP	?	1970	
Derive	LISP	1979	1988	
Maple	C	1979	1985	hashing routines
Mathematica	C	1986	1988	pattern matching
Maxima	CLISP	1982	1998	
Axiom	Aldor	1971-91	2002	object-oriented

1b. Computer algebra systems (CAS)

- General purpose systems:

System	Language	Develop.	Release	Strengths
Reduce	LISP	?	1968	
Macsyma	LISP	?	1970	
Derive	LISP	1979	1988	
Maple	C	1979	1985	hashing routines
Mathematica	C	1986	1988	pattern matching
Maxima	CLISP	1982	1998	
Axiom	Aldor	1971-91	2002	object-oriented

- Specialised systems for:
  - Celestial mechanics
  - Group theory: MAGMA, GAP
  - General Relativity
  - Quantum Field Theory
  - Fluid mechanics
  - Industrial applications

1c. CA. Memory limitations

Recursive algorithm $x_{i+1} = f(x_i)$ with initial seed x_0:

1c. CA. Memory limitations

Recursive algorithm $x_{i+1} = f(x_i)$ with initial seed x_0:

- Numerical: reserve fixed memory per number and

$$x_{i+1} = \text{Truncate}[f(x_i)]$$

- CA: exact algorithm, reallocate memory.
  $\Rightarrow$ Generic memory growth $\Rightarrow$ Generic computing-time growth

1c. CA. Memory limitations

Recursive algorithm $x_{i+1} = f(x_i)$ with initial seed x_0:

- Numerical: reserve fixed memory per number and

$$x_{i+1} = \mathrm{Truncate}[f(x_i)]$$

- CA: exact algorithm, reallocate memory.
  $\Rightarrow$ Generic memory growth $\Rightarrow$ Generic computing-time growth

Examples:

- $\det(A+B) \quad \longrightarrow \quad n!\, 2^n$ terms $\;(\simeq 4 \cdot 10^9$ for $n = 10\,)$

1c. CA. Memory limitations

Recursive algorithm $x_{i+1} = f(x_i)$ with initial seed x_0:

- Numerical: reserve fixed memory per number and

$$x_{i+1} = \mathrm{Truncate}[f(x_i)]$$

- CA: exact algorithm, reallocate memory.
  $\Rightarrow$ Generic memory growth $\Rightarrow$ Generic computing-time growth

Examples:

- *Intermediate expression swell*:
  Simple input $\longrightarrow$ Complex intermediate steps $\longrightarrow$ Simple output

1c. CA. Memory limitations

Recursive algorithm $x_{i+1} = f(x_i)$ with initial seed x_0:

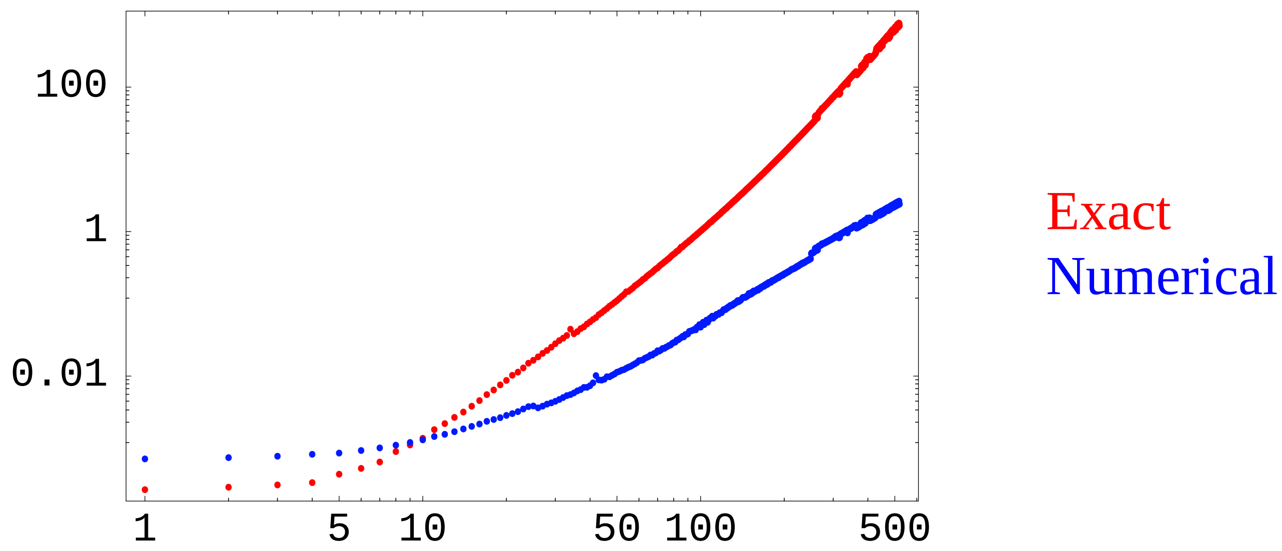
- Numerical: reserve fixed memory per number and

$$x_{i+1} = \text{Truncate}[f(x_i)]$$

- CA: exact algorithm, reallocate memory.
  $\Rightarrow$ Generic memory growth $\Rightarrow$ Generic computing-time growth

Examples:

- Linear systems with random integers $|c_i| \leq 100$. Timing (s):



1d. CA. Intrinsic limitations

Richardson’s theorem (1968):

1d. CA. Intrinsic limitations

Richardson's theorem (1968):

Let be E a set of real functions such that

- If $A(x), B(x) \in E$ then $A(x) \pm B(x), A(x)B(x), A(B(x)) \in E$.
- The rational numbers are contained as constant functions.

1d. CA. Intrinsic limitations

Richardson's theorem (1968):

Let be E a set of real functions such that

- If $A(x), B(x) \in E$ then $A(x) \pm B(x), A(x)B(x), A(B(x)) \in E$.
- The rational numbers are contained as constant functions.

Then for expressions $A(x)$ in E,

- if $\log 2, \pi, e^x, \sin x \in E$ then $A(x) \geq 0$ for all x is unsolvable;

1d. CA. Intrinsic limitations

Richardson's theorem (1968):

Let be E a set of real functions such that

- If $A(x), B(x) \in E$ then $A(x) \pm B(x), A(x)B(x), A(B(x)) \in E$.
- The rational numbers are contained as constant functions.

Then for expressions $A(x)$ in E,

- if $\log 2, \pi, e^x, \sin x \in E$ then $A(x) \geq 0$ for all x is unsolvable;
- if also $\sqrt{x^2} \in E$ then $A(x) \equiv 0$ is unsolvable.

1d. CA. Intrinsic limitations

Richardson's theorem (1968):

Let be E a set of real functions such that

- If $A(x), B(x) \in E$ then $A(x) \pm B(x), A(x)B(x), A(B(x)) \in E$.
- The rational numbers are contained as constant functions.

Then for expressions $A(x)$ in E,

- if $\log 2, \pi, e^x, \sin x \in E$ then $A(x) \geq 0$ for all x is unsolvable;
- if also $\sqrt{x^2} \in E$ then $A(x) \equiv 0$ is unsolvable.
- If furthermore there is a function $B(x) \in E$ without primitive in E then the integration problem is unsolvable. Example: e^{x^2}.

1d. CA. Intrinsic limitations

Richardson's theorem (1968):

Let be E a set of real functions such that

- If $A(x), B(x) \in E$ then $A(x) \pm B(x), A(x)B(x), A(B(x)) \in E$.
- The rational numbers are contained as constant functions.

Then for expressions $A(x)$ in E,

- if $\log 2, \pi, e^x, \sin x \in E$ then $A(x) \geq 0$ for all x is unsolvable;
- if also $\sqrt{x^2} \in E$ then $A(x) \equiv 0$ is unsolvable.
- If furthermore there is a function $B(x) \in E$ without primitive in E then the integration problem is unsolvable. Example: e^{x^2}.

Is Computer Algebra doomed to failure?

1e. CA. The canonicalizer

1e. CA. The canonicalizer

There are families of expressions for which a canonical form can always be defined. Example: polynomials.

1e. CA. The canonicalizer

There are families of expressions for which a canonical form can always be defined. Example: polynomials.

- Definition: $A = B$ iff their canonical forms coincide.
- There are different canonical forms for a family of expressions.

1e. CA. The canonicalizer

There are families of expressions for which a canonical form can always be defined. Example: polynomials.

- Definition: $A = B$ iff their canonical forms coincide.
- There are different canonical forms for a family of expressions.

The algorithm in charge of canonicalization is called *the canonicalizer*.

1e. CA. The canonicalizer

There are families of expressions for which a canonical form can always be defined. Example: polynomials.

- Definition: $A = B$ iff their canonical forms coincide.
- There are different canonical forms for a family of expressions.

The algorithm in charge of canonicalization is called *the canonicalizer*.

Do not confuse

- canonicalization: objective, given a canonical form
- simplification: largely subjective. More difficult. *Mathematica*:

1e. CA. The canonicalizer

There are families of expressions for which a canonical form can always be defined. Example: polynomials.

- Definition: $A = B$ iff their canonical forms coincide.
- There are different canonical forms for a family of expressions.

The algorithm in charge of canonicalization is called *the canonicalizer*.

Do not confuse

- canonicalization: objective, given a canonical form
- simplification: largely subjective. More difficult. *Mathematica*:

$$\texttt{Simplify}[\,(x-1)(x+1)\,] \quad \longrightarrow \quad -1+x^2$$
$$\texttt{Simplify}[\,(x-y)(x+y)\,] \quad \longrightarrow \quad (x-y)(x+y)$$
$$\texttt{Simplify}[\,x^4-x\,] \quad \longrightarrow \quad x(-1+x^3)$$

2. Computer algebra for (GR) tensors

Motivation: Long but “simple” problems

- Perturbation theory
- Bel-Robinson, super energy-momentum tensors, ...

$$4B_{abcd} = C_a{}^e{}_b{}^f C_{cedf} + {}^*C_a{}^e{}_b{}^{f\,*}C_{cedf} \quad \Rightarrow \quad B_{abcd} = B_{(abcd)}$$

- Riemann polynomials:

$$R_{abcd}R^a{}_e{}^c{}_f R^{bfde} = R_{abcd}R^a{}_e{}^c{}_f R^{bedf} - \frac{1}{4}R_{abcd}R^{ab}{}_{ef}R^{cdef}$$

- Lovelock (dimension-dependent) identities
- Component expansions in numerics (code generation)
  `[ Kranc:  Husa, Hinder, Lechner ’04 ]`
- Manipulation and classification of metrics
  `[ GRDB: Ishak, Lake ’02; ICD: Skea ’97]`
- Added benefits: reproducibility, error-free, ...

2b. TCA. Early history

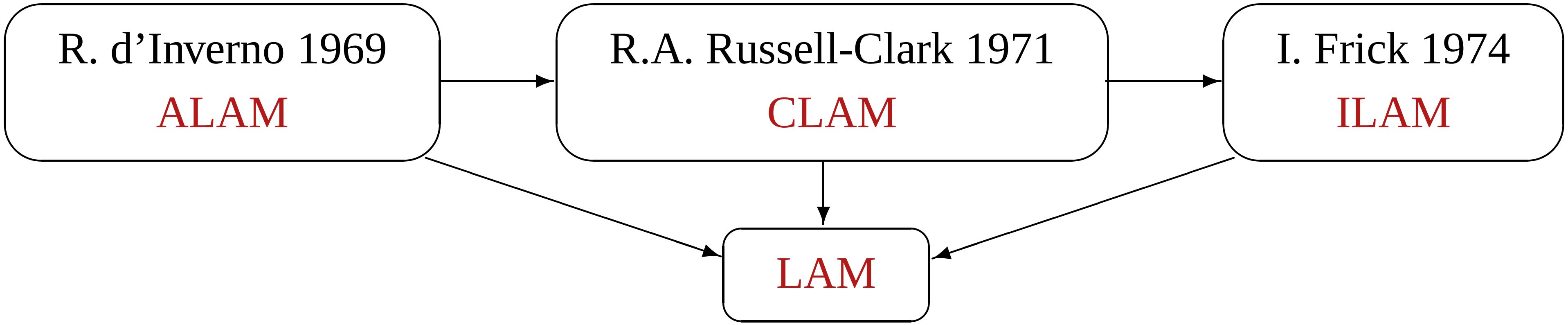
2b. TCA. Early history

R. d'Inverno 1969

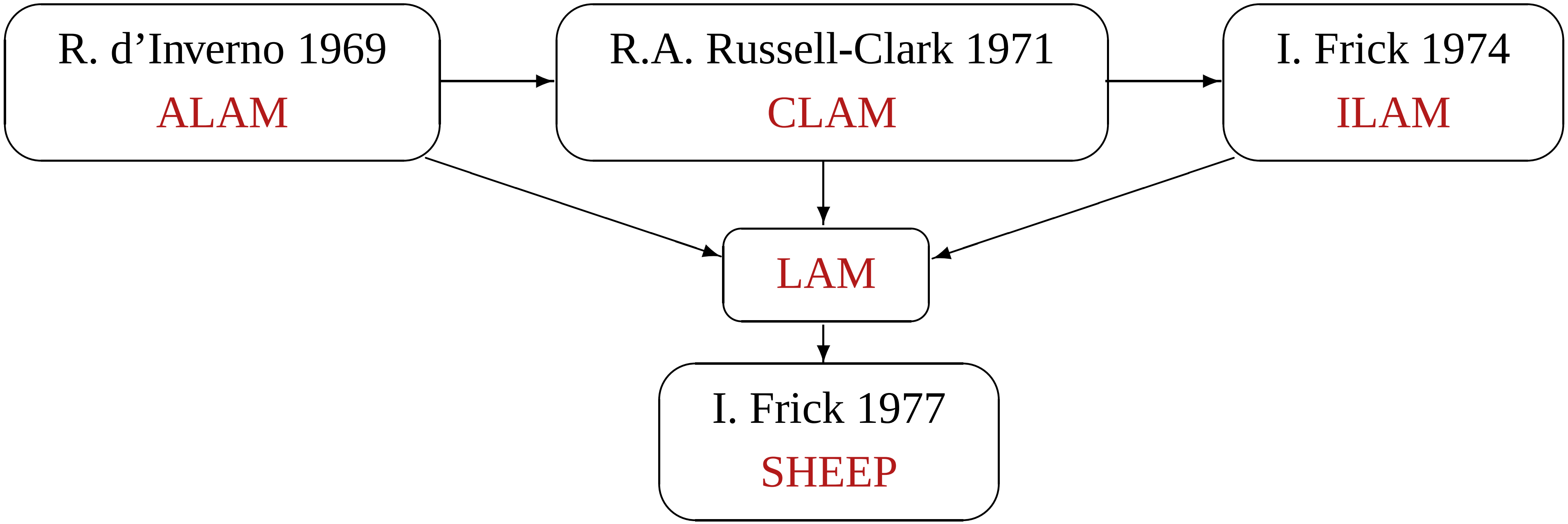
ALAM

2b. TCA. Early history

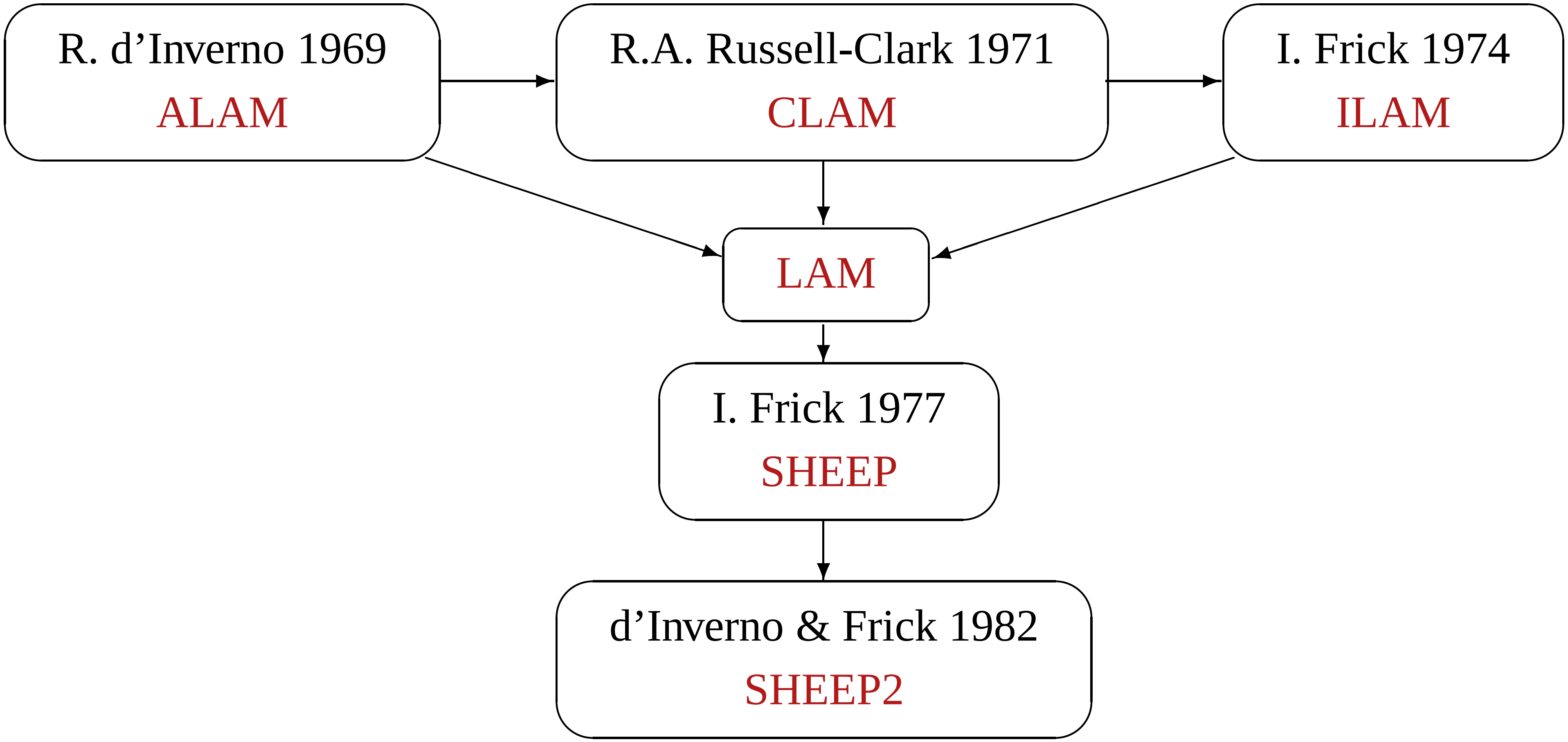
2b. TCA. Early history



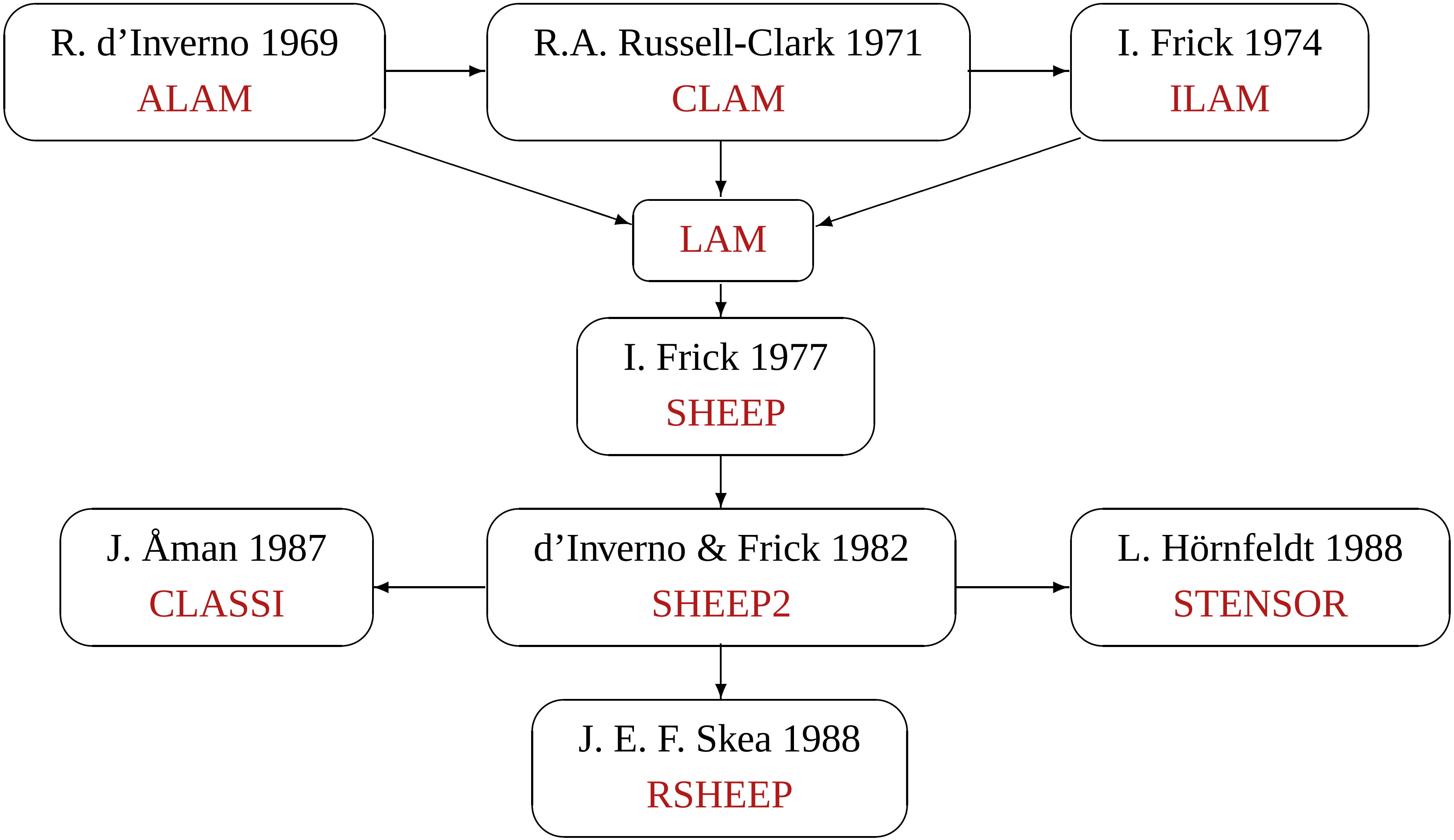
2b. TCA. Early history



2b. TCA. Early history



2b. TCA. Early history



2c. Classes: types of problems

2c. Classes: types of problems

Component computations:

- Give a metric in a coordinate system / frame.
- Compute other tensors from that metric.
- Key issues:
  - Component expansions. Many terms. Memory? `[ Lake '03 ]`
  - Symmetries. Independent components? `[ Klioner '04 ]`

2c. Classes: types of problems

Component computations:

- Give a metric in a coordinate system / frame.
- Compute other tensors from that metric.
- Key issues:
  - Component expansions. Many terms. Memory? [Lake ’03]
  - Symmetries. Independent components? [Klioner ’04]

Abstract computations:

- Define tensors with symmetries, connections, etc.
- Compute and simplify expressions.
- Key issue: canonicalization with symmetries and dummy indices.

2c. Classes: types of problems

Component computations:

- Give a metric in a coordinate system / frame.
- Compute other tensors from that metric.
- Key issues:
  - Component expansions. Many terms. Memory? [Lake '03]
  - Symmetries. Independent components? [Klioner '04]

Abstract computations:

- Define tensors with symmetries, connections, etc.
- Compute and simplify expressions.
- Key issue: canonicalization with symmetries and dummy indices.

Question: Component computations from abstract computations?

2c-2. Classes: sources of complexity

- General expression: tensor polynomial

$$\ldots + 3r^2 R_{abcd} R^{aecf} T^b{}_e \nabla_f \, v^d + R^{abcd} R_{cdef} R^{ef}{}_{ab} + \ldots$$

2c-2. Classes: sources of complexity

- General expression: tensor polynomial

$$\ldots + 3r^2 R_{abcd} R^{aecf} T^b{}_e \nabla_f \, v^d + R^{abcd} R_{cdef} R^{ef}{}_{ab} + \ldots$$

- Expand and canonicalize terms independently (parallelism).

2c-2. Classes: sources of complexity

- General expression: tensor polynomial

$$\ldots + 3r^2 R_{abcd} R^{aecf} T^b{}_e \nabla_f v^d + R^{abcd} R_{cdef} R^{ef}{}_{ab} + \ldots$$

- Expand and canonicalize terms independently (parallelism).
- Tensor product: sort tensors and construct equivalent single tensor with inherited symmetries `[ Portugal '99 ]`.

2c-2. Classes: sources of complexity

- General expression: tensor polynomial

$$\ldots + 3r^2 R_{abcd} R^{aecf} T^b{}_e \nabla_f v^d + R^{abcd} R_{cdef} R^{ef}{}_{ab} + \ldots$$

- Expand and canonicalize terms independently (parallelism).
- Tensor product: sort tensors and construct equivalent single tensor with inherited symmetries `[ Portugal '99 ]`.
- Canonicalize tensor with n indices and arbitrary symmetry:
  - "Simple" algorithms: timings are exponential in n.
  - Efficient algorithms: timings are effectively polynomial in n.

2c-2. Classes: sources of complexity

- General expression: tensor polynomial

$$\ldots + 3r^2 R_{abcd} R^{aecf} T^b{}_e \nabla_f v^d + R^{abcd} R_{cdef} R^{ef}{}_{ab} + \ldots$$

- Expand and canonicalize terms independently (parallelism).
- Tensor product: sort tensors and construct equivalent single tensor with inherited symmetries `[ Portugal '99 ]`.
- Canonicalize tensor with n indices and arbitrary symmetry:
  - "Simple" algorithms: timings are exponential in n.
  - Efficient algorithms: timings are effectively polynomial in n.
- Arrange dummy indices in full expression.

2c-3. Classes: types of symmetries

2c-3. Classes: types of symmetries

Monoterm symmetries (perm groups):

$$R_{bacd} = -R_{abcd}, \qquad R_{cdab} = +R_{abcd}$$

- Most packages use ad hoc exponential algorithms.
- Polynomic algorithms to manipulate the Symmetric Group S_n, based on Strong Generating Set representations
  `[ Schreier, Sims, Knuth, ... 70's, 80's ].`
- Applied to tensor/spinor canonicalization: `[ Portugal et al. '01 ]`

2c-3. Classes: types of symmetries

Monoterm symmetries (perm groups):

$$R_{bacd} = -R_{abcd}, \qquad R_{cdab} = +R_{abcd}$$

- Most packages use ad hoc exponential algorithms.
- Polynomic algorithms to manipulate the Symmetric Group S_n, based on Strong Generating Set representations
  `[ Schreier, Sims, Knuth, ... 70's, 80's ].`
- Applied to tensor/spinor canonicalization: `[ Portugal et al. '01 ]`

Multiterm symmetries (perm algebras):

$$R_{abcd} + R_{acdb} + R_{adbc} = 0$$

- No known efficient algorithms. Solutions?
- Most elegant: Young tableaux `[ Fulling et al. '92; Peeters '05 ]`
- Particular case: dimension-dependent identities `[ Edgar et al. '02 ].`

2d. Tensor packages

MAXIMA: itensor, stensor / ctensor

REDUCE: atensor, RicciR, ExCalc / GRG, GRLIB, RedTen

MAPLE: Riegeom, Canon, MapleTensor / Riemann, Atlas, GRTensorII

MATHEMATICA: MathTensor ($$), dhPark, Tensors in Physics ($), Tensorial ($), Ricci, TTC, EinS, xTensor / GRTensorM, xCoba

Standalone: cadabra

Prototyping: Kranc, RNPL, TeLa

Many other small packages for component computations.

2e. Example 1

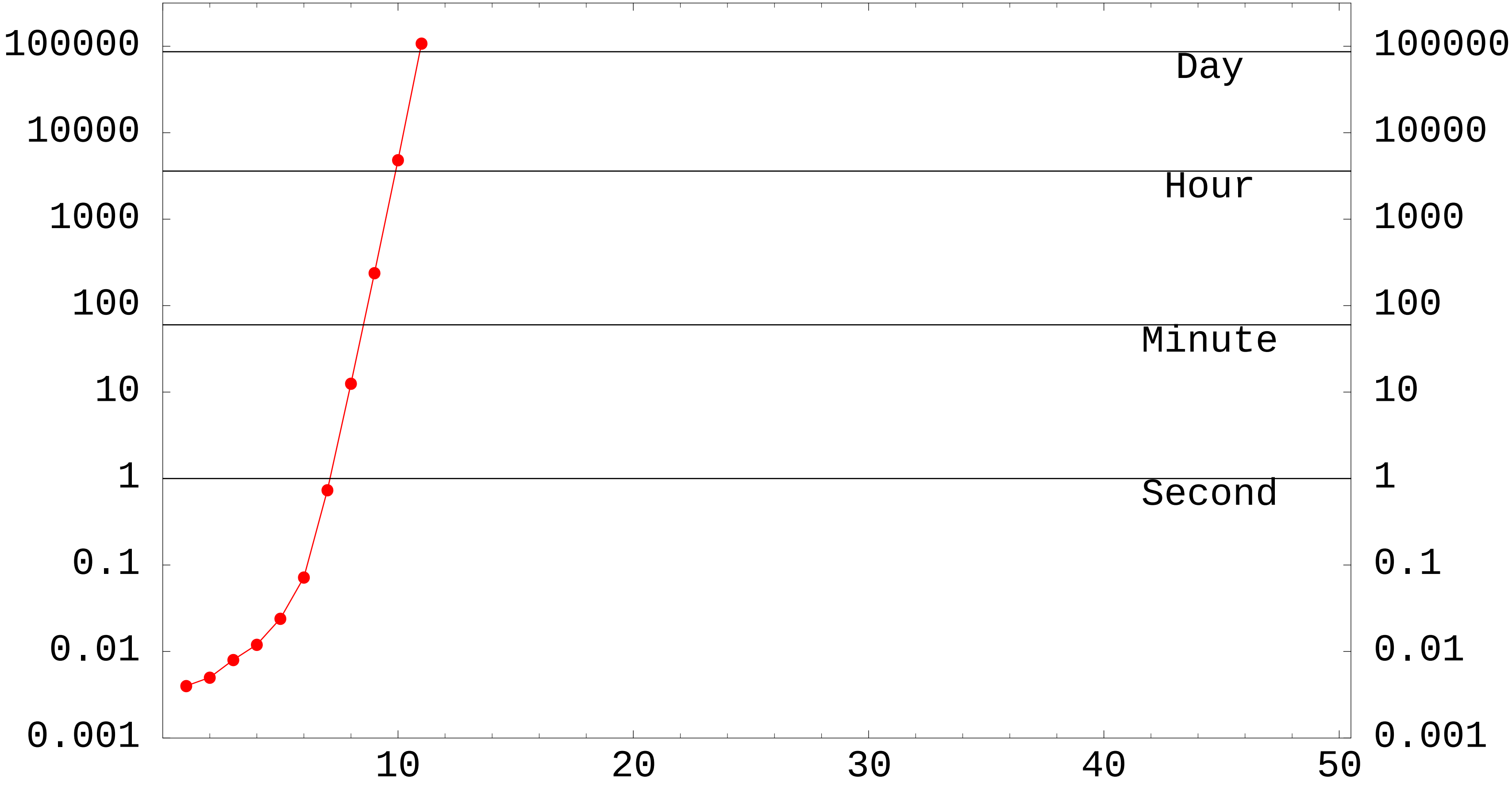
- Antisymmetric tensor $F_{ba} = -F_{ab}$.
- $F^{ab} F_{bc} \overset{n}{\cdots} F^h{}_a = 0$ if number n of tensors is odd.

2e. Example 1

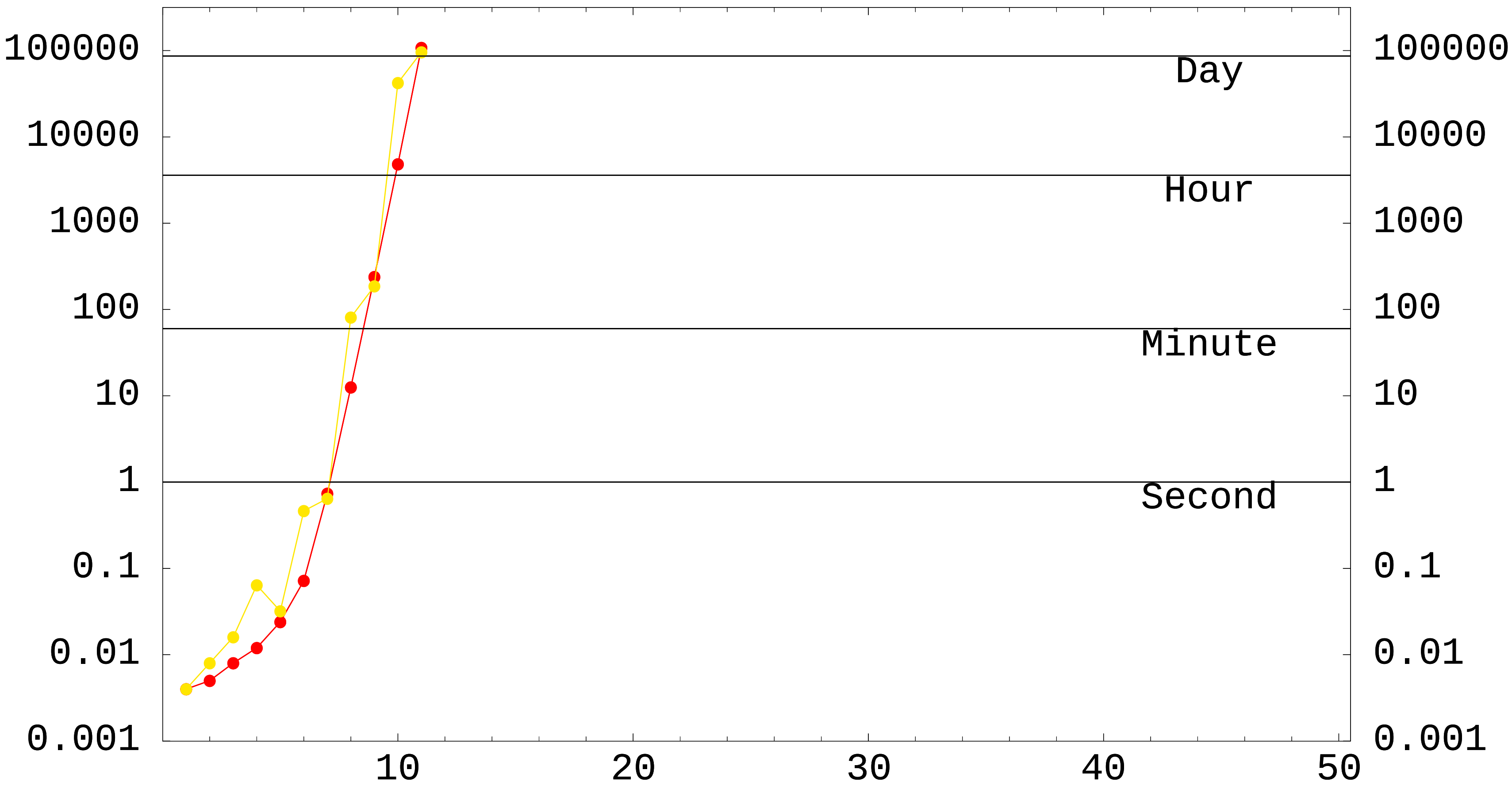
- Antisymmetric tensor $F_{ba} = -F_{ab}$.
- $F^{ab} F_{bc} \overset{n}{\cdots} F^{h}{}_{a} = 0$ if number n of tensors is odd.
- Timings (in seconds)

n	|Perm group|	MathTensor	xTensor
1	2	0	0
3	48	0.01	0.01
5	3840	0.02	0.03
7	645120	1.12	0.05
9	185794560	350	0.07
11	$8.2\ 10^{10}$	107745	0.09
19	$6.4\ 10^{22}$	?	0.28
29	$4.7\ 10^{39}$	?	0.94
39	$1.1\ 10^{58}$	?	2.7
49	$3.4\ 10^{77}$	?	6.5
59	$8.0\ 10^{97}$	?	13.7

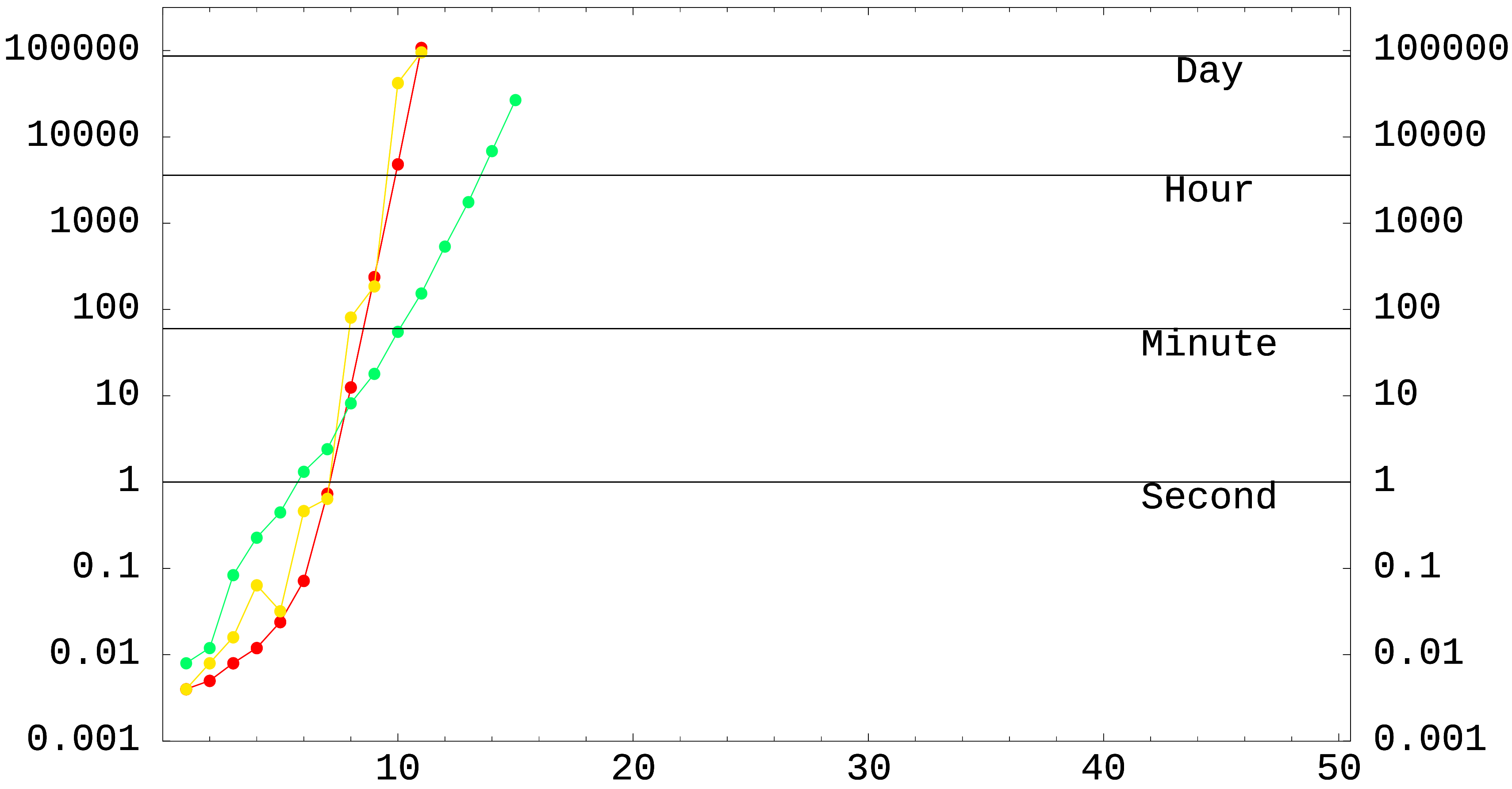
2e-2. Example 1



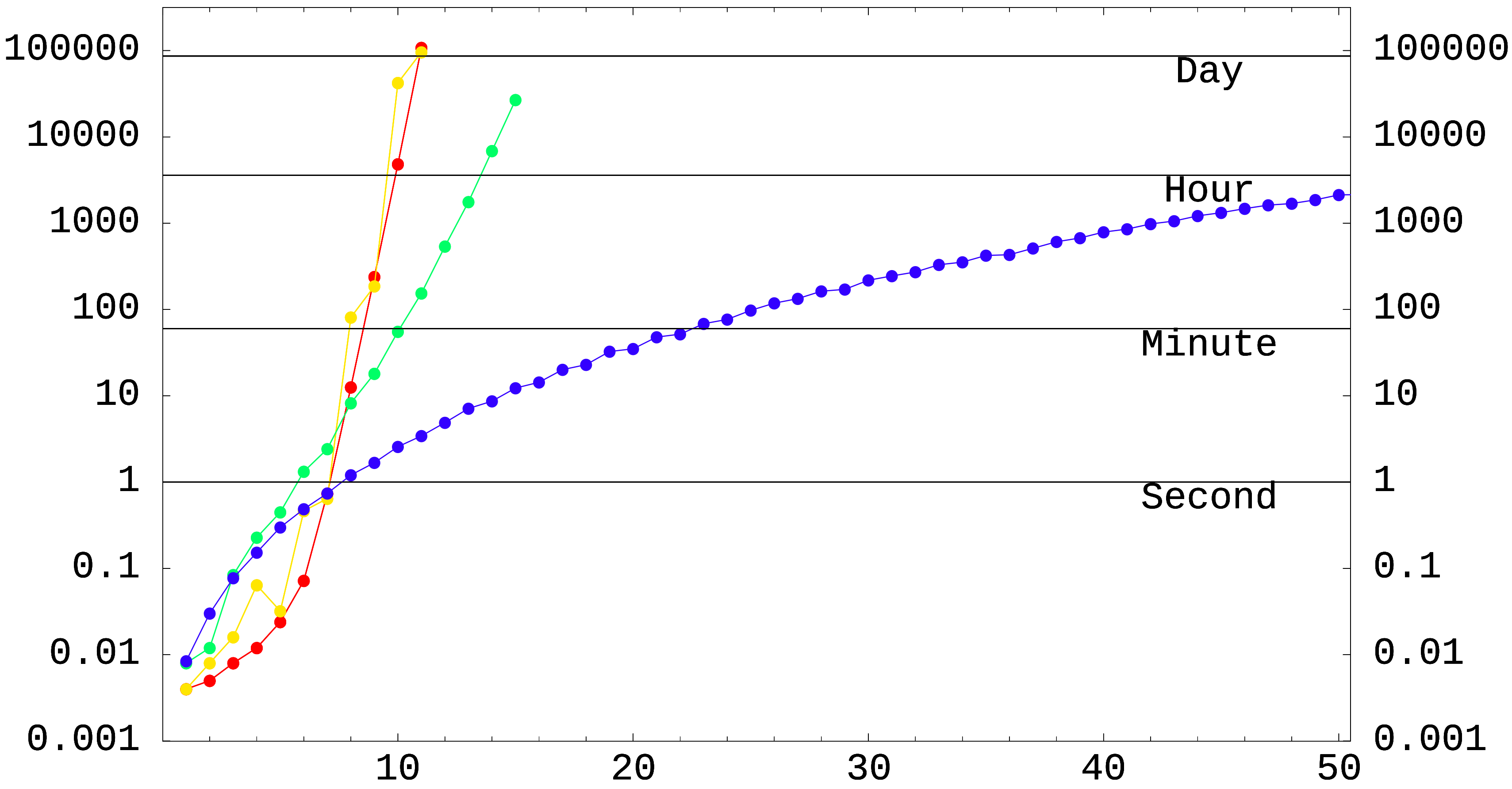
2e-2. Example 1



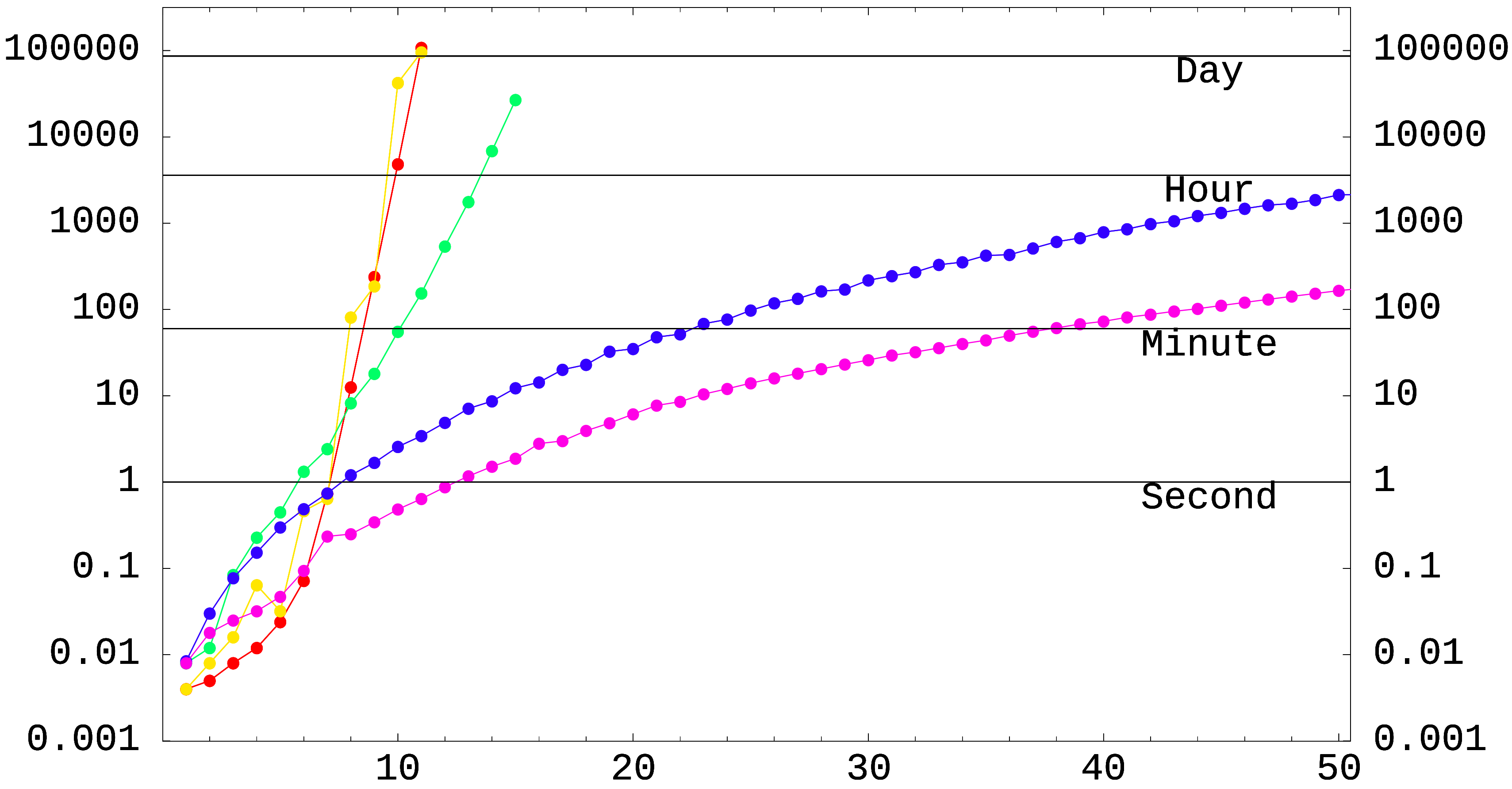
2e-2. Example 1



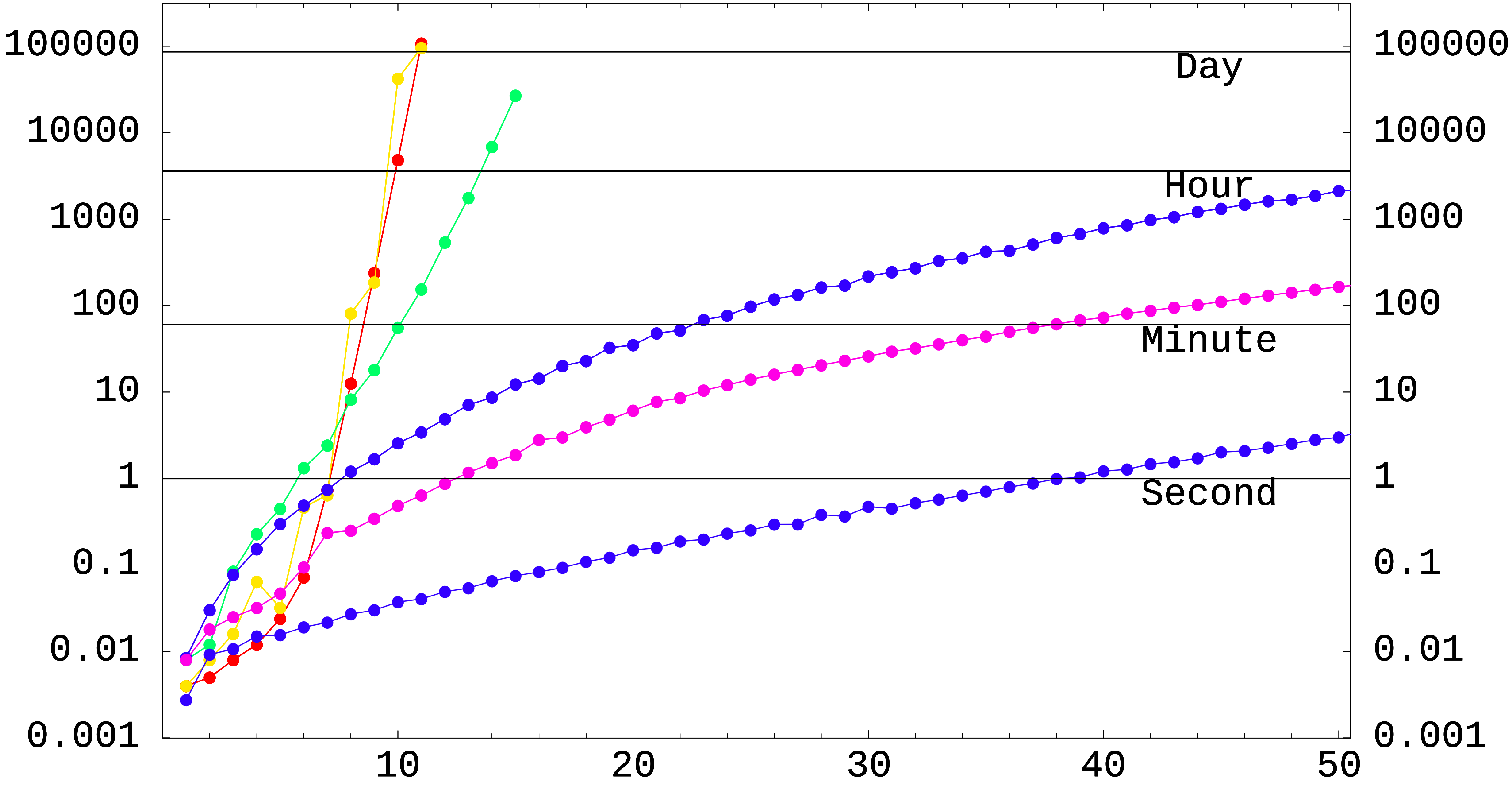
2e-2. Example 1



2e-2. Example 1



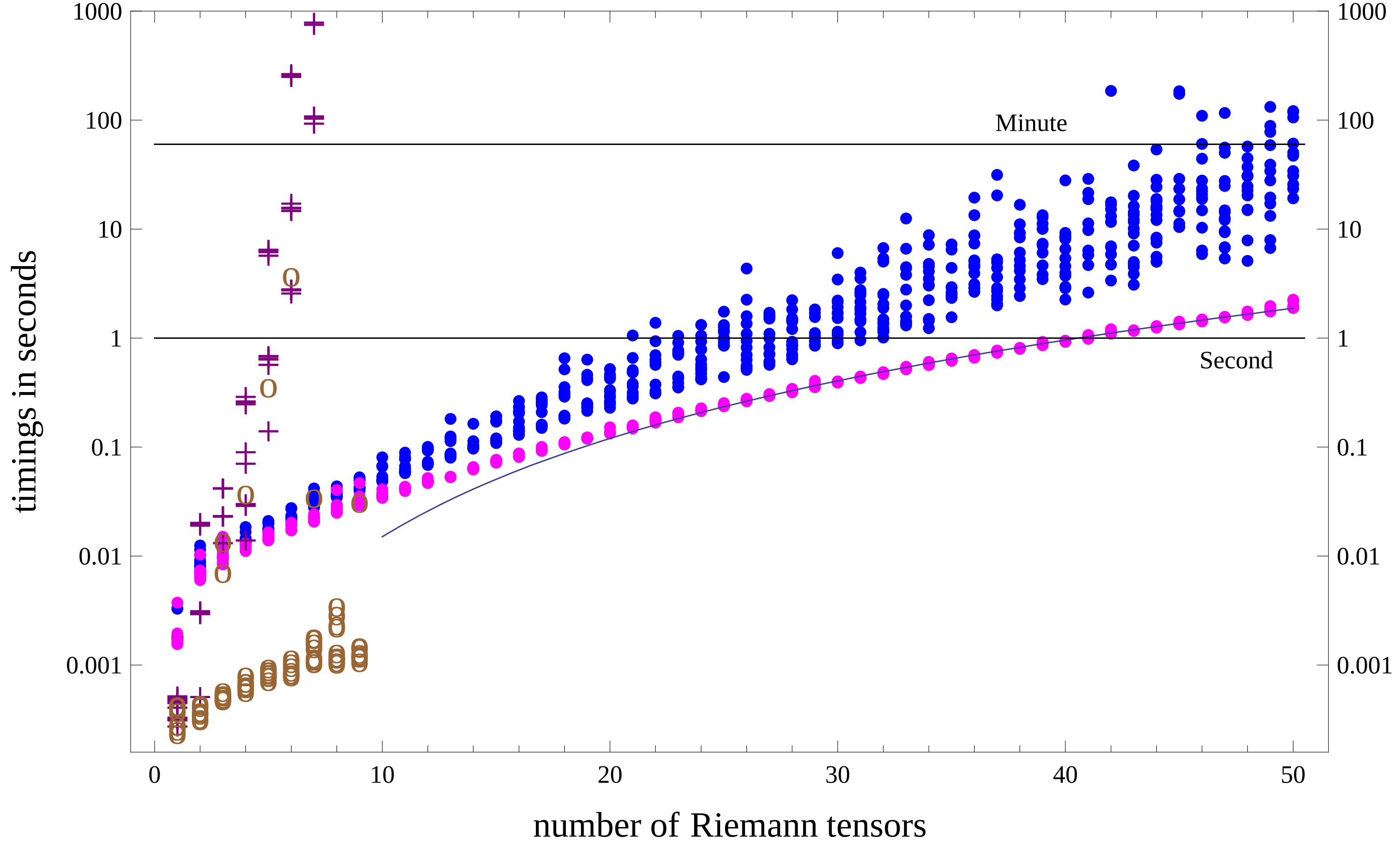
2e-2. Example 1



2e-3. Example 2

Random monomial Riemann scalars:

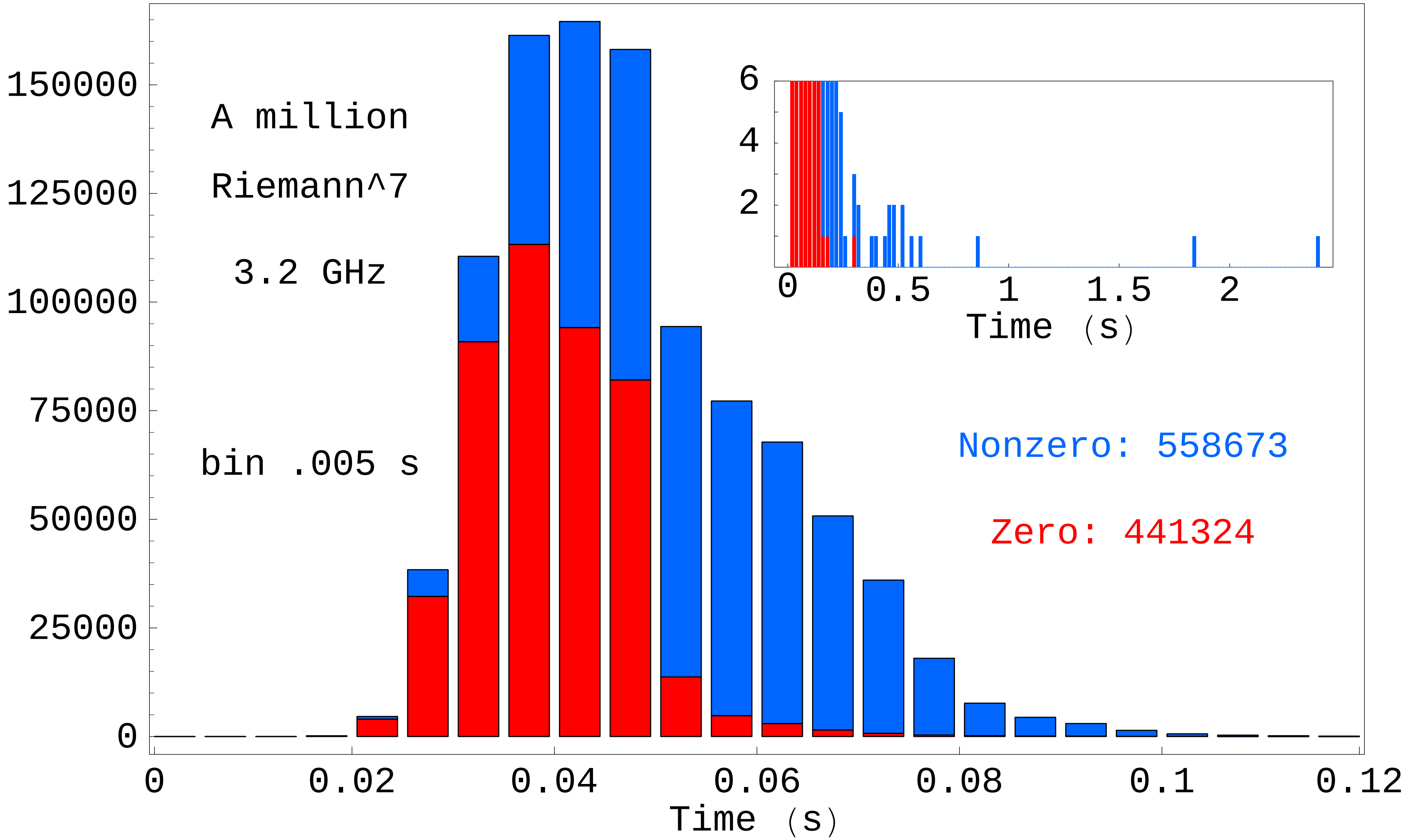
$$g^{a_1 b_7} \dots g^{d_n c_5} R_{a_1 b_1 c_1 d_1} \dots R_{a_n b_n c_n d_n}$$



The algorithm uses the intersection algorithm, which is known to be exponential in the worst case.

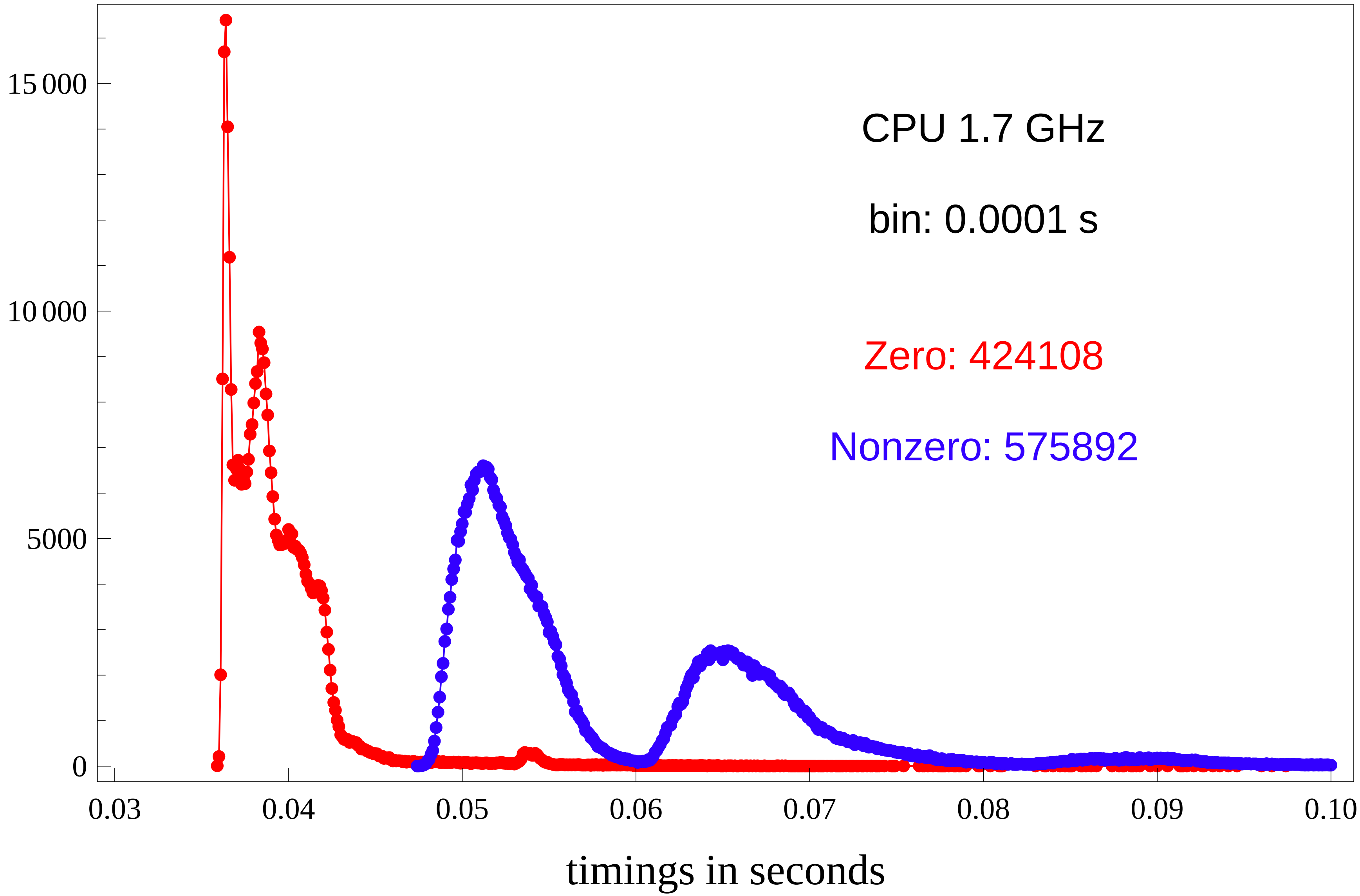
2e-4. Example 2

A million random monomial Riemann^7 scalars:



2e-5. Example 2

A million random monomial Riemann^{10} scalars:



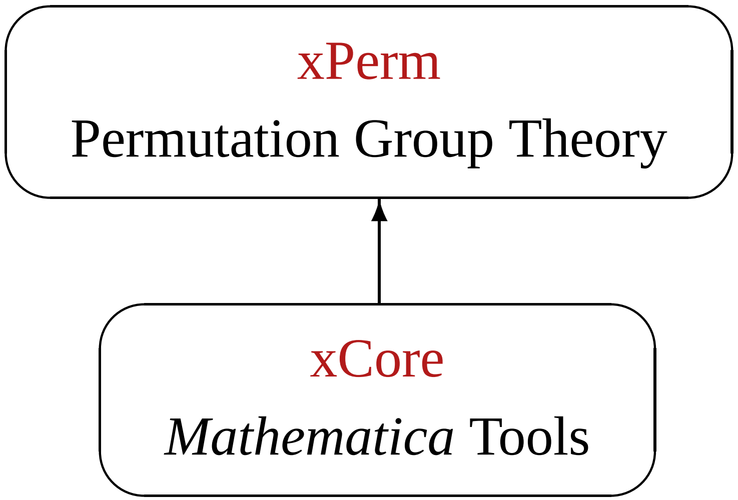
3. The xAct project

3. The *xAct* project

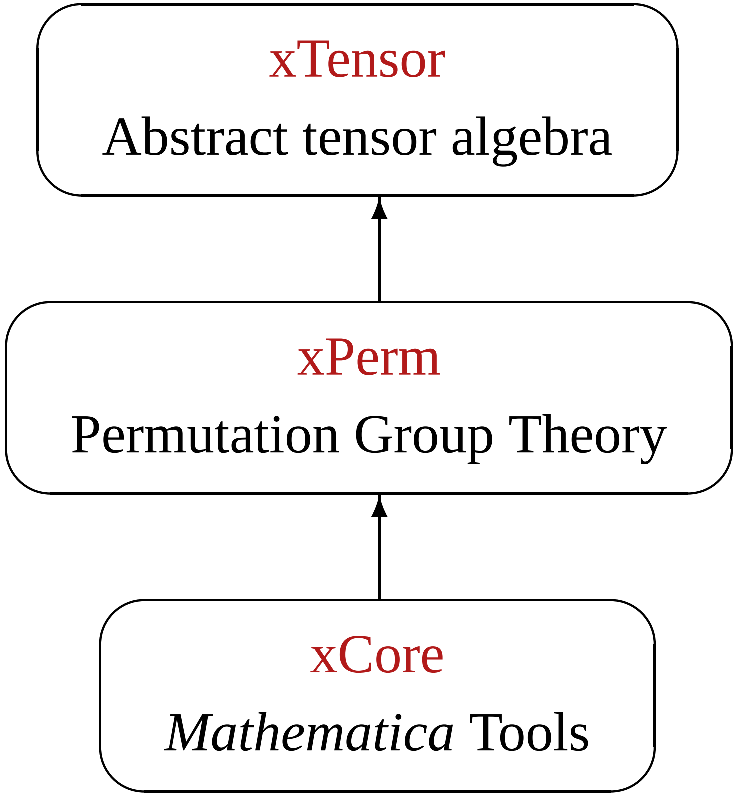
xCore

Mathematica Tools

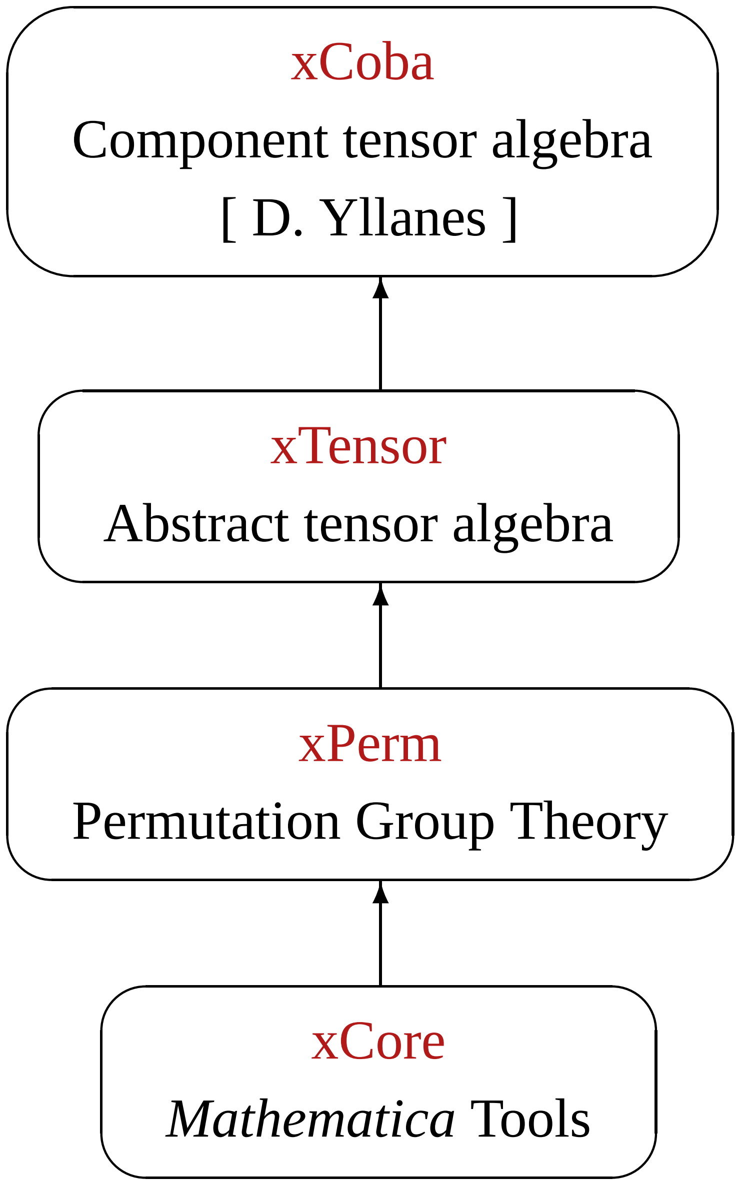
3. The xAct project



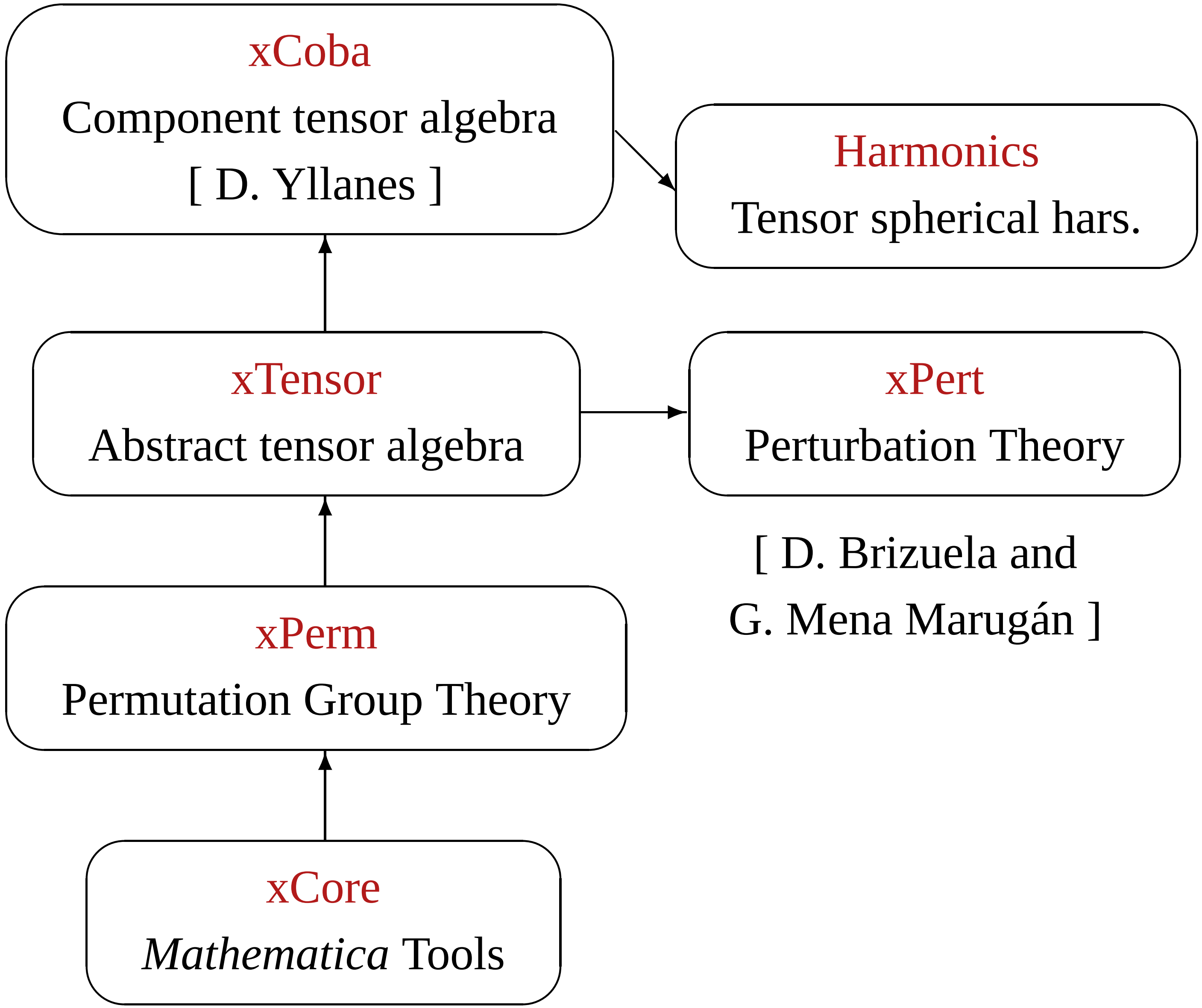
3. The xAct project



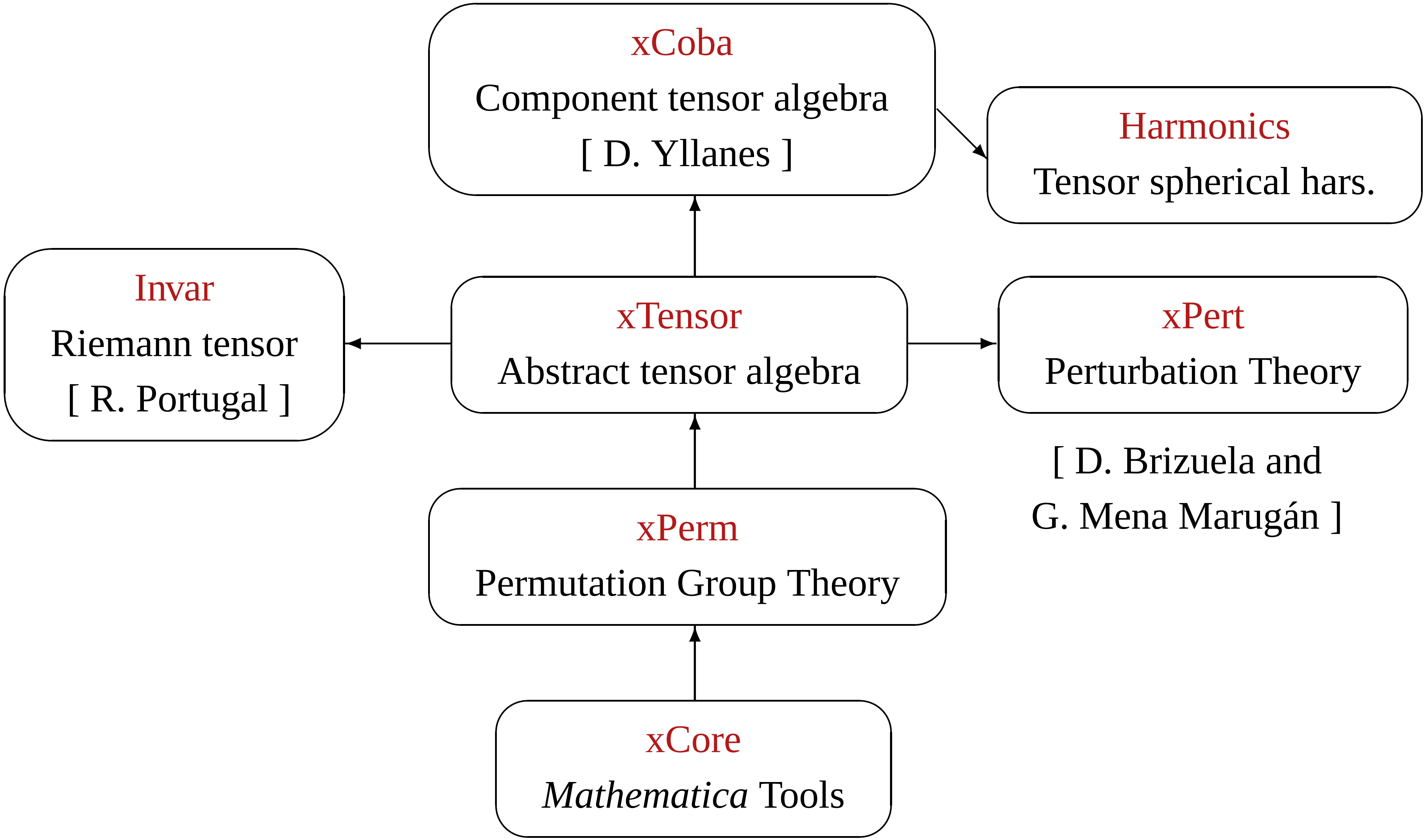
3. The *xAct* project



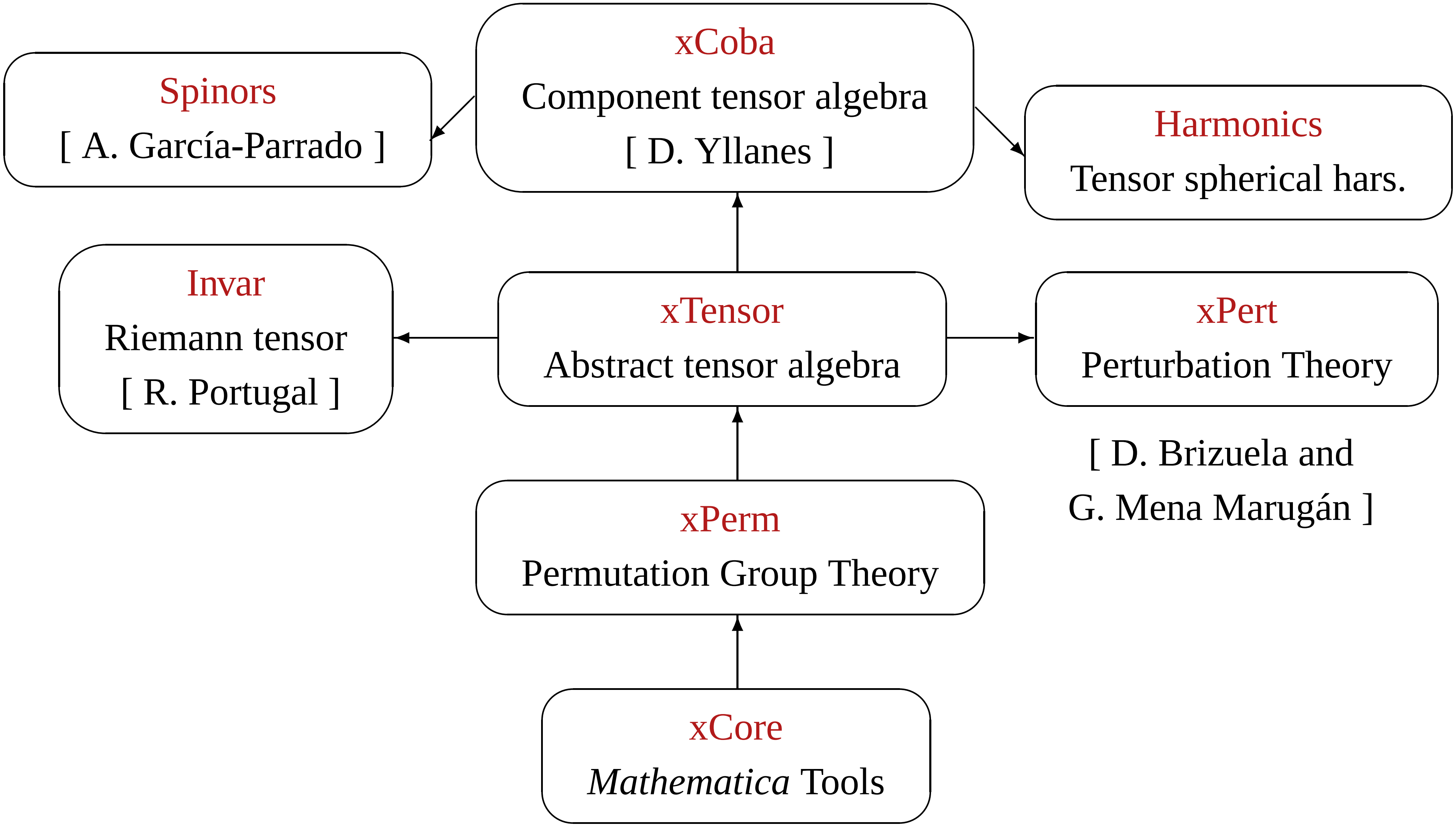
3. The xAct project



3. The xAct project



3. The xAct project



3b. Features

3b. Features

Strights:

- Fast monoterm canonicalization.
- Mathematical structure, GR-oriented.
- Free software (GPL).
- Extensive, *Mathematica*-style documentation.

3b. Features

Strengths:

- Fast monoterm canonicalization.
- Mathematical structure, GR-oriented.
- Free software (GPL).
- Extensive, *Mathematica*-style documentation.

Current weaknesses:

- Multiterm canonicalization missing.
- Tensor-editing interface missing.
- Incomplete development of calculus with charts.

3b. Features

Strengths:

- Fast monoterm canonicalization.
- Mathematical structure, GR-oriented.
- Free software (GPL).
- Extensive, *Mathematica*-style documentation.

Current weaknesses:

- Multiterm canonicalization missing.
- Tensor-editing interface missing.
- Incomplete development of calculus with charts.

Other data:

- ©2002–2009, GPL. Version 0.7 in March 2004; currently in 0.9.8
- 17000 lines of Mathematica code + 2500 lines of C code.
- 31 articles have used it:

3c. Results

- Hyperbolicity analysis of the Einstein equations (Gundlach & JMM)
- High order perturbation theory in GR (Brizuela & JMM)
- Invariants of the Riemann tensor (JMM & Portugal)
- The light-cone theorem (Choquet-Bruhat, Chruściel & JMM)
- Superfield integrals in string theory (Green et al.)
- Dynamical laws of superenergy (García-Parrado)
- Initial data sets for the Schwarzschild spacetime (GP & Valiente)
- Cosmological perturbation theory (Pitrou et al.)
- Post-Newtonian computations (Blanchet et al.)
- Quantum Field Theory (Álvarez et al.)
- “Galileon” (Deffayet et al.)

3d. What xTensor can do

3d. What xTensor can do

- Define one or several manifolds, and products of them.
- Define (complex) vector bundles on them.
- Define tensors with arbitrary monoterm symmetries.
- Define connections of any type. Automatic Christoffel, Riemann, ... generation.
- Define one or several metrics on each manifold. Warped metrics. Induced metrics.

3d. What xTensor can do

- Define one or several manifolds, and products of them.
- Define (complex) vector bundles on them.
- Define tensors with arbitrary monoterm symmetries.
- Define connections of any type. Automatic Christoffel, Riemann, ... generation.
- Define one or several metrics on each manifold. Warped metrics. Induced metrics.
- Parametric derivatives, Lie derivatives, Poisson brackets.

3d. What xTensor can do

- Define one or several manifolds, and products of them.
- Define (complex) vector bundles on them.
- Define tensors with arbitrary monoterm symmetries.
- Define connections of any type. Automatic Christoffel, Riemann, ... generation.
- Define one or several metrics on each manifold. Warped metrics. Induced metrics.
- Parametric derivatives, Lie derivatives, Poisson brackets.
- Single canonicalization routine (ToCanonical).

3d. What xTensor can do

- Define one or several manifolds, and products of them.
- Define (complex) vector bundles on them.
- Define tensors with arbitrary monoterm symmetries.
- Define connections of any type. Automatic Christoffel, Riemann, ... generation.
- Define one or several metrics on each manifold. Warped metrics. Induced metrics.
- Parametric derivatives, Lie derivatives, Poisson brackets.
- Single canonicalization routine (ToCanonical).
- Transformation rules, which detect the character of the indices, the manifold of the indices, the symmetries of the tensors, and take into account metrics.

3d. What xTensor can do

- Define one or several manifolds, and products of them.
- Define (complex) vector bundles on them.
- Define tensors with arbitrary monoterm symmetries.
- Define connections of any type. Automatic Christoffel, Riemann, ... generation.
- Define one or several metrics on each manifold. Warped metrics. Induced metrics.
- Parametric derivatives, Lie derivatives, Poisson brackets.
- Single canonicalization routine (ToCanonical).
- Transformation rules, which detect the character of the indices, the manifold of the indices, the symmetries of the tensors, and take into account metrics.
- Use all machinery of *Mathematica*.

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for ${T^a}_b$?

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for ${T^a}_b$?
  - `T[a,-b]`

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for ${T^a}_b$?
  - `T[a,-b]`
  - `Tensor[Name["T"], Indices[Up[a], Down[b]]]`

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for $T^a{}_b$?
  - `T[a,-b]`
  - `Tensor[Name["T"], Indices[Up[a], Down[b]]]`
  - Translator between internal / external notations

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for ${T^a}_b$?
  - `T[a,-b]`
  - `Tensor[Name["T"], Indices[Up[a], Down[b]]]`
  - Translator between internal / external notations
  - Choice: `T[a,-b]`

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for ${T^a}_b$?
  - `T[a,-b]`
  - `Tensor[Name["T"], Indices[Up[a], Down[b]]]`
  - Translator between internal / external notations
  - Choice: `T[a,-b]`
- Notation for covariant derivatives:
  - (Penrose & Rindler) What are $\nabla_2 v_3$ and $\partial_2 v_3$?
    Is it ${e_2}^a {e_3}^b \nabla_a v_b$ or ${e_2}^a \nabla_a \left({e_3}^b v_b\right)$?

3e. xTensor notations

- Use always abstract tensor notation. Penrose abstract-indices.
- Wald's book conventions. Ashtekar's approach to gauge.
- Symbols have a type: tensor, abstract-index, manifold, ...
- Notation for $T^a{}_b$?
  - `T[a,-b]`
  - `Tensor[Name["T"], Indices[Up[a], Down[b]]]`
  - Translator between internal / external notations
  - Choice: `T[a,-b]`
- Notation for covariant derivatives:
  - (Penrose & Rindler) What are $\nabla_2 v_3$ and $\partial_2 v_3$?
    Is it $e_2{}^a e_3{}^b \nabla_a v_b$ or $e_2{}^a \nabla_a \left(e_3{}^b v_b\right)$?
  - Choice: ∇_a and ∂_a are operators: `Cd[-a][expr]`
- Lie derivatives: `LieD[ v[i] ][ expr ]`

4. The Riemann invariants

- Scalar inv $R^{abcd}R_{ab}{}^{ef}R_{cdef}$, or differential inv $R_{ab}\nabla_c\nabla_d R^{acbd}$.
- Used in the classification of spacetimes, GR Lagrangian expansions, QG renormalization, etc.

4. The Riemann invariants

- Scalar inv $R^{abcd}R_{ab}{}^{ef}R_{cdef}$, or differential inv $R_{ab}\nabla_c\nabla_d R^{acbd}$.
- Used in the classification of spacetimes, GR Lagrangian expansions, QG renormalization, etc.
- Haskins 1902: 14 independent scalar invs in 4d.
- Narlikar and Karmarkar 1948: first list of 14 scalar invs: $(R, R^2, W^2, R^3, W^3, R^2W, W^4, R^4, R^2W^2,$ $W^5, R^2W^3, R^4W, R^4W^2, R^4W^3)$.

4. The Riemann invariants

- Scalar inv $R^{abcd}R_{ab}{}^{ef}R_{cdef}$, or differential inv $R_{ab}\nabla_c\nabla_d R^{acbd}$.
- Used in the classification of spacetimes, GR Lagrangian expansions, QG renormalization, etc.
- Haskins 1902: 14 independent scalar invs in 4d.
- Narlikar and Karmarkar 1948: first list of 14 scalar invs: $(R, R^2, W^2, R^3, W^3, R^2W, W^4, R^4, R^2W^2, W^5, R^2W^3, R^4W, R^4W^2, R^4W^3)$.
- Other invariants are (possibly nonpolynomial) functions of them.
- Completeness: polynomic basis for all invariants. Relations?

4. The Riemann invariants

- Scalar inv $R^{abcd}R_{ab}{}^{ef}R_{cdef}$, or differential inv $R_{ab}\nabla_c\nabla_d R^{acbd}$.
- Used in the classification of spacetimes, GR Lagrangian expansions, QG renormalization, etc.
- Haskins 1902: 14 independent scalar invs in 4d.
- Narlikar and Karmarkar 1948: first list of 14 scalar invs: $(R, R^2, W^2, R^3, W^3, R^2W, W^4, R^4, R^2W^2,$ $W^5, R^2W^3, R^4W, R^4W^2, R^4W^3)$.
- Other invariants are (possibly nonpolynomial) functions of them.
- Completeness: polynomic basis for all invariants. Relations?
- Sneddon 1999: 38 s-invs (deg 11). Complete basis. (Rotor calculus.)
- Carminati-Lim 2007: graph-based construction of s-inv relations.

4. The Riemann invariants

- Scalar inv $R^{abcd}R_{ab}{}^{ef}R_{cdef}$, or differential inv $R_{ab}\nabla_c\nabla_d R^{acbd}$.
- Used in the classification of spacetimes, GR Lagrangian expansions, QG renormalization, etc.
- Haskins 1902: 14 independent scalar invs in 4d.
- Narlikar and Karmarkar 1948: first list of 14 scalar invs: $(R, R^2, W^2, R^3, W^3, R^2W, W^4, R^4, R^2W^2,$ $W^5, R^2W^3, R^4W, R^4W^2, R^4W^3)$.
- Other invariants are (possibly nonpolynomial) functions of them.
- Completeness: polynomic basis for all invariants. Relations?
- Sneddon 1999: 38 s-invs (deg 11). Complete basis. (Rotor calculus.)
- Carminati-Lim 2007: graph-based construction of s-inv relations.
- Fulling et al 1992: bases for d-invs up to 10 derivatives and R^6.

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants
- Use cyclic property: 1639 invariants

4b. Riemann invariants up to degree 7

- 7 Riemann $\quad\longrightarrow\quad$ 28 indices $\quad\longrightarrow\quad$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants
- Use cyclic property: 1639 invariants
- Antisymmetrise over sets of 5 indices in 4d: 4311 integer equations

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants
- Use cyclic property: 1639 invariants
- Antisymmetrise over sets of 5 indices in 4d: 4311 integer equations
- Gauss elimination: Intermediate swell problem:
  - Original integers: $|c_i| \leq 384$

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants
- Use cyclic property: 1639 invariants
- Antisymmetrise over sets of 5 indices in 4d: 4311 integer equations
- Gauss elimination: Intermediate swell problem:
  - Original integers: $|c_i| \leq 384$
  - Final integers: $|c_f| \leq 4978120$

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants
- Use cyclic property: 1639 invariants
- Antisymmetrise over sets of 5 indices in 4d: 4311 integer equations
- Gauss elimination: Intermediate swell problem:
  - Original integers: $|c_i| \leq 384$
  - Final integers: $|c_f| \leq 4978120$
  - Largest intermediate integer:

41284320888114626312105608472587963577277605659619553121090337199745221229120263938390479190798331362
9332658685142520797678355583630907136564045657358541395134420154091637344181884734088835920651682654
5838550350981924166243854816386358011913196352080821446913112544207777945582431400973483457511551249
4271650405780114863779964319571108875740447236120954785394441817343113274871351581501474081446209153
5133993980627211654318697002059693685910102607365896788999230680327719504392651078493689021476459822
1917466623055176060658271638645490139036389024466375930303586688550738508615214422459534528028026604
3392962118745453989341765134526682155897073108863212658336777829476719031970391332987380834358579837
4768508365933468022268161668651405869982994847652173877241170117828300225631267244981449350418876807
8308059566617955048754211100127225300485494079978006577938025856377710049543176142178387315401497328

  - Required 8Gb RAM and 384-bytes integers (BigNum: GMP).

4b. Riemann invariants up to degree 7

- 7 Riemann $\longrightarrow$ 28 indices $\longrightarrow$ $28! \simeq 3.0 \cdot 10^{29}$ s-invs
- Use monoterm symmetries: 16352 invariants
- Use cyclic property: 1639 invariants
- Antisymmetrise over sets of 5 indices in 4d: 4311 integer equations
- Gauss elimination: Intermediate swell problem:
  - Original integers: $|c_i| \leq 384$
  - Final integers: $|c_f| \leq 4978120$
  - Largest intermediate integer:

4128432088811462631210560847258796357727760565961955312109033719974522122912026393839047919079833136 2
9332658685142520797678355583630907136564045657358541395134420154091637344181884734088835920651682654
5838550350981924166243854816386358011913196352080821446913112544207777945582431400973483457511551249
4271650405780114863779964319571108875740447236120954785394441817343113274871351581501474081446209153
5133993980627211654318697002059693685910102607365896788999230680327719504392651078493689021476459822
1917466623055176060658271638645490139036389024466375930303586688550738508615214422459534528028026604
3392962118745453989341765134526682155897073108863212658336777829476719031970391332987380834358579837
4768508365933468022268161668651405869982994847652173877241170117828300225631267244981449350418876807
8308059566617955048754211100127225300485494079978006577938025856377710049543176142178387315401497328

  - Required 8Gb RAM and 384-bytes integers (BigNum: GMP).
- We get the 27 invs of Sneddon's basis up to degree 7 (6 dual), plus all polynomial expression of any other invariant. Invar package: CPC 2007.
- Database of 645 625 relations up to 12 metric derivatives. CPC 2008.

4c. Riemann invariants

Degree	A	A*	B	B*	C	C*	D	D*
1	1	1	1	0	1	0	1	0
2	3	4	2	1	2	1	2	1
3	9	27	5	6	3	2	3	2
4	38	232	15	40	4	1	3	1
5	204	2582	54	330	5	2	3	2
6	1613	35090	270	3159	8	2	4	2
7	16532	558323	1639	–	7	(1)	3	(1)
8	217395	–	13140	–	(9)	(1)	(2)	(1)
9	3406747	–	–	–	(11)	(1)	(3)	(1)
10	–	–	–	–	(9)	(1)	(1)	(1)
11	–	–	–	–	(9)	(0)	(1)	(0)
12	–	–	–	–	(9)	(0)	(0)	(0)

A: Permutation symmetries, B: Cyclic symmetry,
C: Dim-dep identities, D: Products of duals (*).

5. Conclusions

5. Conclusions

- Brief review of the field of Computer Algebra, focusing on the importance of the canonicalizer.

5. Conclusions

- Brief review of the field of Computer Algebra, focusing on the importance of the canonicalizer.
- There are important problems in GR and other fields which require Tensor Computer Algebra.

5. Conclusions

- Brief review of the field of Computer Algebra, focusing on the importance of the canonicalizer.
- There are important problems in GR and other fields which require Tensor Computer Algebra.
- For the first time we have tensor packages with efficient canonicalization algorithms for monoterm symmetries, and they are free software!

5. Conclusions

- Brief review of the field of Computer Algebra, focusing on the importance of the canonicalizer.
- There are important problems in GR and other fields which require Tensor Computer Algebra.
- For the first time we have tensor packages with efficient canonicalization algorithms for monoterm symmetries, and they are free software!
- There are multiterm algorithms, but we need something more efficient. Idea: databases of solutions (example: Invar package).

5. Conclusions

- Brief review of the field of Computer Algebra, focusing on the importance of the canonicalizer.
- There are important problems in GR and other fields which require Tensor Computer Algebra.
- For the first time we have tensor packages with efficient canonicalization algorithms for monoterm symmetries, and they are free software!
- There are multiterm algorithms, but we need something more efficient. Idea: databases of solutions (example: Invar package).
- xAct implements the fastest algorithms, in a GR-oriented structure based on Penrose abstract indices. Well tested and documented.

`http://metric.iem.csic.es/Martin-Garcia/xAct/`

`http://luth.obspm.fr/~luthier/Martin-Garcia/xAct/`