RCL: Reduce learnt clauses
Gilles Audemard, Jean-Marie Lagniez, Bertrand Mazure, Lakhdar Saïs

To cite this version:
Gilles Audemard, Jean-Marie Lagniez, Bertrand Mazure, Lakhdar Saïs. RCL: Reduce learnt clauses. 2013. hal-00872811

HAL Id: hal-00872811
https://hal.science/hal-00872811
Submitted on 14 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
RCL: Reduce learnt clauses

Gilles Audemard, Jean-Marie Lagniez, Bertrand Mazure, and Lakhdar Saïs

Université Lille-Nord de France CRIL - CNRS UMR 8188 Artois, F-62307 Lens
{audemard,lagniez,mazure,sais}@cril.fr

Abstract. This note describes features of the version of RCL that entered the SAT-race 2010 affiliated to the SAT’2010 conference in Edinburgh, Scotland, UK.

1 Overview

RCL is a SAT solver which includes all the modern enhancements of the DPLL procedure as they can be found in solvers such as RSAT [4] and MINISAT [2]. These enhancements include watched literal to the unit propagation, first-UlP learning scheme, frequent restarts (Luby strategy [3]), activity-based decision heuristics (VSIDS), and the phase learning policy is used [4]. The main improvement consists in storing the binary clauses in an adjacency list, and in reducing the learnt clauses. These learnt clauses are reduced by resolution with the binary clauses. For reducing learnt clauses database, clauses are sorted by using the phase. This step allows to associate a weight with each clause. This weight is the number of falsified literals. Once the clauses are sorted, half of them are kept. For not too large instances, we use SatElite as a pre-processor [1].

2 Code

The system is written in C and has about 3000 lines of code. It was submitted to the race as a 64 bit binary. It is written from scratch.

References