Giac and GeoGebra: improved Gröbner basis computations

Z. Kovács, B. Parisse

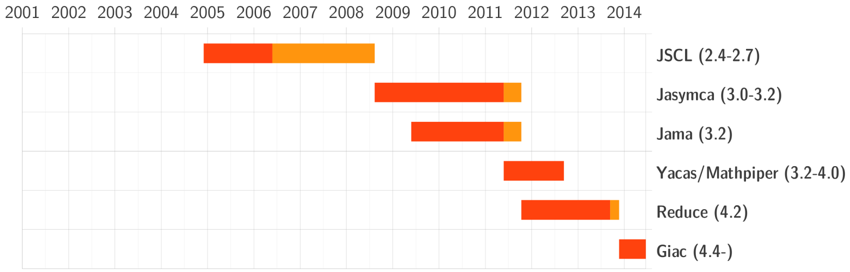
JKU Linz, University of Grenoble I

November 25, 2013

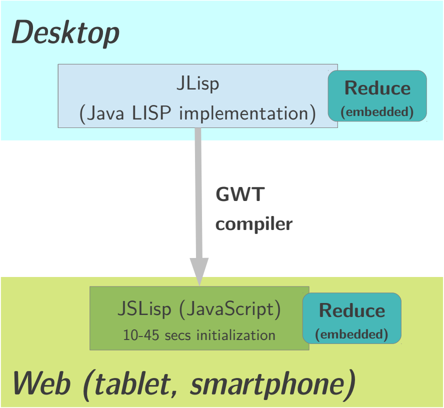
Two parts talk

1. GeoGebra (Z. Kovács)
2. Gröbner basis over $\mathbb{Q}$ in Giac (B. Parisse)

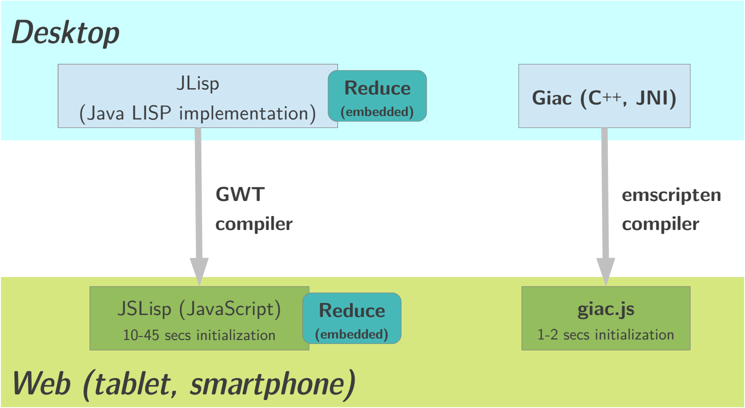
History of used CAS in GeoGebra



Comparing JSLisp...



Comparing JSLisp and giac.js



Gröbner basis benchmark (problems in elem. geom.)

Ubuntu Linux 11.10, Intel Xeon CPU E3-1220 V2 3.1 GHz, RAM 1 GB

Test	Maxima	JAS 2	Reduce	Singular	CoCoA	Giac
Thales	0.2	0.46	0.11	0.00	0.08	0.03
Heights	0.29	0.51	0.11	0.00	0.29	0.03
Medians	0.4	0.65	0.12	0.00	0.14	0.09
Bisectors	0.42	0.5	0.1	0.00	0.09	0.01
Euler		1.66	0.2	0.01	0.14	0.01
Nine	1.19	1.5	0.11	0.01	0.13	0.01
Angle	36.08	1.74	0.75	0.01	0.31	0.04
Simson						
Pappus		3.37		0.5	9.28	4.9
Simson-R		5.77	6.07	0.07	0.87	0.15
Pappus-R		2.33	2.18	0.02	0.34	0.4
Average	***6.43***	***1.85***	***1.08***	***0.06***	***1.17***	***0.57***
Average*	***56.43***	***7.85***	***14.41***	***6.06***	***7.17***	***6.56***

Direct applications in GeoGebraWeb

- Machine generated geometry theorem proving ▸ Play
- Real-time locus equation computation ▸ Play

Main points

- Let $I =< f_1, ..., f_m >$ be the ideal generated by $f_1, ..., f_n$ polynomials in $\mathbb{Q}[x_1, .., x_n]$, $<$ be a total monomial ordering (currently revlex ordering supported), G the corresponding Gröbner basis.
- Giac implements the modular algorithm described in E. Arnold, Journal of Symbolic Computation, 2003.
- It finds the Gröbner basis modulo several primes (Buchberger algorithm with F4-like linear algebra), with parallelisation.
- The first prime run records informations that speed up further prime runs (*learning trick* like F4remake from Joux-Vitse).
- The user chooses between a fast probabilistic answer (with confidence level) or a much slower certified answer.
- Reference: arXiv:1309.4044

Computation in $\mathbb{Z}/p\mathbb{Z}$

- Begin main loop: all critical pairs with minimal total degree are reduced simultaneously (like in F4).
- Collect all monomials of all shifted polynomials belonging to a pair, excluding the leading term (using a heap).
- Heap division by the current basis (Monagan, Pearce) without taking care of the coefficients, to each basis element corresponds a quotient. Records this for next prime runs.
- Reduction by division of the critical pairs (pairs are grouped like in the Buchberger algorithm, unlike in the F4 algorithm, there is no simplification step).
- Dense inter-reduction of the remainders. Non-zero lines are added to the basis, and the list of pairs is updated (Gebauer-Möller rules), then run again the main loop
  Keep track of 0 reducing pairs for further runs.

Modular algorithm

- Imagine we do the same computation on $\mathbb{Q}$ and normalize the non zero reducing critical pairs to have coefficients in $\mathbb{Z}$ and be primitive. If all the leading coefficients do not cancel modulo a prime p, then G modulo p is the Gröbner basis on $\mathbb{Z}/p\mathbb{Z}$.
- Algorithm: reconstruct a basis $\tilde{G}$ on $\mathbb{Q}$ using rational reconstruction of the Chinese remainder of Gröbner basis G_{p_i} modulo several primes p_i having the same leading monomials. Once $\tilde{G}$ stabilizes, check that the original f_i belongs to the ideal generated by $\tilde{G}$ (fast first check).
- Arnold theorem: if $\tilde{G}$ is a Gröbner basis, then $G = \tilde{G}$ (slow second check).
- The proof does not require that all G_{p_i} are Gröbner basis, it is sufficient that one of the G_{p_i} is a Gröbner basis.

Probabilistic vs certified

- First run returns a certified Gröbner basis modulo p, next runs are not certified if we use the *learning* speed-up: failure may happen if one of the critical pair reduces to 0 modulo the first prime, but not in $\mathbb{Q}$. This is extremly unlikely because non-zero reducing critical pairs have many terms, probability is $1/p^{\#\text{terms}}$.
- If several primes were used to reconstruct the Gröbner basis $\tilde{G}$ and the first check passes, it is also very unlikely that the leading monomials do not coincide with G.
- Default behavior in Giac is to certify the Gröbner basis (second check) if the number of elements of the basis is less than a given constant (currently 50), otherwise the second check is not done unless the user has set `proba_epsilon` to 0. Depending on this value, $\tilde{G}$ is checked to be a Gröbner basis modulo a few primes.

Benchmarks

- Giac is several times faster than Singular modulo *p*, it solves problem on $\mathbb{Q}$ that can not be solved by Singular (cyclic8 in less than 2 minutes, cyclic9 in about 1 day).
- The probabilistic algorithm on $\mathbb{Q}$ is as fast as Maple, about 3 times slower than Magma on 1 CPU for cyclic* or katsura* and sometimes almost as fast for more random inputs. The certified algorithm is slower... but you must also check closed-source output to make a valid mathematical proof.
- Low memory footprint (e.g. 3 GB of RAM for cyclic9 vs 7.7 GB for Magma). This is important for the CAS using Giac as kernel (Xcas, GeoGebra, qcas; HP Prime calculator; PocketCAS, CAS Calc P11, Androcas, Xcas Pad iOS and Android applications; pygiac).

Comparison with Singular

Mac OS X.6, Dual Core i5 2.3 GHz, RAM 2 $\times$ 2Go

	Giac mod p	Giac run2	Singular mod p	Giac $\mathbb{Q}$ 1e-16	Giac $\mathbb{Q}$ certif.	Singular $\mathbb{Q}$
cyclic7	.5–.58	0.1	2.0	4.2	21	>2700
cyclic8	7.2–8.9	1.8	52.5	106	258	»
cyclic9	633–1340	200	?	1 day	»	»
katsura8	.06–.07	0.009	0.2	0.53	6.6	4.9
katsura9	.29–.39	0.05	1.37	3.2	54	41
katsura10	1.5–2.3	0.3	11.65	20.7	441	480
katsura11	10–14	2.8	86.8	210	4610	?
katsura12	76–103	27	885	1950	»	»
alea6	0.8–1.1	.26	4.18	204	738	>1h

Further work

- FGLM should be optimized, e.g. for the `solve` and `eliminate` commands.
- Other open-source packages (Singular, CoCoA, Macaulay...) could be interested in this algorithm, it is “fast enough”, relies on basic Gröbner basis theory and does not require advanced sparse linear algebra over $\mathbb{Z}$ (10K of source code in Giac, perhaps 20 times more in Fgb?).
  Life is short and ROM is full.