A First-Order Calculus Based on Propositional BDDs

Joachim Posegga **Klaus Schneider**

Universität Karlsruhe
Institut für Logik, Komplexität und Deduktionssysteme
Institut für Rechnerentwurf und Fehlertoleranz
Postfach 6980, 7500 Karlsruhe, FRG
Email: {posegga|schneide}@ira.uka.de

Dec 24, 1993

1 Introduction

Binary Decision Diagrams (BDDs) and variants of them are known in Computer Science since many decades and have been successfully applied to various domains. Especially experience in hardware verification (see e.g. (Brace *et al.*, 1990)) has shown that BDDs are well suited as an underlying datastructure for proving properties of propositional formulæ.

BDDs are usually used for representing decidable languages like propositional logic. In most cases reduced, ordered BDDs are used, which are a unique (canonical) normal form for Boolean functions: a fixed ordering on propositional variables is used to eliminate redundancy in the paths of BDDs; this is exploited to achieve a canonical representation. However, decidable logics are often not sufficiently expressive for certain applications and it is desirable to have means for applying BDDs to more powerful languages like first-order logic. We outline an elegant and simple way for achieving this. Before going into details, we will briefly remind the reader of previously published results on first-order BDDs (cf. (Goubault & Posegga, 1994)).

If we neglect orderings in BDDs, we get, from a theoretical point of view, nothing but a logical formula (possibly in a graphical representation) built with *if-then-else* as the only logical connective[1].

Several approaches for lifting non-ordered BDDs to first-order logic have been proposed: the earliest one known to the authors is due to Ewa Orłowska (1969): she describes a BDD-like calculus for a subset of first-order logic. This subset is decidable, but her method can be carried forward to the semi-decidable class of prenex normal form formulæ.

Orłowska's paper was written from a logician's point of few and understanding the relation to BDDs requires careful reading. An approach, where this is easier to see was proposed by Posegga (Posegga, 1993).

Other authors consider ordered BDDs for first-order logic; this cannot achieve canonicity, since a unique normal form is not computable for semi-decidable languages. However, it can still be a useful for avoiding redundancy in deduction. Billon (1991) describes an approach to representing quantifier-free formulæ of first-order logic by so-called *Typed Decision Graphs*, which are an extension of ordered BDDs. Independently of Billon, proposed method for representing the same class for formulæ with a variant of non-ordered BDDs. Another approach was proposed by Goubault (1993): he uses standard, ordered BDDs for representing formulæ in conjunctive normal form.

All these approaches to lifting first-order BDDs have the disadvantage that they cannot represent general formulæ: they do not provide means for explicitly representing quantifiers and can therefore be applied to quantifier-free formulæ, only. This as disadvantageous, as first-order formluæ cannot be treated literally if they contain quantifiers; we must apply some preprocessing for translating them into a form without quantifiers before representing them as a BDD. The overhead involved with this can also result in less efficient deduction[2]

A technique for representing quantified formulæ in non-ordered BDDs was proposed by Posegga (1993). Posegga observed that BDDs and semantic tableau are closely related calculi (Posegga, 1993,

[1]Such formulæ have already been considered in 1854 by George Boole (Boole, 1958); Alonzo Church showed about one century later that *if-then-else* is a primitive basis for propositional logic (Church, 1956, §24, pp. 129ff)

[2]Due to lack of space, we cannot discuss this in detail; the key observation is the following: for first-order deduction, some extension rule for universally quantified propositions is necessary. Such an extension rule must allow for using the quantified proposition arbitrarily (but finitely) often during deduction. Eliminating quantifiers, however, means extending the scope of quantifiers maximally; then, these extensions are also maximized, which can introduce redundant literals when applying such an extension rule during the proof search.

Chapter 6) and carried the quantifier rules in semantic tableaux forward to BDDs: nested BDDs correspond to quantified sub-formulæ, and an extension rule resembles the expansion of universally quantified formulæ in tableaux. Schneider et.al. (1993) carries these quantifier rules forward to ordered BDDs and applies the resulting calculus to hardware verification.

The approach presented in this paper uses ordered BDDs for arbitrarily quantified formulæ; we represent quantified subformulæ by nested BDDs as introduced in (Posegga, 1993), but contrary to Schneider et.al. (1993), the underlying BDD calculus remains unchanged. This is achieved by embedding reduced, propositional BDDs in a first-order meta-level: quantified first-order formulæ are represented by special propositional variables (atoms). Their semantics is hidden to the propositional BDD and implemented on the meta-level by several extension rules that modify a propositional BDD with such variables.

The major advantage of the approach presented here is that the mechanisms for handling propositional BDDs need not to be changed at all; we just add rules for manipulating BDDs, but these do not affect their propositional treatment. Thus, we can use a standard BDD-package and use it for first-order inference. The first-order part is controlled on a meta-level, and the propositional inference rules remain unchanged, as all first-order constructs are hidden to the algorithms dealing with the the propositional BDDs.

2 The Calculus

For the sake of simplicity we will only consider formulæ in Skolemized negation normal form with properly renamed variables[3]. We assume the existence of a function *bdd* which maps a propositional formula into a BDD. We also assume the existence of standard logical operations on BDDs like conjunctively and disjunctively combining two BDDs and negating a BDD[4]. These operations will be denoted by the functions $bdd_\wedge$ and $bdd_\vee$, respectively.

The basic idea of our calculus is to embed "standard" BDDs in a first-order calculus. We use the function *f2bdd* on a first-order formula, by assuming that a formula is always treated as a propositional one; to achieve this, we simply regard first-order atoms and quantified formulæ as propositional atoms[5]. This can be formalized by defining an injective mapping from the set of all first–order atoms to a set of propositional ones. The mapping does not treat variables and function symbols appearing in an atom in a special way, but simply regards an atom as a string.

Furthermore, we use a special set of propositional atoms $\gamma_1, \gamma_2, \ldots$, that will be used to denote the scope of quantifiers in first-order formulæ, together with the variable that is universally quantified. Nodes in a BDD that are labeled with such atoms will be called γ-nodes; if γ_i is a γ-node, then $\mathcal{G}_i$ will denote the BDD representing the scope of the i^{th} quantifier in the input formula, and $\mathcal{V}_i$ denotes the quantified variable.

Definition 2.1 (First-order BDDs).
A BDD for a first-oder formula F is defined in the following way:

$$f2bdd(F) = \begin{cases} bdd(F) & \text{if } F \text{ is a literal} \\ bdd_\wedge(f2bdd(A), f2bdd(B)) & F = A \wedge B \\ bdd_\vee(f2bdd(A), f2bdd(B)) & F = A \vee B \\ bdd(\gamma_i) & \text{if } F = \forall x\ \Phi, \\ & \text{where } \mathcal{G}_i = f2bdd(\Phi) \text{ and } \mathcal{V}_i = x \end{cases}$$

The first graph ("Initial") of Figure 1 is an example first-order BDD for the formula

$$(\forall x\ P(x) \wedge Q(x)) \wedge \neg P(a) \wedge \neg Q(b)$$

and the ordering $\gamma_i < Pa < Pb < Qa < Qb$. The γ-node at the root of the BDD has been replaced by a node holding the BDD for the quantified subformula. All atoms appear without brackets in order to denote that they are actually treated as propositional atoms.

[3]The calculus can be easily carried forward to general first-order formulæ.

[4]This corresponds to the standard functionality offered by existing implementations of BDD packages. The calculus will neither depend on a particular implementation, nor will it require special orderings, so we can ignore all these details.

[5]We will not use the notion "propositional variable" here to avoid confusion with variables in first-order formulæ.

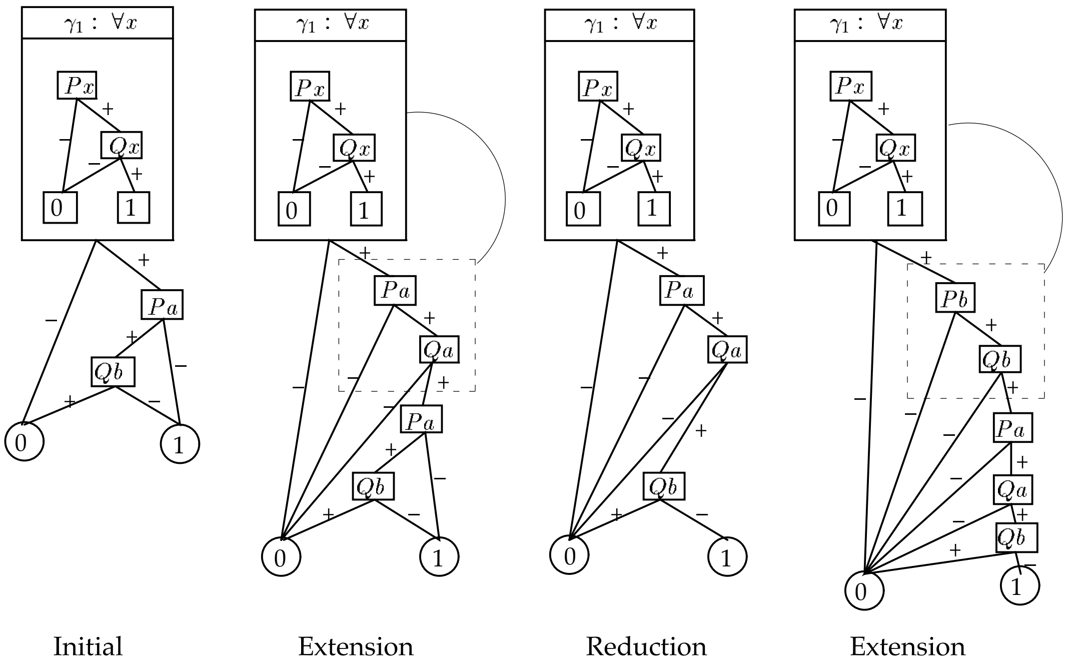

Figure 1. Examples of a First-order Proof with BDDs

It should be easy to see that the function *f2bdd* preserves satisfiability of a formula F. Note, that an important property of propositional BDDs is necessarily missing in this framework: *f2bdd* does not produce a unique normal form for a first-order formula. This is a consequence of the fact that first-order logic is only semi-decidable.

In order to prove a propositional formula, the conversion into a reduced ordered BDD is sufficient as each unsatisfiable formula has the normal form $\mathbf{0}$. In first order logic, however, this is not sufficient, as eventually copies of quantified formulæ are required. In order to obtain a proof procedure for first-order logic, we require an iteration step that alters *f2bdd*(F) until unsatisfiability can eventually be detected by reducing the extension to $\mathbf{0}$. The proposed procedure to achieve this is the following: a BDD containing a γ-node γ_i can be *extended* by replacing γ_i with $\gamma_i \wedge (\mathcal{G}_i\sigma)$, where σ is a substitution mapping $\mathcal{V}_i$ in $\mathcal{G}_i$ to a ground term. Usually such an operation is provided by the composition operation of BDD-packages, so there is no need to actually modify a mechanism for handling standard BDDs.

Choosing the γ-node and selecting an appropriate substitution is of course a crucial point which is essential for the efficiency of the proof search. In general, it is desirable to choose σ such that the BDD becomes as small as possible after the extension, but also criteria of *fairness* have to be observed to guarantee completeness of an implementation.

The second graph in Figure 1 shows such an extension with the substitution $\sigma = [x/a]$. A subsequent reduction results in the third graph of the figure. After a second extension with the same γ-node and $[x/b]$, the resulting BDD (the forth graph) reduces to the single node $\mathbf{0}$. This shows that the initial formula was unsatisfiable.

The correctness of extensions is again rather straightforward: an extension corresponds to replacing a formula $\forall x\ \Phi$ by $(\forall x\ \Phi) \wedge \Phi\sigma$, which is sound for any substitution σ. Completeness is less obvious, but given in the sense that there exists a finite sequence of extensions that result in an inconsistent BDD, if the initial formula was unsatisfiable (Herbrand's Theorem).

3 Conclusion

We have described a first-order proof procedure based on a standard, propositional BDD formalism. It works by keeping special nodes (propositional variables) in BDDs that represent BDDs of quantified subformulæ. Such "quantified" BDDs are subsequently inserted into the superior BDD until a sufficient number of distinct ground instances of them have been produced in order to yield a propositionally inconsistent BDD.

Our method can be elegantly implemented by "wrapping" a standard, propositional BDD package into a first-order environment controlling the proof search. All propositional deduction is left unchanged, so there is no need to adapt the behavior of the functions working on propositional BDDs. This shows how to lift ordered BDDs to(wards) first-order logic without loosing efficiency in the propositional case. This is advantageous for applications of BDDs, where propositional logic is often sufficient, but first-order constructs are useful in some cases.

From the "pure" first-order perspective , the approach has the disadvantage that no free variables can be kept in BDDs: all extensions of universally quantified sub-BDDs must be fully instantiated. Guessing these substitutions can indeed be difficult, and domain-specific heuristics should be used. One can set up a similar procedure without this restriction and integrate free variables and unification. This, however, requires either to re-design the algorithms working on propositional BDDs, or to include expensive operations on the first-order meta-level. In both cases, the fundamental principle of most BDD-based algorithms, that local changes do not have global side-effects, must be sacrificed.

References

Basin, D., Fronhöfer, B., Hähnle, R., Posegga, J., & Schwind, C. (eds). (1993). *Proc. 2nd Workshop on Theorem Proving with Analytic Tableaux and Related Methods*. Marseilles, France. Published by: Max-Planck-Institut für Informatik, Saarbrücken, Germany (technical report 213).

Billon, J.-P. (1991). *A New Approach of Theorem Proving for Non Clausal First Order Logic with Equality based on Generalized Shannon's Decomposition Principle*. Tech. rept. ORDA/DMA/91037. Bull Corporate Research Center, Paris, France.

Boole, G. (1958). *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. New York: Dover. (First Edition 1854).

Brace, K. S., Rudell, R. L., & Bryant, R. E. (1990). Efficient Implementation of a BDD Package. *Pages 40 – 45 of: Proc. 27th ACM/IEEE Design Automation Conference*. IEEE Press.

Church, A. (1956). *Introduction to Mathematical Logic*. Vol. 1. Princeton, New Jersey: Princeton University Press. Sixth printing 1970 .

Goubault, J. (1993). Syntax Independent Connections. *In: (Basin* et al., *1993)*.

Goubault, J., & Posegga, J. (1994). BDDs and Automated Deduction. *In: Proc. 8th International Symposium on Methodologies for Intelligent Systems*. Lecture Notes in Artificial Intelligence. Charlotte, NC: Springer Verlag.

Orłowska, E. (1969). Automatic Theorem Proving in a Certain Class of Formulae of Predicate Calculus. *Bull. de L'Acad. Pol. des Sci., Série des sci. math., astr. et phys.*, **XVII**(3), 117 – 119.

Posegga, J. (1993). *Deduktion mit Shannongraphen für Prädikatenlogik erster Stufe*. Sankt Augustin: Infix Verlag.

Schneider, K., Kumar, R., & Kropf, T. (1993). Hardware Verification using First Order BDDs. *In: Proc. CHDL*.