We start reporting preliminaries results by referring to two published articles. Follows a discussion about these results. Later some related topics are also discussed. We finally propose ideas to improve the current results and to obtain general lessons from these results.

# 1 Introduction

Classical planning consists of finding a sequence of actions that transforms a known initial state into a goal, given a description of states and actions in terms of a set of variables. Conformant planning, on the other hand, is a variation of classical planning where the initial state is partially known and actions can have non-deterministic effects. A conformant plan must achieve the goal for all possible initial states and transitions.

Classical planning has been mapped into SAT with very remarkable results. Optimal planners based on SAT proceeds by looking for plans with length $n$, and starting with $n = 0$ until a plan is found. On the other hand, existence of a conformant plan with polynomial horizon is $\Sigma_2^p$ [Tur02], so it cannot be mapped directly into SAT. Castellini *et al.* [CGT03] report a conformant planner based on two inner SAT calls.

Knowledge compilation is concerned with the problem of mapping logical theories into suitable target languages that make certain desired operations tractable [SK96, CD97, DM02a]. For example, the compilation into Ordered Binary Decision Diagrams (OBDDs) renders a large number of operations tractable [Bry92]. While in all these cases, the compilation itself is intractable, its expense may be justified if the operations are to be used a sufficiently large number of times in the intended application. Moreover, while the compilation will run in exponential time and space in the worst case, it will not necessarily do so on average. Indeed, the compilation of theories into OBDDs has been found useful in formal verification [CGP00] and more recently in planning [GT99, CR00].

A more recent compilation language is *deterministic Decomposable Negation Normal Form* (d–DNNF [Dar02]). Theories in d–DNNF are directed acyclic graph, which leafs are literals, and the internal nodes are $\wedge$ or $\vee$. It graph can be understood as an exhaustive run of DPLL where every subproblem solved is solved once and cached [HD]. The d–DNNF target language support a rich set of polynomial–time operations and queries, as model counting and projection, that can be exponential over CNF. The well known OBDD normal form is a special case of d–DNNF. In fact, there are OBDDs which are exponentially bigger than their equivalents d–DNNF [DM02a].

We have written two articles about conformant planning using knowledge compilation to d–DNNF allowing to do operations or to verify properties over a problem in order to solve it. The first is titled *Pruning Conformant Plans by Counting Models on Compiled d–*DNNF *Representations* [PBDG05]. The second is titled *Mapping Conformant Planning to SAT through Compilation and Projection* [PG05]. On both works we assume deterministic actions. We consider solutions with one action per time step (serial setting) or allowing non-interfering actions to execute in the same time step (parallel setting). In the following section we summarize the principal ideas of both works in order to discuss about their results and propose further work. We assume that the

reader has read both articles. They are attached at the end of this report.

## 2 Preliminary work

When mapping classical planning to SAT, each model of the resulting theory corresponds to one possible execution, starting from an initial state until achieve the goal. In conformant planning we need a unique plan that works for *any* initial state. The models of a planning theory with non-determinism, and fixing a plan, correspond one-to-one to the different executions of the plan. Assuming that actions are deterministic, there is one execution for each initial state. Let $MC(T)$ the number of models of the theory $T$, $T(P)$ the propositional theory associated to the planning problem $P$, $T_0$ the theory describing the possible initial states, and $p$ plan valid for any initial state. Therefore,

$$MC(T_0) = MC(T(P) + p) \tag{1}$$

The number of models of the planning theory, assuming a plan, is equal to the number of initial situations.

This idea can be generalized for verifying partial plans. It requires an logical operation called *projection* (the dual of forgetting [LR94]) which allow us to obtain an equivalent theory but considering a subset of variables. Because d–DNNF supports model counting and projection in linear time, we can verify in linear time if a partial plan is valid or not. The planner reported in [PBDG05] report a search algorithm for conformant planning than look for valid plans doing this verification on each node. We will refer to it planner as the *model-counting-based planner*.

We comment also in [PBDG05] that it is possible to extract conformant plans with no search from compiled theories. However, it requires the compiler to consider the variables of the problem in an specific order that makes the compilation infeasible in most cases.

In [PG05] we argued than using projection, given an initial state $s_0$, the possible plans are the models of the following equation:

$$\mathsf{project}[\ T(P)\ +\ s_0\ ;\ \mathit{Actions}\ ] \tag{2}$$

which are the models of $T(P) + s_0$, refereed only to actions fluents.

Therefore, a conformant plan can be obtained by a *single* invocation to a SAT solver over (2) aggregated over all the initial states:

$$\bigwedge_{s_0\ \in\ models(T_0)} \mathsf{project}[\ T(P)\ +\ s_0\ ;\ \mathit{Actions}\ ] \tag{3}$$

It can be done because d–DNNFs allow to calculate the projection in linear time. This planner, called *sat-based planner* in this report, shows an important improvement over the reported in [PBDG05]. Notice that in this case we do not require model counting as in the model-counting-based planner.

With these ideas, explained in more detail in their respective articles, we obtained results comparable with the state of the art in optimal conformant planning. In all the experiments we have done, the compilation of theories has never been the bottleneck. We could compile many problems than we could

not solve by searching or calling a SAT solver. Our planner is strong in problems with high uncertainty, which solutions depends strongly on the interaction between the possible initial states. For example, in [PG05] we report solving `cube-center-9`. We do not known about any other optimal or sub-optimal conformant planning than can solve this problem. On the other hand, problems with low uncertainty, more close to classical planning are more difficult. A conformant planner reported by Brafman *et al.* (CFF, [BH04]) report good performance on many domains. Many of this problems generate very big theories and we cannot compile them. On the other hand, CFF is weak in some domains where we have very good results.

# 3 Discussion

Looking at the presented results, some questions arises. Could we use OBDDs instead of d–DNNFs as compiled form of the theories?. As we said, d–DNNFs are known to be more succinct than OBDDs. Moreover, mapping conformant planning to SAT, we need to forget over many variables in one shoot, which is not possible with OBDD [DM02b]. However, in future approaches to conformant planning through knowledge compilation, we cannot discard OBDDs or any other compilation language.

The performance of the model-counting-based planner is heavy affected by the variable selection: on every node of the search, which action variable choose to try. Other problem is that the compiled theories are in some cases quite big. Some of them has more than one million nodes that are visited on each model counting, on each node. Much of this process is repeated from a node to other but the only difference if the value of a leaf of the d–DNNF. Reuse some of this calculation is possible and may have a big impact.

We tried the no-search scheme, which allows to solve very easy problems. The resulting theories allow to obtain *all* the plans for a problem. Then, we tried to solve by compilation some layers of actions in the planning theory, leaving the other actions to be solved by searching. We wanted to find a trade-off between no-search and search, but it was difficult to find. The problem is that as more action layers we try to solve during compilation, the d–DNNF becomes bigger. As a consequence, when there are less actions for search, the node generation is very slow and we cannot explore this small space.

The sat-based planner is not affected by this phenomenon. Modern SAT solvers are quite good in avoiding unnecessary calculus. In my opinion, it is one reason of the improvement over the model-counting-based planner.

The experience in SAT approach to classical planning shows that we can improve the results by getting extra information from the original problem. For example, add the planning graph into the propositional theory allows much better results [KS99]. Relaxations and heuristics play an important role in modern planners. A problem with the model-counting-based planner is that, in general, it detects when a plan is inconsistent with *one* initial state. Rintanen [Rin04] obtains good results by trying on every pair of states to obtain a heuristic function.

The sat-based planner encodes the relation between different executions, so the SAT solver can avoid to explore other zones than *being inconsistent with one initial state*. Actually, we do not know if it makes a difference or not. Solving a

conformant planning problem with our sat approach was in many cases *easier* than solve the same problem modified to have two initial states. We want to find ways to guarantee that a branch can be pruned out because a reason involving more than one initial state.

If we want to solve fast some simple problems, it could be useful to consider an epistemic point of view. An agent may need to be sure about something before doing other actions and achieve the goal. In the emptyroom problem, where a blind agent look for the center of a room, an agent needs to find the corner of the space in order to guarantee that will be in the center after some actions. Cimatti *et al.* [CRB04] report heuristics based on this ideas, which lead to interesting results for the sub-optimal setting. The problem with these heuristics is that actions that *reduce the uncertainty* are considered better, but it is not always the best way to achieve the goal. A solution may require to lose information in general in order to be sure about something.

Symmetry plays a central role when the model-counting-based planner cannot decide if there is a plan for horizon $n$. We do not have explicit control over the behavior of the SAT solver but symmetries appears everywhere in planning. We are not thinking about dealing with them explicitly. Instead, we want to focus on other options to reduce the search space, both in the explicit search or the SAT solver search.

A variation of the propositional encoding we are using can make a difference. Parallel actions allows solving more complex problems. We are using a straightforward semantic where two actions interfere when they affect a common variable. Refining more this, we would be able to solve in one time step some problems used during the evaluation of CFF[BH04] (for example, the Safe problem). Rintanen discuss about other options for encode parallelism [RHN04]. The work about planning graphs and knowledge compilation of Geffner [Gef94] is also relevant, even for the serial setting. Other encoding can impact on the compilability of the theories and the final performance. Another option is to avoid the compilation step by doing projection through copy and renaming in the CNF. We can calculate $T(P) + s_0$ and rename all the fluents non corresponding to actions. Concatenating these CNFs we obtain a theory equivalent to eq. 3. It maybe in many cases smaller than the equivalent made through compilation to d–DNNF. These method can be useful in problems with a few initial states.

An immediate question is about non-deterministic planning. Could be interesting to add explicit support to non-deterministic action [CGT03]. We wonder if a policy mapping from states to actions, as used by Bonet and Geffner [BG00], can be obtained from a d–DNNF compilation of the propositional theory.

# 4 Collateral results

In this section, we comment about topics related with our objective of solve conformant planning problems, but are not critical to obtain such planner. Because d–DNNF is starting to be used, we want to obtain general lessons than can be applied in other contexts. Conformant planning can be formulated as a QBF of the form $\exists plan \forall s_0 \exists f_i \; \Phi$ [Rin99]. Some ideas working on QBF can work also in conformant planner and viceversa, at least for simple QBFs.

### 4.1 d–DNNF

The compiler to d–DNNF used [Dar04] needs a decomposition tree over the theory. In our both planners, we require the compiler to decompose CNF theories following an specific order, starting from the initial state, actions at time 0, fluents at time 1, until the goal. This order allows faster compilation that the orders the compiler can generate itself using approximated methods [Dar04]. A question is why this order allows compile so well our theories. Experiments reported in [Dar04] shows that our theories are probably the biggest compiled to d–DNNFs.

The complexity of compilation to d–DNNF depends on the width of the decomposition tree [Dar04]. Maybe in our case something else is helping to boost the performance. The order used follows the *stratification* of the propositional theory. The compiling algorithm depends exponentially on the width $w$ of the theory because in the worst case a clique of size $w$ will be fully represented. As consequence, there be an exponential number of nodes in the d–DNNF corresponding to this clique. But the compiler do not always create a node for any variable. It uses unit propagation to simplify the theory and avoid node creation. In our theories, in the serial setting, when the algorithm assign a action fluent to true in a time step, it cause all the other action fluents to be false. Maybe it is reducing the time required to compile. Obtaining a result of lower complexity for planning or stratified theories would be a important contribution to the use of d–DNNFs. Some ideas can come from structural complexity.

Using d–DNNFs in general, we need to be careful about repeating model counting or other queries on big theories. It can be too expensive for search algorithms.

### 4.2 QBF

Conformant planning can be formulated as QBF [Rin99]. We pointed out in [PG05] that the approach used for our sat-based planner is directly applicable to QBFs of the form: $\exists x \forall y \exists z \Phi$.

The no-search scheme we discussed in section 3 allow also to obtain all the solutions to any QBF as long as it can be compiled to a d–DNNF. Interestingly, recently [CMBLM05] reports a similar result for QBFs whose matrix is compiled in OBDD form: the QBF is tractable if the ordering used in the compilation is compatible with the ordering of the variables in the prefix.

## 5 Proposal

We want to focus on the problem of building a good optimal conformant planner using knowledge compilation, looking also at related problems than allow to gain insight into the relation with other problems or areas. We want a solver than can deal with problems near to classical planning as well as problems where uncertainty plays a more complex role.

One way to improve our current results in by focusing on the problems we cannot solve, for example the benchmarks used in the evaluation of CFF [BH04]. We want to probe doing projection through CNF renaming. Actually, our planner language only supports conditional effects. To deal with this problems

we need to add support to classical preconditions and be sure that spurious models are not generated. In that case, a modification of the propositional encoding may be needed [FG00].

To improve the performance of the model-counting-based planner, we want to support re-counting that allows to save time when counting the models under small variations of the instantiated variables. The work on watched data structures in SAT and QBF is relevant on this topic [Y.M94][GGN+03].

Other objective is to obtain appropriated relaxations than can improve the performance of the planner. We would like to adapt the heuristic of Rintanen [Rin04], who uses it within a search algorithm in belief state space. We do not search in the belief space, but in the plan space, so this idea is not directly applicable.

Appropriated relaxations or other ideas to boost the performance can come from the epistemic point of view. We want to look for approaches to conformant planning using epistemic logic, and to look for uses of propositional logic to encode more complex logics.

We want to study other options for propositional encodings of planning theories. We are going to look at Rintanen's [RHN04] and Geffner's [Gef94] works.

We do not discard that the incremental no-search scheme could be used effectively.

Finally, we want to understand what is doing the compilation of our planning theories to d–DNNFs. For this, we want to probe the lower complexity of compiling planning or stratified theories to d–DNNFs. We want to look at results of the behavior of algorithms with specific kinds of problems. Discover a way to detect stratified orders in other propositional theories and be able to compile them would give support to our hypothesis.

# References


[BG00] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. of AIPS-2000*, pages 52–61. AAAI Press, 2000.

[BH04] Ronen Brafman and Jörg Hoffmann. Conformant planning via heuristic forward search: A new approach. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, 2004.

[Bry92] R. E. Bryant. Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[CD97] M. Cadoli and F. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[CGP00] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.

[CGT03] Claudio Castellini, Enrico Giunchiglia, and Armando Tacchella. Sat-based planning in complex domains: concurrency, constraints and nondeterminism. *Artif. Intell.*, 147(1-2):85–117, 2003.

[CMBLM05] S. Coste-Marquis, D. Le Berre, F. Letombe, and P. Marquis. Propositional fragments for knowledge compilation and quantified boolean formulae. In *Proc. AAAI-05*, 2005. To appear.

[CR00] Alessandro Cimatti and Marco Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

[CRB04] A. Cimatti, M. Roveri, and P. Bertoli. Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*, 159:127–206, 2004.

[Dar02] A. Darwiche. On the tractable counting of theory models and its applications to belief revision and truth maintenance. *J. of Applied Non-Classical Logics*, 2002.

[Dar04] A. Darwiche. New advances in compiling cnf into decomposable negation normal form. In *Proc. ECAI 2004*, pages 328–332, 2004.

[DM02a] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[DM02b] A. Darwiche and P. Marquis. A knowledge compilation map. *J. of AI Research*, 17:229–264, 2002.

[FG00] Paolo Ferraris and Enrico Giunchiglia. Planning as satisfiability in nondeterministic domains. In *Proceedings AAAI-2000*, pages 748–753, 2000.

[Gef94] H. Geffner. Planning graphs and knowledge compilation. In D. Dubois, C. Welty, and M. Williams, editors, *Proceedings of the Fourth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-04)*, pages 662–672. AAAI Press, 1994.

[GGN+03] Ian P. Gent, Enrico Giunchiglia, Massimo Narizzano, Andrew G. D. Rowley, and Armando Tacchella. Watched data structures for qbf solvers. In *SAT*, pages 25–36, 2003.

[GT99] F. Giunchiglia and P. Traverso. Planning as model checking. In *Proceedings of ECP-99*. Springer, 1999.

[HD] J. Huang and A. Darwiche. Dpll with a trace: From sat to knowledge compilation. In *Proc. IJCAI 2005*. to appear.

[KS99] H. Kautz and B. Selman. Unifying SAT-based and Graph-based planning. In T. Dean, editor, *Proceedings IJCAI-99*, pages 318–327. Morgan Kaufmann, 1999.

[LR94] F. Lin and R. Reiter. Forget it! In *Working Notes, AAAI Fall Symposium on Relevance*, pages 154–159. American Association for Artificial Intelligence, 1994.

[PBDG05] H. Palacios, B. Bonet, A. Darwiche, and H. Geffner. Pruning conformant plans by counting models on compiled d-DNNF representations. In *Proc. of the 15th Int. Conf. on Planning and Scheduling (ICAPS-05)*. AAAI Press, 2005. To appear.

[PG05] H. Palacios and H. Geffner. Mapping conformant planning to sat through compilation and projection. In *Submitted to the First Workshop on Quantified Constraint Satisfaction Problems - CP-2005*, 2005.

[RHN04] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä. Parallel encodings of classical planning as satisfiability. In José Júlio Alferes and João Alexandre Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA'04)*, volume 3229 of *Lecture Notes in Computer Science*, pages 307–319, Lisbon, Portugal, September 2004. Springer-Verlag.

[Rin99] Jussi Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

[Rin04] Jussi Rintanen. Distance estimates for planning in the discrete belief space. In *Proc. AAAI-04*, pages 525–530, 2004.

[SK96] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[Tur02] Hudson Turner. Polynomial-length planning spans the polynomial hierarchy. In Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni, editors, *JELIA*, volume 2424 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2002.

[Y.M94] S. Malik Y.Mahajan, Z. Fu. ZChaff2004: An Efficient SAT Solver. *Selected Revised Papers Series: Lecture Notes in Computer Science*, 1994. to appear.