# Barcelogic SAT-solver
# System Description

Ignasi Abío, Roberto Asín, Robert Nieuwenhuis, Albert Oliveras,
Enric Rodríguez-Carbonell
Technical Univ. of Catalonia, Barcelona.

June, 2010

## 1 About Barcelogic

Barcelogic is a carefully designed and implemented CDCL SAT-solver. It is implemented in C++ and compiled for 32-bit architecture. Its design is aimed at obtaining a clean and flexible system so that we can easily adapt the solver to our various research needs. For instance, it is now used for SMT, MaxSAT, Proof generation and Core extraction. Moreover, its flexible implementation allows us to easily modify it in order to boost its performance in different application areas such as timetabling, scheduling, EDA industry, etc.

Barcelogic is a carefully implemented CDCL SAT-solver with the following main characteristics:

- *Conflict analysis:* 1UIP learning scheme.
- *Decision heuristic:* EVSIDS heuristic.
- *Preprocessor:* Satelite
- *Restarts:* adaptive Luby restarts as explained below.
- *Cleanup policy:* Not very frequent cleanups, deleting only lemmas with more than three literals with small activity.

Binary clauses are stored in specific structures in order to achieve a faster propagation and give it priority over longer clauses.

## 2 Special algorithms

### 2.1 Unit detection

In order to derive some new unit clauses logically entailed by the original clause set we have developed a unit-propagation-based method, motivated by the the fact that unit propagation is an extremely optimized procedure in any SAT-solver. The propagation-based unit detection algorithm we propose is to be executed at decision level zero and is as follows.

Consider a clause $C$ and take its first literal $l$. Assume $l$ is true, and apply unit propagation, marking all literals that become true in this process. Now, repeat the process with the second literal and so on. If some literal $l'$ becomes true in the propagation of all literals of $C$, we know that $l'$ is a unit consequence of the input. The underlying idea is easy to understand. For a clause $C$ to be true, at least one of its literals must be true. If we know that, for every literal $l$ in the clause, $F \wedge l \models l'$ then $l'$ must be true in any model of $F$.

## 2.2 Adaptive Luby restarts

Our SAT-solver implements a Luby-based restart strategy, but with a slight modification that tries to make it adaptive: we observe the behavior (e.g. the backjumping levels) of the search during the last $L$ steps and decide, if a restart has been proposed, whether the search behaves well enough to postpone the restart or not. More specifically, based on Luby's series, we propose a restart after certain number of conflicts have occurred. When this happens, we check whether the minimum (and maximum) backjump levels found are below (above) certain min (max) threshold constants $C_{min}$ ($C_{max}$). For our implementation, we have used: $L = 24$, $C_{min} = \frac{V}{10000} + 10$ and $C_{max} = V - (\frac{V}{10000} + 10)$ where $V$ is the number of variables. If this is the case, we skip the restart.