From release 0.91 to release 0.95.1

The Alt-Ergo developers team
Overview of the main changes

- arithmetic enhancements
- AC symbols
- new built-in theories: arrays, enumerated data types, records
- models / unsat cores extraction
- a graphical interface (AltGr-Ergo)
- Alt-Ergo-Zero library
Arithmetic reasoning

Linear arithmetic on $\mathbb{Z}$

- a new decision procedure FM-Simplex
- good results on QF-LIA category of SMT benchmarks
- published at [IJCAR 2012]

Non-linear arithmetic

- Euclidean division and modulo operators
- interval calculus
- non-linear multiplication
- good results on ANR Decert benchmark
AC Symbols

AC(X), new algorithm for combining a Shostak theory X with a decision procedure for AC symbols

- published at [TACAS 2011, LMCS 2012]
- EATCS award for Best Theoretical Paper at ETAPS 2011

logic ac u : int, int -> int
goal g :
  forall x,y,z,a,b:int.
  u(a,b)-b = x and u(a+b,c) = y and b = 0 ->
  u(0,y) = u(c,x)
New built-in theories

Functional arrays

logic a : (int, int) farray
goal g1 : forall i:int. i=6 -> a[i<-4][5] = a[i-1]

Records

type 'a t = { a : int; b : 'a }
goal g2 : forall v,w:int t.
  2 * v.a = 10 -> { v with b = 5} = w -> w.a = 5

Enumerated data types

type t = A | B | C
logic P : t -> prop
goal g3 : forall x:t. P(C) -> x<>A and x<>B -> P(x)
Models extraction

logic x "model:0", y "model:0" : int
goal g: x >= 42 -> x <> y -> y = 45 -> ((x + 1)) <= 40

alt-ergo -model <file>

Propositional:
42 <= x
x <> y
y = 45

Theory:
y = X1(arith):[45 [int]]
x <> y

Relation:
x ∈ [42; 44] ∪ [46; +∞[ 
Unsat cores extraction

logic x, y : int
goal g: x \geq 4 \rightarrow x \not< y \rightarrow y = 2 \rightarrow y - x \leq 0

alt-ergo -proof <file>

Proof:
4 \leq x
y = 2
(y - x) > 0

x \not< y is not used to derive the unsatisfiability
AltGr-Ergo: capabilities

- selection/deletion of axioms and hypotheses
- deletion/modification of triggers
- manual (and possibly partial) axioms instantiation
- highlight which axioms/hypotheses were useful to prove a goal
- axioms instantiation and decision procedures profiling
- save/replay modifications in/from a session file
AltGr-Ergo: example

```plaintext
axiom def_axiom1:
    a_brake_emergency_model = of_int1(0)
    and a_brake_emergency_model1 = sec_3.13.7_monitoring_inputs_70_1
    and a_brake_emergency_model2 = sec_3.13.2_monitoring_inputs_7

type speed_t

logic attr__ATTRIBUTE_MODULUS4 : real

predicate in_range4 (x:real) =
    forall x:real [in_range4(x)].
    in_range4(x)
    <= x
    and x <= from_int(340282346638528859811704183484516925440)

logic to_real2 : speed_t -> real

logic of_real2 : real -> speed_t

predicate eq4 (x:speed_t,y:speed_t) =
    forall v: real [v.speed_t + v.speed_t = v.speed_t]

goal WP_parameter_def
```

Evaluation

What is next?
AltGr-Ergo: example

```
axiom def_axiom:
  a_brake_emergency_model = of_int1(0)
  and a_brake_emergency_model1 = sec_3_13_2_monitoring_inputs_70_15
  and a_brake_emergency_model2 = sec_3_13_2_monitoring_inputs_70_15

type speed_t

logic attr__ATTRIBUTE_MODULUS4 : real

predicate in_range4 (x:real) = forall x:real [in_range4(x)].
  in_range4(x) <= x
  and x <= from_int(340282346638528859811704183484516925440)

logic to_real2 : speed_t -> real

logic of_real2 : real -> speed_t

predicate end (x:speed_t,y:speed_t) = forall y:speed_t [x+y] mod 4

goal WP_parameter_def
```
A new library

Alt-Ergo zero

an OCaml SMT library

Enhanced and light version of Alt-Ergo:

- a new SAT solver based on a re-implementations of minisat
- incremental
- support several instances
- no quantifiers
- used in model-checking and k-induction

Toccata: LRI & INRIA–Saclay
Additional explored topics

- a lightweight proofs certification mechanism using COQ
- built-in support of floating point numbers
  - integration of Gappa in Alt-Ergo [SMT-Workshop 2012]
### Why3 benchmark

<table>
<thead>
<tr>
<th></th>
<th>trunk</th>
<th>0.95.1</th>
<th>0.94</th>
<th>0.93</th>
<th>0.92.2</th>
<th>0.91</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid</td>
<td>1841</td>
<td>1841</td>
<td>1811</td>
<td>1773</td>
<td>1737</td>
<td>1685</td>
</tr>
<tr>
<td>time</td>
<td>353</td>
<td>362</td>
<td>465</td>
<td>411</td>
<td>527</td>
<td>555</td>
</tr>
<tr>
<td>unknown</td>
<td>21</td>
<td>20</td>
<td>25</td>
<td>24</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>time</td>
<td>47</td>
<td>13</td>
<td>31</td>
<td>25</td>
<td>33</td>
<td>45</td>
</tr>
<tr>
<td>timeout</td>
<td>58</td>
<td>59</td>
<td>83</td>
<td>99</td>
<td>128</td>
<td>175</td>
</tr>
<tr>
<td>errors</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>33</td>
<td>39</td>
</tr>
</tbody>
</table>

1920 formulas timeout: 30 seconds
## Hi-lite public benchmark

<table>
<thead>
<tr>
<th></th>
<th>trunk</th>
<th>0.95.1</th>
<th>0.94</th>
<th>0.93</th>
<th>0.92.2</th>
<th>0.91</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>valid</strong></td>
<td>2418</td>
<td>2397</td>
<td>2352</td>
<td>1526</td>
<td>2286</td>
<td>2374</td>
</tr>
<tr>
<td><strong>time</strong></td>
<td>847</td>
<td>723</td>
<td>988</td>
<td>158</td>
<td>869</td>
<td>1110</td>
</tr>
<tr>
<td><strong>unknown</strong></td>
<td>649</td>
<td>487</td>
<td>138</td>
<td>76</td>
<td>378</td>
<td>387</td>
</tr>
<tr>
<td><strong>time</strong></td>
<td>2931</td>
<td>1438</td>
<td>975</td>
<td>196</td>
<td>1217</td>
<td>1353</td>
</tr>
<tr>
<td><strong>timeout</strong></td>
<td>518</td>
<td>691</td>
<td>995</td>
<td>283</td>
<td>910</td>
<td>815</td>
</tr>
<tr>
<td><strong>errors</strong></td>
<td>0</td>
<td>10</td>
<td>100</td>
<td>1700</td>
<td>11</td>
<td>7</td>
</tr>
</tbody>
</table>

**3583 formulas**

**timeout : 30 seconds**
What is next?

- Floating point numbers, COQ certification, lemmas instantiation, models generation, non-linear arithmetic, ...
- ANR bware project
  - improving Alt-Ergo for POs coming from Atelier-B
- commercial support for Alt-Ergo by OCamlPro