On CNF Encodings of Decision Diagrams

Ignasi Abío[1] and Graeme Gange[3]
Valentin Mayer-Eichberger[1,2] and Peter J. Stuckey[1,3]

[1]NICTA [2]University of New South Wales [3]University of Melbourne
[1]`<firstname>.<lastname>@nicta.com.au`, [3]`gkgange@unimelb.edu.au`

Abstract. Decisions diagrams such as Binary Decision Diagrams (BDDs), Multi-valued Decision Diagrams (MDDs) and Negation Normal Forms (NNFs) provide succinct ways of representing Boolean and other finite functions. Hence they provide a powerful tool for modelling complex constraints in discrete satisfaction and optimization problems. Generic propagators for these global constraints exist, but they are complex and hard to implement. An alternative approach to making use of them for solving is to encode them to CNF, using SAT style solving technology to implement them efficiently. This may also have advantages since it is naturally incremental and exposes intermediate literals which may well be useful as search decisions for solving the problem.
In this paper we explore different ways that we can map these constraints to CNF, and the different properties these mappings maintain. Surprisingly the most used encoding of BDDs does not maintain domain consistency in arbitrary BDDs. We also consider the strength of propagation with respect to the intermediate literals. We give experiments which compare the performance of the different encodings.

1 Introduction

Decisions diagrams such as Binary Decision Diagrams (BDDs), Multi Decision Diagrams (MDDs) and Negation Normal Forms (NNFs) provide succinct ways of representing Boolean and other finite functions. Hence they provide a powerful tool for modelling complex constraints in discrete satisfaction and optimization problems.

Constraint programming solvers include generic propagators for propagating constraints represented by BDDs [16], MDDs [8] and NNFs [15], since they are highly flexible, and hence useful in many different models. But these propagators are complex and hard to implement.

An alternative approach to making use of them for solving is to encode them to CNF, using SAT style solving technology to implement them efficiently. If the remainder of the problem is naturally modelled in CNF then this allows a SAT solver to tackle the problem.

A SAT encoding may also be preferable within a CP solver, as it avoids the need for implementing complex propagators, is naturally incremental, and exposes intermediate literals as candidates for search and learning. A good encoding is critical in lazy decomposition approaches [1], where a propagator that participates in many conflicts is replaced by a CNF decomposition during runtime.

In this paper we explore di�erent approaches for encoding decision diagrams to CNF. [1] The contributions of this paper are:

- An investigation of a large design space for encoding decision diagrams
- We clarify the picture of BDD/MDD/NNF encodings, analyse their propagation strength and correct some misunderstandings in the literature.
- We introduce an encoding of BDDs and MDDs where unit propagation implements propagation completeness.
- Experiments which compare the performance of the di�erent encodings.

2 Preliminaries

2.1 SAT Solving

We denote the Boolean value true by $\top$ and false by $\bot$.

Let $\mathcal{Y} = \{y_1, y_2, \ldots\}$ be a fixed set of propositional *variables.* If $y \in \mathcal{Y}$ then y and $\neg y$ are *positive* and *negative literals*, respectively. The *negation* of a literal l, written $\neg l$, denotes $\neg y$ if l is y, and y if l is $\neg y$. A *clause* is a disjunction of literals $\neg y_1 \vee \cdots \vee \neg y_p \vee y_{p+1} \vee \cdots \vee y_n$, sometimes written as $y_1 \wedge \cdots \wedge y_p \rightarrow y_{p+1} \vee \cdots \vee y_n$. A *CNF formula* F is a conjunction of clauses.

A set of literals A is *contradictory* if $\exists y. \{y, \neg y\} \subset A$. A (partial) *assignment* A is a set of literals which is not contradictory. A literal l is *true* in A if $l \in A$, is *false* in A if $\neg l \in A$, and is *undefined* in A otherwise. An *extension* of an assignment A is an assignment A' where $A' \supset A$. A *complete assignment* is an assignment with no undefined literals. Given a partial assignment A, a *completion* of A is an extension of A which is a complete assignment.

A complete assignment A satisfies formula ϕ if replacing each y in ϕ which is true in A with $\top$ and replacing each y in ϕ which is false in A with $\bot$ gives an expression which evaluates to $\top$. A partial assignment A satisfies formula ϕ, written $A \models \phi$ if every completion of A satisfies ϕ.

Systems that decide whether a CNF formula F has any model are called SAT solvers, and the main inference rule they implement is *unit propagation*: given a CNF F and an assignment A, find a clause in F such that all its literals are false in A except at most one, say l, which is undefined, add l to A and repeat the process until reaching a fix-point. See e.g. [21] for more details.

For some set of clauses C, we shall use $UP_C(A)$ to denote the set of literals inferred by unit propagation on C starting from assignment A. We will omit the C subscript when clear from the context. Note that $UP_C(A)$ may be contradictory, in which case unit propagation has detected unsatisfiability.

[1] A longer version of this paper including proofs of all Theorems can be found at people.eng.unimelb.edu.au/pstuckey/mddenc.pdf.

2.2 Propositional Encodings

Problems of interest rarely (if ever) begin in CNF form. Boolean formulae ϕ must be first converted into some equisatisfiable conjunction of clauses F_ϕ. The seminal work here is the Tseitin transformation [25], later refined by Plaisted and Greenbaum [22], which introduces a variable for each sub-formula and adds clauses to enforce the semantics of each connective.

While equisatisfiability is sufficient for correctness, the choice of decomposition can have a great impact on solver performance. A major consideration here is *propagation strength* – that is, given some partial assignment A and formula ϕ, what can be said of $UP_{F_\phi}(A)$.

There are a number of properties we may wish of F_ϕ.

- An encoding F_ϕ for a formula ϕ is *correct* if any complete assignment A on $vars(\phi)$ where $A \models \phi$, then A has an extension satisfying F_ϕ, and any complete assignment $A \models \neg\phi$ has no extension satisfying F_ϕ.
- An encoding F_ϕ for a formula ϕ *implements consistency* if for every assignment A over $vars(\phi)$ where $A \models \neg\phi$, then $UP_{F_\phi}(A)$ is contradictory.
- An encoding F_ϕ for a formula ϕ *implements domain consistency* when for each literal l over $vars(\phi)$, if $A \models \phi \to l$ then $l \in UP_{F_\phi}(A)$.
- An encoding F_ϕ for a formula ϕ *implements unit refutation completeness* [26] (also called *SLUR* [19]) when for assignment B over $vars(F_\phi)$ where $B \models \neg F_\phi$, then $UP_{F_\phi}(B)$ is contradictory.
- An encoding F_ϕ for a formula ϕ *implements propagation completeness* [6, 19] when for each literal l over $vars(F_\phi)$, $B \models F_\phi \to l$ then $l \in UP_{F_\phi}(B)$.

Another important consideration is the encoding size. In general, smaller encodings are more efficient than larger ones, if both have the same propagation strength.

2.3 At-most-one and Exactly-one Constraints

Given a set of literals $l_1, \ldots, l_n$, the *At-most-one* (AMO) constraint over these literals is defined as $l_1 + l_2 + \ldots + l_n \leq 1$.

There are several ways to encode AMO into SAT [14, 3, 7]. Here, we consider the ladder encoding. It introduces variables $\{a_i := l_1 \vee \ldots \vee l_i \mid 1 \leq i < n\}$ and clauses $\{a_i \to a_{i+1}, l_i \to a_i, l_{i+1} \to \neg a_i\}$. It is easy to see that this encoding is propagation complete.

Given a set of literals $l_1, \ldots, l_n$, the *Exactly-one* (EO) constraint over these literals is defined as $l_1 + l_2 + \ldots + l_n = 1$. Notice that

$$\mathrm{EO}(\{l_1, \ldots, l_n\}) = \mathrm{AMO}(\{l_1, \ldots, l_n\}) \wedge (l_1 \vee \ldots \vee l_n)$$

This defines a propagation complete encoding for EO given a propagation complete encoding of AMO.

2.4 Direct Encoding for Integer Variables

There are different methods for encoding integer variables into SAT (see for instance [27, 18]). In this paper we use the direct encoding.

Let x be an integer variable with domain $[a, b]$. The *direct encoding* introduces Boolean variables $[\![x = i]\!]$ for $a \leq i \leq b$. A variable $[\![x = i]\!]$ is true iff $x = i$. The encoding also introduces the constraint $\mathrm{EO}(\{[\![x = i]\!] \mid a \leq i \leq b\})$.

We will sometimes treat Boolean variables b as integers with domain [0,1].

We will implicitly assume that the direct encoding clauses $\mathrm{EO}(\{[\![x = i]\!] \mid a \leq i \leq b\})$ are part of any encoding of formula using integers x. We also assume all assignments A are closed under unit propagation of these clauses.

We extend the notion of satisfaction to formulae involving integer variables, as follows. A complete assignment A satisfies ϕ if replacing each Boolean variable as before, and each integer variable x_i by j if $[\![x_i = j]\!] \in A$ (since $A \models \mathrm{EO}(\{[\![x_i = j]\!] \mid a \leq j \leq b\})$ there must be exactly one) and evaluating the resulting ground expression gives $\top$. We extend the notation $A \models \phi$ as before.

2.5 Multi-valued Decision Diagrams

A directed acyclic graph $\mathcal{M}$ is called an *ordered Multi-valued Decision Diagram (MDD)* if it satisfies the following properties:

- It has two terminal nodes, namely $\mathcal{T}$ (true) and $\mathcal{F}$ (false).
- Each non-terminal node is labeled by an integer variable $\{x_1, x_2, \cdots, x_n\}$. This variable is called *selector variable*.
- Every node labeled by x_i has the same number of outgoing edges, namely $b_i - a_i + 1$, where $[a_i, b_i]$ is the domain of x_i.
- If an edge connects a node with a selector variable x_i and a node with a selector variable x_j, then $j > i$.

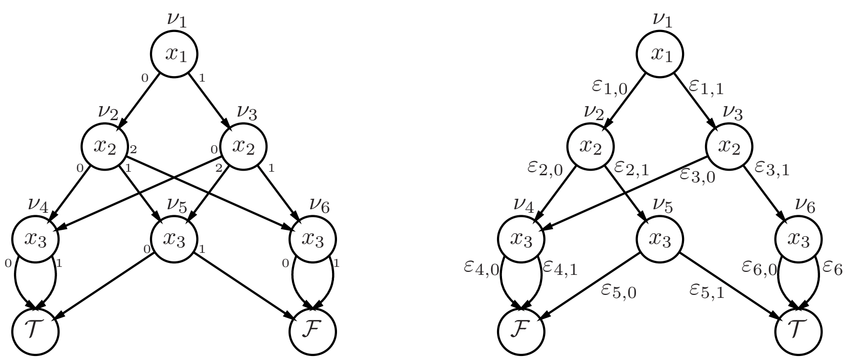
The MDD is *quasi-reduced* if no isomorphic subgraphs exist. It is *reduced* if, moreover, no nodes with only one child exist. A *long edge* is an edge connecting two nodes with selector variables x_i and x_j such that $j > i + 1$. In the following we only consider quasi-reduced ordered MDDs without long edges, and we just refer to them as MDDs for simplicity.[2] We refer to [24] for further details about MDDs.

Given an MDD $\mathcal{M}$ we use ρ to refer to its *root node*. Given a node $\nu \in \mathcal{M}$, we write $\mathrm{var}(\nu) = x_j$ when node ν is labelled by x_j. Given an edge $\varepsilon \in \mathcal{M}$, we write $\varepsilon = \mathrm{edge}(\nu, \mu, [\![x_i = j]\!])$ if ε joins the node ν and μ when $x_i = j$.

An MDD represents a formula over integer variables: a MDD node ν with selector x with domain $[a, b]$ and children $\nu_a, \nu_{a+1}, \ldots, \nu_b$ represents the formula ϕ_ν where

$$\phi_\nu \equiv \bigvee_{i \in [a,b]} x = i \wedge \phi_{\nu_i}$$

[2] Notice, however, that every result in this paper holds for non-reduced MDDs without long edges, and with some modifications of the rules the results also extend to non-reduced MDDs with long edges.



(a) (b)

Fig. 1. (a) MDD of $x_2 = 0 \vee (x_3 = 0 \wedge x_2 - x_1 = 1)$ and (b) BDD of $x_2 \wedge (x_1 \vee x_3)$

where ϕ_{ν_i} is the formula represented by node ν_i, and $\phi_{\mathcal{T}} = \top$ and $\phi_{\mathcal{F}} = \bot$.

Example 1 Let us consider the MDD encoding of $x_2 = 0 \vee (x_3 = 0 \wedge x_2 - x_1 = 1)$, with $x_1, x_3 \in \{0, 1\}$ and $x_2 \in \{0, 1, 2\}$, shown in Figure 1(a). In this case $\rho = \nu_1$, $\text{var}(\nu_3) = x_2$, and the rightmost edge from ν_3 is $\text{edge}(\nu_3, \nu_6, x_2 = 1)$. $\phi_{\nu_4} \leftrightarrow \top$, $\phi_{\nu_5} \leftrightarrow x_3 = 0$, $\phi_{\nu_6} \leftrightarrow \bot$, and hence $\phi_{\nu_2} \leftrightarrow (x_2 = 0 \wedge \top) \vee (x_2 = 1 \wedge x_3 = 0) \vee (x_2 = 2 \wedge \bot)$ or equivalently $\phi_{\nu_2} \leftrightarrow x_2 = 0 \vee (x_2 = 1 \wedge x_3 = 0)$. $\square$

A *binary decision diagram (BDD)* is an MDD with only Boolean variables. For a BDD $\mathcal{M}$ we can consider a non-terminal node ν as a triple (x, t, f) where there are two outgoing edges $\text{edge}(\nu, t, x)$ and $\text{edge}(\nu, f, \neg x)$. The BDD node ν represents the formula $\phi_\nu \equiv ITE(x, \phi_t, \phi_f)$ or equivalently $(x \wedge \phi_t) \vee (\neg x \wedge \phi_f)$.

2.6 Negation Normal Form Formulae

A *negation normal form* formula (NNF) is a rooted, directed acyclic graph (DAG) where each leaf node is labeled with x or $\neg x$ and each internal node is labeled with $\wedge$ or $\vee$ and can have arbitrarily many children.

NNFs are a more general form of decision diagram than BDDs, and can be exponentially more compact to represent the same formula [11]. We can use NNFs to express formulae over finite domain integer variables using the direct encoding.

But NNFs in general are too expressive, so usually we require some additional properties, such as:

decomposable An NNF $\mathcal{N}$ is *decomposable* if for each conjunction ϕ in $\mathcal{N}$, the conjuncts of ϕ do not share variables. That is, if $\phi_1, \ldots, \phi_n$ are the children of and-node ϕ, then $vars(\phi_i) \cap vars(\phi_j) = \emptyset$ for $i \neq j$.

deterministic An NNF $\mathcal{N}$ is *deterministic* if for each disjunction ϕ in $\mathcal{N}$, each two disjuncts of ϕ are logically contradictory. That is, if $\phi_1, \ldots, \phi_n$ are the children of or-node ϕ, then $\phi_i \wedge \phi_j \models \bot$ for $i \neq j$.

smooth An NNF $\mathcal{N}$ is *smooth* if for each disjunction ϕ in $\mathcal{N}$, each disjunct of ϕ mentions the same variables. That is, if $\phi_1, \ldots, \phi_n$ are the children of or-node ϕ, then $vars(\phi_i) = vars(\phi_j)$ for $i \neq j$.

3 Encoding MDDs

3.1 Encoding BDDs

The BDD encoding of MiniSat+ [13] is defined as follows: For each non-terminal BDD node $\nu = (x, t, f)$ we generate a Boolean variable ν which represents the truth value of the BDD rooted at ν.

For each non-terminal node $\nu = (x, t, f)$, we generate the following clauses:

B1 $t \wedge x \rightarrow \nu$.
B2 $\neg t \wedge x \rightarrow \neg\nu$.
B3 $f \wedge \neg x \rightarrow \nu$.
B4 $\neg f \wedge \neg x \rightarrow \neg\nu$.
B5 $t \wedge f \rightarrow \nu$.
B6 $\neg t \wedge \neg f \rightarrow \neg\nu$.

Define encoding $\mathsf{MiniSAT}$ as B1–B6, together with the terminal and root clauses: $\mathcal{T}$ (the true terminal is true), $\neg\mathcal{F}$ (the false terminal is false) and ρ (the root of the tree must be true).

Note while Een and Sorensen [13] refer to this as a Tseitin encoding, it is not since Tseitin [25] does not include an ITE constructor, so in the Tseitin encoding $ITE(x, t, f)$ needs to be encoded as $(x \wedge t) \vee (\neg x \wedge f)$.

The encoding contains $O(s)$ variables and clauses, where s is the size of the BDD.

Een and Sorensen [13] show that this encoding maintains domain consistency when used to encode (sorted) pseudo-Boolean constraints

Theorem 1 ([13]). *Unit propagation on the $\mathsf{MiniSAT}$ encoding for a BDD for pseudo-Boolean constraint $\sum_{i=1}^{n} c_i x_i \geq d$ maintains domain consistency, assuming the coefficients c_i are in non-increasing order.* □

This theorem does not hold without the ordering criterion. Consider the BDD encoding $x_1 + 2x_2 + x_3 \geq 3$ (or equivalently $x_2 \wedge (x_1 \vee x_3)$) shown in Figure 1(b). Any solution of the BDD requires x_2 is $\top$. Unit propagation on the $\mathsf{MiniSAT}$ encoding generates $\neg\mathcal{F}, \mathcal{T}, \nu_1, \neg\nu_4, \nu_6$ and nothing else.

Theorem 2. *Unit propagation on the clauses (B2), (B4), (B6), $\neg\mathcal{F}$, ρ for a BDD maintains consistency.* □

All in all, the encoding is compact (especially if only clauses (B2), (B4), (B6), $\neg\mathcal{F}$ and ρ are used), but the propagation strength is low.

3.2 Encodings MDDs with One Variable per Node

The first set of encodings for MDDs, used for example in [2], are generalizations of the MiniSat+ encoding. This is natural since they are also used to encode pseudo-Boolean and linear constraints.

For each node ν at level i, with children $\nu_{a_i}, \nu_{a_i+1}, \ldots, \nu_{b_i}$, where the domain of x_i is $[a_i, b_i]$.

M1 $\neg\nu_j \wedge [\![x_i = j]\!] \rightarrow \neg\nu$ (generalizes B2 and B4).
M2 $\nu_j \wedge [\![x_i = j]\!] \rightarrow \nu$ (generalizes B1 and B3).
M3 $\nu_{a_i} \wedge \nu_{a_i+1} \wedge \cdots \wedge \nu_{b_i} \rightarrow \nu$ (weakly generalizes B5).
M4 $\neg\nu_{a_i} \wedge \neg\nu_{a_i+1} \wedge \cdots \wedge \neg\nu_{b_i} \rightarrow \neg\nu$ (weakly generalizes B6).

With these clauses, we can define different encodings:

Minimal: Clauses M1, $\neg\mathcal{F}$, ρ.
GenMiniSAT: Clauses M1–M4, $\mathcal{T}$, $\neg\mathcal{F}$, ρ.

Minimal is very compact, but its propagation strength is low, moreover when the original variables are fixed it does not necessarily fix all the node variables, and hence does not preserve solution counts. GenMiniSAT is the natural generalization of the BDD encoding from [13] to MDDs. Again, it is not the Tseitin encoding [25] of the MDD. Both encodings use $O(s)$ variables and $O(sd)$ clauses, where s is the MDD size and d is the maximum domain size of variables x.

Proposition 1 *Let $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ be a complete assignment on variables x satisfying the MDD $\mathcal{M}$. Then, there exists a complete assignment $B \supset A$ over the variables x, ν satisfying clauses GenMiniSAT.* □

Proposition 2 *Let $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ be a complete assignment on variables x not satisfying the MDD $\mathcal{M}$, then clauses ρ and M1 propagate $\mathcal{F}$.* □

Corollary 1 *Minimal and GenMiniSAT are correct.* □

These two encodings, however, do not detect inconsistencies:

Example 2 Consider again the MDD of $x_2 = 0 \vee (x_3 = 0 \wedge x_2 - x_1 = 1)$, with $x_1, x_3 \in \{0, 1\}$ and $x_2 \in \{0, 1, 2\}$ shown in Figure 1(a).

After simplification, GenMiniSAT consists of the following clauses:

$$\begin{array}{llll}
\neg[\![x_1 = 0]\!] \vee \nu_2, & \neg[\![x_1 = 1]\!] \vee \nu_3, & \nu_2 \vee \nu_3, & \neg[\![x_2 = 0]\!] \vee \nu_2, \\
\neg\nu_4 \vee \neg[\![x_2 = 1]\!] \vee \nu_2, & \nu_4 \vee \neg[\![x_2 = 1]\!] \vee \neg\nu_2, & \neg[\![x_2 = 2]\!] \vee \neg\nu_2 & \neg[\![x_2 = 0]\!] \vee \nu_3, \\
\neg\nu_4 \vee \neg[\![x_2 = 2]\!] \vee \nu_3, & \nu_4 \vee \neg[\![x_2 = 2]\!] \vee \neg\nu_3, & \neg[\![x_2 = 1]\!] \vee \neg\nu_3 & \neg[\![x_3 = 0]\!] \vee \nu_4, \\
\neg[\![x_3 = 1]\!] \vee \neg\nu_4. & & &
\end{array}$$

Consider the partial assignment $A = \{\neg[\![x_2 = 0]\!], \neg[\![x_3 = 0]\!], [\![x_3 = 1]\!]\}$. It cannot be extended to a complete assignment satisfying the MDD. However, unit propagation does not fail.

The same happens with Minimal, since it is a subset of GenMiniSAT. □

3.3 Tseitin Encoding of an MDD

In this section we describe an alternative encodings for an MDD, the Tseitin encoding [25]. It detects inconsistencies with respect to the original variables but does not enforce domain consistency.

The Tseitin encoding introduce Boolean variables representing the formula of each edge. Let ν be a node at level i, with outgoing edges $\{\varepsilon_j \mid j \in J\}$. Let $\varepsilon = \text{edge}(\nu, \mu, [\![x_i = j]\!])$ be an edge of $\mathcal{M}$, then the Boolean variable ε encoding the edge represents the formula $[\![x_i = j]\!] \wedge \phi_\mu$.

The clauses of the Tseitin encoding are, for each node ν and edge ε

T1 $\nu \rightarrow \bigvee_j \varepsilon_j$.
T2 $\varepsilon \rightarrow \nu$.
T3 $\varepsilon \rightarrow \mu$.
T4 $\varepsilon \rightarrow [\![x_i = j]\!]$.
T5 $\mu \wedge [\![x_i = j]\!] \rightarrow \varepsilon$.

The Tseitin encoding, Tseitin, consists of clauses T1–T5, $\mathcal{T}$, $\neg\mathcal{F}$ and ρ. Therefore, it consists in $O(sd)$ variables and clauses, where s is the MDD size and d the maximum domain size of variables x.

Proposition 3 *Let $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ be a complete assignment on variables x satisfying the MDD $\mathcal{M}$. Then, there exists a complete assignment $B \supset A$ over the variables x, ν, ε satisfying clauses Tseitin.* □

Proposition 4 *Let A be a partial assignment on variables $\{x_i, x_{i+1}, \ldots, x_n\}$, and let ν be a node of $\mathcal{M}$ at level i. Assume that there is no completion A' of A satisfying the MDD rooted at ν. Then, unit propagation on clauses Tseitin and A enforces $\neg\nu$.* □

As a corollary, we can prove:

Theorem 3. *Tseitin is correct; i.e., given a complete assignment of the input variables, this encoding finds an inconsistency if and only if the assignment does not satisfy $\mathcal{M}$. Moreover, it implements consistency.* □

However, Tseitin does not preserve domain consistency.

Example 3 Let us consider the BDD of $x_2 \wedge (x_1 \vee x_3)$, shown in Figure 1(b). Tseitin, once simplified, generates the following clauses:

$$\begin{array}{llll}
\varepsilon_{1,0} \vee \varepsilon_{1,1}, & \neg\nu_2 \vee x_1 \vee \varepsilon_{1,0}, & \neg\varepsilon_{1,0} \vee \neg x_1, & \neg\varepsilon_{1,0} \vee \nu_2, \\
\neg\nu_3 \vee \neg x_1 \vee \varepsilon_{1,1}, & \neg\varepsilon_{1,1} \vee x_1, & \neg\varepsilon_{1,0} \vee \nu_3, & \neg\nu_2 \vee \varepsilon_{2,1}, \\
\neg\nu_5 \vee \neg x_2 \vee \varepsilon_{2,1}, & \neg\varepsilon_{2,1} \vee \nu_2, & \neg\varepsilon_{2,1} \vee x_2, & \neg\varepsilon_{2,1} \vee \nu_5, \\
\neg\nu_3 \vee \varepsilon_{3,1}, & \neg x_2 \vee \varepsilon_{3,1}, & \neg\varepsilon_{3,1} \vee \nu_3, & \neg\varepsilon_{3,1} \vee x_2, \\
\neg\nu_5 \vee \varepsilon_{5,1}, & \neg x_3 \vee \varepsilon_{5,1}, & \neg\varepsilon_{5,1} \vee \nu_5, & \neg\varepsilon_{5,1} \vee x_3.
\end{array}$$

Consider the partial assignment $A = \emptyset$. Notice that x_2 is not propagated even though that there is no solution of $\mathcal{M}$ with $\neg x_2$. Clause $x_2 \vee \varepsilon_{2,0} \vee \varepsilon_{3,0}$ would propagate x_2. □

Also, Tseitin does not implement unit refutation completeness:

Example 4 Consider the BDD of the constraint $\text{XOR}(x_1, x_2, x_3, x_4)$ shown in Figure 2. Node ν_2 represents the constraint $\text{XOR}(x_2, x_3, x_4)$, and node ν_3 represents $\neg\,\text{XOR}(x_2, x_3, x_4)$. It is clear, therefore, that the partial assignment $B = \{\nu_2, \nu_3\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. However, Tseitin does not find any conflict. □

3.4 Path-Based Encodings

Under the encodings described in Sections 3.2 and 3.3, the semantics of variables match the Boolean formula they represent – a node/edge variable is true (in a complete assignment) iff the corresponding formula is true.

In this section, we describe a set of *path-based* encodings. Like the Tseitin encoding these introduce one variable per node and per edge, but the interpretation of these variables is different. Under a path-based encoding, ν (or ε) is true iff the path from the root r to $\mathcal{T}$ defined by the selector variables passes through ν (resp. ε).

Unlike the previous encodings, the variables introduced here cannot be reused if a sub-formula occurs in multiple constraints. However, we shall see that this interpretation allows us to make much stronger inferences.

A related treatment of path-based encodings of the `regular` constraint to CNF can be found in Bacchus work in [4] and by Quimper and Walsh in [23] in context of the `grammar` constraint. Our study provides a complete analysis of such encodings for decision diagrams and introduces a novel encoding with stronger propagation properties.

We generate clauses for each node ν and connecting it to each of its outgoing edge ε_j and each of it incoming edges δ_j, as well as clauses for each edge $\varepsilon = \text{edge}(\nu, \mu, [\![x_i = j]\!])$.

P1 $\nu \wedge [\![x_i = j]\!] \rightarrow \varepsilon_j$.
P2 $\nu \rightarrow \bigvee_j \delta_j$ where $\nu \neq \rho$
P3 $[\![x_i = j]\!] \rightarrow \bigvee\{\varepsilon' \mid \varepsilon' = \text{edge}(\nu, \mu, [\![x_i = j]\!])$ for some $\nu, \mu \in \mathcal{M}\}$.
P4 $\text{EO}(\{\nu' \in \mathcal{M} \mid \text{Level}(\nu') = i\})$.

Clauses P1 enforce that a node on the path puts its outgoing edge on the path. Clauses P2 require each node on the path (except the root) has an incoming edge. Clauses P3 require that each integer value has an edge that supports it. Clauses P4 require that exactly one node on each level is $\top$.

With these clauses, we can define different encodings:

BasicPath: Clauses P1–P2, T1–T4, $\mathcal{T}$, $\neg\mathcal{F}$, ρ.
NNFPath: BasicPath and clauses P3.
LevelPath: BasicPath and clauses P4.
CompletePath: BasicPath and clauses P3–P4.

All the encodings require $O(sd)$ variables and clauses, where s is the MDD size and d the maximum domain size of variables x.

A complete assignment A over the variables x_i defines a path in $\mathcal{M}$ in the obvious way. This path is denoted by $\nu_1 = \rho$, ε_1, ν_2, ε_2, ... By definition of the MDD, the assignment is compatible with $\mathcal{M}$ if and only if $\nu_{n+1} = \mathcal{T}$.

A complete assignment B over variables x_i, ν, ε is compatible with $\mathcal{M}$ if

- $A := B \cap (\{[\![x_i = j]\!] \mid 1 \leq i \leq n, j \in [a_i, b_i]\}\{\neg[\![x_i = j]\!] \mid 1 \leq i \leq n, j \in [a_i, b_i]\})$ is compatible with $\mathcal{M}$.
- $\nu \in B$ iff $\nu = \nu_i$ for some i on the path defined by A.

– $\varepsilon \in B$ iff $\varepsilon = \varepsilon_i$ for some i on the path defined by A.

Proposition 5 *Given a complete assignment A on the variables x compatible with $\mathcal{M}$, there exists a complete assignment $B \supset A$ over the variables x, ν, ε satisfying clauses CompletePath.* □

Proposition 6 *Let A be a partial assignment on variables x. Let $UP(A)$ be the set of propagated literals with BasicPath. Let ν be a node of $\mathcal{M}$, and ε be an edge of $\mathcal{M}$. Then:*

– $\neg\nu \in UP(A)$ *if* $A \wedge \nu \models \neg\mathcal{M}$.
– $\neg\varepsilon \in UP(A)$ *if* $A \wedge \varepsilon \models \neg\mathcal{M}$. □

Let us explain the idea behind the proof. If ν has not been propagated to false, we can create a path from ρ to $\mathcal{T}$ passing through ν, where all the nodes of this path have not been propagated to false. This path will define a completion B satisfying $\mathcal{M}$ with $\nu \in B$.

To build this path, we start from ν. Since $\neg\nu \notin UP(A)$, ν must have a parent that has also not been propagated to false. This node, again, has a parent that has not been propagated to false, etc. That gives a path from ρ to ν. In the same way, ν has a child that has not been propagated to false, and this child has a child that has not been propagated to false, etc. That gives a path from ν to $\mathcal{T}$. Concatenating both paths, we obtain the desired path from ρ to $\mathcal{T}$.

Theorem 4. *BasicPath maintains consistency by unit propagation.* □

BasicPath, however, does not maintain domain consistency. For that we need clauses P3.

Example 5 Let us consider the BDD of $x_2 \wedge (x_1 \vee x_3)$, shown at Figure 1(b). BasicPath, once simplified, generates the following clauses:

$$\begin{array}{l}
x_1 \vee \varepsilon_{1,0},\quad \neg x_1 \vee \varepsilon_{1,1},\ \neg\nu_2 \vee x_2,\quad \neg\nu_3 \vee x_2,\\
\neg\nu_5 \vee x_3,\quad \varepsilon_{1,0} \vee \varepsilon_{1,1},\ \neg\nu_2 \vee \varepsilon_{2,1},\ \neg\nu_3 \vee \varepsilon_{3,1},\\
\neg\nu_5 \vee \varepsilon_{5,1},\ \neg\nu_2 \vee \varepsilon_{1,0},\ \neg\nu_3 \vee \varepsilon_{1,1},\ \neg\nu_5 \vee \varepsilon_{2,1},\\
\varepsilon_{3,1} \vee \varepsilon_{5,1}\quad \neg\varepsilon_{2,1} \vee \nu_2,\ \neg\varepsilon_{3,1} \vee \nu_3,\ \neg\varepsilon_{5,1} \vee \nu_5,\\
\neg\varepsilon_{1,0} \vee \nu_2,\ \neg\varepsilon_{1,1} \vee \nu_3,\ \neg\varepsilon_{2,1} \vee \nu_5,\ \neg\varepsilon_{1,0} \vee \neg x_1,\\
\neg\varepsilon_{1,1} \vee x_1,\ \neg\varepsilon_{2,1} \vee x_2,\ \neg\varepsilon_{3,1} \vee x_2,\ \neg\varepsilon_{5,1} \vee x_3.
\end{array}$$

Consider the partial assignment $A = \emptyset$. Then, unit propagation does not propagate x_2 even though that there is no solution of $\mathcal{M}$ with $\neg x_2$. Clause $x_2 \vee \varepsilon_{2,0} \vee \varepsilon_{3,0}$, from P3, would propagate x_2. □

As Corollary of Proposition 5 and Theorem 4, it follows that

Theorem 5. *Encodings BasicPath, NNFPath, LevelPath and CompletePath are correct; i.e., given a complete assignment of the input variables, these encodings find an inconsistency if and only if the assignment does not satisfy $\mathcal{M}$.* □

Theorem 6. *NNFPath maintains domain consistency by unit propagation.* □

NNFPath maintains domain consistency with respect to the original variables. However, since a SAT solver will not di�erentiate between original variables and auxiliary ones, partial assignments, in general, contain both type of variables. And, without clauses P4, the encodings are not propagation complete:

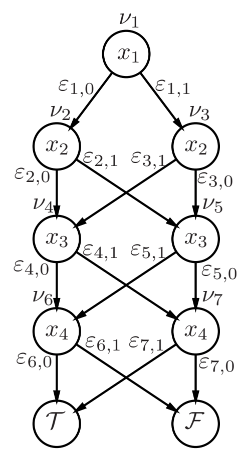


Fig. 2. BDD of XOR(x_1, x_2, x_3, x_4)

Example 6 Consider the MDD shown in Figure 2, representing the constraint XOR(x_1, x_2, x_3, x_4). Consider the partial assignment $B = \{\nu_4, \nu_5\}$. It is clear that B cannot be extended to a complete assignment satisfying $\mathcal{M}$, since no path can contain two nodes on the same level. However, NNFPath does not find any conflict. □

To maintain consistency with respect to all variables, clauses P4 are needed. In that case, we can generalize the previous results to assignments containing auxiliary variables:

Proposition 7 *Let B be a partial assignment on all the variables. Let $UP(B)$ be the set of propagated literals with LevelPath. Let ν be a node of $\mathcal{M}$, and ε be an edge of $\mathcal{M}$. Then:*

1. $\neg\nu \in UP(B)$ *if* $B \wedge \nu \models \neg\mathcal{M}$.
2. $\neg\varepsilon \in UP(B)$ *if* $B \wedge \varepsilon \models \neg\mathcal{M}$.
3. $\nu \in UP(B)$ *if* $B \wedge \neg\nu \models \neg\mathcal{M}$.
4. $\varepsilon \in UP(B)$ *if* $B \wedge \neg\varepsilon \models \neg\mathcal{M}$.

Theorem 7. *LevelPath is unit refutation complete.* □

LevelPath does not maintain domain consistency on all variables, though. Example 5 shows a counterexample. To obtain domain consistency we once more need the clauses P3.

Theorem 8. *CompletePath is propagation complete.* □

The path based encoding do have one weakness compared to the Tseitin encoding. Since they require only a single path throught the MDD, we cannot allow di erent MDD constraints that share a sub-MDD to reuse the same encoding, we need a di erent copy of the encoding for each constraint. This is not the case for Tseitin encdings where the node variable ν just represents the truth value of the sub-formula encoded by the MDD rooted at ν. To our knowledge this restriction is not very significant in the CP context. No such sharing exists in any of our benchmarks. The bulk of nodes in an MDD are in the middle and unlikely to be shared. Moreover, separating MDDs per constraint for translation allows us to use di erent variable orderings for each MDD and thus reduce the number of nodes required. On the other hand, if substantial sharing of nodes among the di erent MDDs happens then a Tseitin encoding could be beneficial, since it translates this sharing to the CNF level.

The table below shows the sizes and propagation strength of the di erent encodings. As before, s is the size of the MDD, d is the maximum domain size of variables x and n is the number of variables x. Notice that usually $n \ll s$.

	Minimal	GMinisat	Tseitin	BasicP	NNFP	LevelP	ComplP
Variables	s	s	$s(d+1)$	$s(d+1)$	$s(d+1)$	$s(d+2)$	$s(d+2)$
Clauses	sd	$s(2d+2)$	$s(4d+1)$	$s(4d+2)$	$s(4d+2)$ $+nd$	$s(4d+5)$	$s(4d+5)$ $+nd$
Consisistent	✘	✘	✔	✔	✔	✔	✔
Dom. Consis.	✘	✘	✘	✘	✔	✘	✔
Ref. Compl.	✘	✘	✘	✘	✘	✔	✔
Prop. Compl.	✘	✘	✘	✘	✘	✘	✔

4 Encoding NNFs

BDDs are a special case of NNFs and hence NNF encodings provide an alternate approach to encoding BDDs. There is an existing encoding for NNFs given by [20]. When applied correctly to MDDs it results in the NNFPath (hence the name). But care has to be taken in NNF encodings, without the right restrictions on the form of the NNF the encodings are incorrect!

An encoding of an NNF $\mathcal{N}$ to clauses is given by [20]. Each node ν is associated with a literal, also called ν. For leaf nodes the literal is just the label of the node. For non-leaf nodes the literal is a new Boolean variable. The clauses we make use of are

N1 $\nu \rightarrow \nu_1 \vee \cdots \vee \nu_k$ for each $\vee$-node ν with children $\nu_1, \ldots, \nu_k$
N2 $\nu \rightarrow \nu_i, 1 \leq i \leq k$ for each $\wedge$-node ν with children $\nu_1, \ldots, \nu_k$

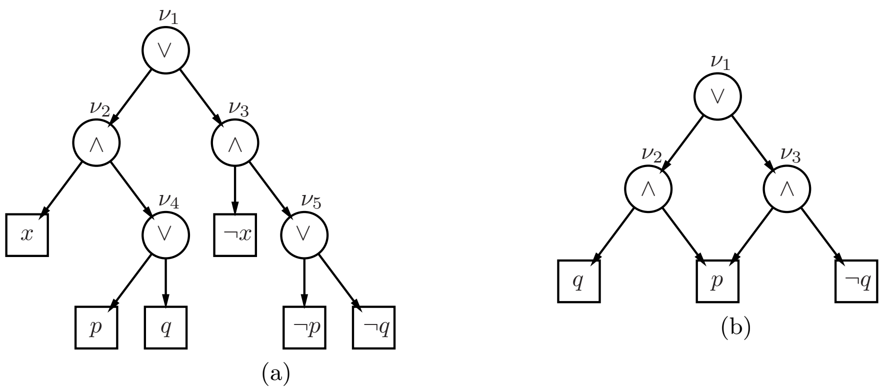


Fig. 3. NNF for formula (a) $(x \wedge (p \vee q)) \vee (\neg x \wedge (\neg p \vee \neg q))$ and (b) $(\neg q \wedge p) \vee (p \wedge q)$

N3 $\nu \rightarrow p_1 \vee \cdots \vee p_m$ for each node ν with incoming edges from nodes $p_1, \ldots, p_m$.

We consider two encodings: BaseNNF Clauses N1–N2 and ρ, and ExtNNF Clauses N1–N3 and ρ as defined in [20].

Theorem 9. *Given an NNF $\mathcal{N}$ then BaseNNF is a correct encoding.* □

Note that this *correctness* result *does not apply* to ExtNNF unless the NNF is smooth and decomposable. Jung [20] also claim that ExtNNF enforces domain consistency for decomposable NNFs, but this too is incorrect.

Example 7 The NNF shown in Figure 3(a) is decomposable, deterministic but not smooth (e.g. the two children of node ν_4 do not mention the same variables). The ExtNNF encoding is

$$\begin{array}{rlllll}
N1: & \nu_1 \rightarrow \nu_2 \vee \nu_3 & \nu_4 \rightarrow p \vee q & \nu_5 \rightarrow \neg p \vee \neg q & & \\
N2: & \nu_2 \rightarrow x & \nu_2 \rightarrow \nu_4 & \nu_3 \rightarrow \neg x & \nu_3 \rightarrow \nu_5 & \\
N3: & \nu_2 \rightarrow \nu_1 & \nu_3 \rightarrow \nu_1 & x \rightarrow \nu_2 & \nu_4 \rightarrow \nu_2 & \neg x \rightarrow \nu_3 \\
 & \nu_5 \rightarrow \nu_3 & p \rightarrow \nu_4 & q \rightarrow \nu_4 & \neg p \rightarrow \nu_5 & \neg q \rightarrow \nu_5 \\
\rho: & \nu_1 & & & &
\end{array}$$

Consider the assignment $A = \{x, \neg q\}$ unit propagation determines $\nu_1, \nu_2, \nu_4, p, \nu_5, \nu_3, \neg x$. and hence a contradiction. This is wrong since there is a model of the NNF $\{x, \neg q, p\}$. □

Example 8 Consider the smooth, decomposable and deterministic NNF for $(\neg q \wedge p) \vee (p \wedge q)$ shown in Figure 3(b). Then the clauses of ExtNNF are

$$\begin{array}{rlllll}
\rho: & \nu_1 & N1: & \nu_1 \rightarrow \nu_2 \vee \nu_3 & & \\
N2: & \nu_2 \rightarrow \neg q & \nu_2 \rightarrow p & \nu_3 \rightarrow p & \nu_3 \rightarrow q & \\
N3: & \nu_2 \rightarrow \nu_1 & \nu_3 \rightarrow \nu_1 & \neg q \rightarrow \nu_2 & p \rightarrow \nu_2 \vee \nu_3 & q \rightarrow \nu_3
\end{array}$$

Any model of the formula must make p true, but unit propagation on these clauses derives only ν_1. What is missing is information that $\neg p$ does not appear in the NNF. This means p *must hold!* □

Bench	Type	Search	#Inst		Prop	Minimal	GMinisat	Tseitin	BasicP	NNFP	LevelP	ComplP
			286	#sol	**282**	88	195	184	150	185	157	187
		VSIDS	78	com	**1.97**	-	5.33	27.81	58.73	14.09	42.51	24.86
	SAT		286	all	**23.82**	903.64	395.26	473.05	617.42	457.79	607.16	457.85
			179	#sol	132	143	151	**156**	**156**	108	**156**	104
Nurse		prog	80	com	**3.42**	-	6.19	6.61	18.39	54.63	29.96	50.86
			179	all	329.63	284.73	212.5	181.65	**171.95**	516.36	177.19	526.63
			46	#sol	32	29	**46**	27	31	33	32	32
	UNSAT	VSIDS	26	com	42.73	-	**8.09**	229.45	98.31	26.87	71.55	69.26
			46	all	402.57	626.02	**231.34**	631.03	450.35	380.4	413.69	437.38
			120	#sol	114	85	96	97	**116**	115	110	**116**
		VSIDS	78	com	109.8	-	166.51	161.91	**51.54**	88.07	68.91	117.44
			120	all	213.84	535.59	457.94	444.8	**174.65**	252.11	224.41	276.17
Shift	OPT		56	#sol	49	48	**56**	48	55	50	52	48
		prog	48	com	100.11	-	28.64	113.06	**24.52**	74.09	34.02	79.97
			56	all	257.02	240.44	**60.42**	268.34	161.76	232.17	176.28	239.97
			14	#sol	**14**	12	12	6	12	9	12	6
Pent	ALL	prog	6	com	**6.67**	-	8.21	18.27	14.57	16.02	8.8	15.36
			14	all	**279.43**	352.82	505.92	693.54	626.07	653.08	387.67	692.3

Table 1. Results on nurse rostering, shift scheduling and pentominoes.

To fix Jung's encoding we add the following clauses

N4 $\neg l$ for each literal l for $vars(\mathcal{N})$ which does not appear in $\mathcal{N}$.

We denote by FullNNF Clauses N1–N4 and ρ.

Theorem 10. *Given a smooth decomposable NNF $\mathcal{N}$ then FullNNF is a correct encoding.*

Theorem 11. *Given a smooth decomposable NNF $\mathcal{N}$, then unit propagation on FullNNF enforces domain consistency.* □

It follows that FullNNF is equivalent to NNFPath if applied to MDDs rewritten as NNF. To summarise the results in this section we provide the following table.

	BaseNNF	ExtNNF	FullNNF
Clauses	N1-N2	N1-N3	N1-N4
Correctness	Always	Smooth and Decomposable	Smooth and Decomposable
Domain Consistent	✘	✘	✔

5 Experiments

We show results on three benchmarks: nurse rostering, shift scheduling and pentominoes (Nurse, Shift and Pent).[3] The MDD encodings are implemented as

[3] Benchmarks are available from people.eng.unimelb.edu.au/pstuckey/mddenc.tar.gz.

eager translations of MDDs within the LCG solver Chuffed [10, 9] and compared with a native MDD propagator with learning [17]. We use SAT branching heuristics(VSIDS) and the programmed search as specified in the models (prog). We omit instances not solved by any solver using that search. For each model we show: (#sol) the number of instances solved (SAT and UNSAT for Nurse, to optimality for Shift, all solutions for Pent); (com) the mean solving time in seconds for all benchmarks solved by all solvers (except Minimal); and (all) the mean solving time of all benchmarks using timeout (1200s) for unsolved instances. The results on the encoding Minimal are omitted for *com* and for Pent since it does not preserve solution counting. Best results are in bold, and second best are underlined.

In case of satisfiable instances of Nurse the results show that encodings do not compete with the native propagator. This is not surprising as the search quickly finds the solutions without being disturbed by the complete CNF model generated by the eager encodings. For the UNSAT instances decompositions and their intermediate literals show their strength and beat the propagator. GenMiniSAT shows best performance for these UNSAT instances with VSIDS. The encodings also have an advantage over the propagator when programmed search is used, but it is unclear which one dominates.

For Shift the results show that when using activity based search and branching takes place on auxiliary variables, the path based approaches are generally superior.

The main advantage of the native propagator is that its explanations are built in a more deterministic fashion and hence tend to be more reusable. Furthermore, since the propagator only generates a fraction of the variables of the eager encoding, the search is less likely get trapped in an unfruitful search space using VSIDS. The difference in results on SAT and UNSAT instances of Nurse clearly indicate that a combination of the propagator and a lazy encoding as in [1] would be a strong approach.

6 Conclusion and Future Work

This paper resulted from discussions that uncovered our own misunderstanding of the strength of decision diagram encodings. We were surprised to discover that the usual BDD encoding is not domain consistent. In this paper we seek to remove this confusion, and demonstrate a wealth of different encoding possibilities, with different properties.

The experimental results show that there is unlikely to be one single best encoding for MDDs, and hence an important direction of future work is to determine when each encoding is best. Possibly a portfolio approach varying over encodings of each constraint is a fruitful and pragmatic technique to solve hard problems in practice.

Another interesting direction of future work is to determine a propagation complete encoding for NNFs. It appears the result may require restricting to Sentential Decision Diagrams [12] a form of NNF with a uniform V-tree.

The literature on CNF encodings focuses on consistencies wrt. primary variables of the constraint, whereas we have shown that consistency on auxiliary variables are worthwhile to look at. Our work concentrated on translations of decision diagrams and we would like to extend this research to other constraints like `linear` and `sequence`. State-of-the-art CNF encodings of `cardinality` are the next candidate for this investigation.

In case of theoretical results, an interesting direction is to establish lower bounds on the size of encodings implementing certain consistencies for concrete constraints. The strong relationship between CNF encodings and monotone circuits established in [5, 19] demonstrates a powerful tool for this purpose.

Acknowledgement NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., Stuckey, P.J.: To Encode or to Propagate? The Best Choice for Each Constraint in SAT. In: Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings. pp. 97–106 (2013)
2. Abío, I., Stuckey, P.J.: Encoding linear constraints into SAT. In: Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings. pp. 75–91 (2014)
3. Ansótegui, C., Manyà, F.: Mapping problems with finite-domain variables into problems with boolean variables. In: SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings (2004)
4. Bacchus, F.: GAC Via Unit Propagation. In: Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. pp. 133–147 (2007)
5. Bessiere, C., Katsirelos, G., Narodytska, N., Walsh, T.: Circuit complexity and decompositions of global constraints. In: IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 412–418 (2009)
6. Bordeaux, L., Marques-Silva, J.: Knowledge compilation with empowerment. In: SOFSEM 2012: Theory and Practice of Computer Science - 38th Conference on Current Trends in Theory and Practice of Computer Science, Špindlerův Mlýn, Czech Republic, January 21-27, 2012. Proceedings. pp. 612–624 (2012)
7. Chen, J.: A new SAT encoding of the at-most-one constraint. In: Proceedings of the Tenth International Workshop of Constraint Modelling and Reformulation (2010)
8. Cheng, K.C.K., Yap, R.H.C.: Maintaining generalized arc consistency on ad hoc r-ary constraints. In: Principles and Practice of Constraint Programming, 14th International Conference, CP 2008, Sydney, Australia, September 14-18, 2008. Proceedings. pp. 509–523 (2008)
9. Chu, G.: Chuffed, https://github.com/geoffchu/chuffed

10. Chu, G.G.: Improving combinatorial optimization. Ph.D. thesis, The University of Melbourne (2011)
11. Darwiche, A.: On the tractable counting of theory models and its application to truth maintenance and belief revision. Journal of Applied Non-Classical Logics 11(1-2), 11–34 (2001)
12. Darwiche, A.: Sdd: A new canonical representation of propositional knowledge bases. In: IJCAI. pp. 819–826 (2011)
13. Eén, N., Sörensson, N.: Translating Pseudo-Boolean Constraints into SAT. JSAT 2(1-4), 1–26 (2006)
14. Frisch, A.M., Peugniez, T.J., Doggett, A.J., Nightingale, P.: Solving non-boolean satisfiability problems with stochastic local search: A comparison of encodings. J. Autom. Reasoning 35(1-3), 143–179 (2005)
15. Gange, G., Stuckey, P.J., Hentenryck, P.V.: Explaining propagators for edge-valued decision diagrams. In: Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings. pp. 340–355 (2013)
16. Gange, G., Stuckey, P.J., Lagoon, V.: Fast set bounds propagation using a BDD-SAT hybrid. J. Artif. Intell. Res. (JAIR) 38, 307–338 (2010)
17. Gange, G., Stuckey, P.J., Szymanek, R.: MDD propagators with explanation. Constraints 16(4), 407–429 (2011)
18. Gent, I.P.: Arc consistency in SAT. In: Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002. pp. 121–125 (2002)
19. Gwynne, M., Kullmann, O.: A framework for good SAT translations, with applications to CNF representations of XOR constraints. CoRR abs/1406.7398 (2014), http://arxiv.org/abs/1406.7398
20. Jung, J.C., Barahona, P., Katsirelos, G., Walsh, T.: Two Encodings of DNNF Theories. In: ECAI'08 Workshop on Inference methods based on Graphical Structures of Knowledge (2008)
21. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(T). J. ACM 53(6), 937–977 (2006)
22. Plaisted, D.A., Greenbaum, S.: A Structure-Preserving Clause Form Translation. J. Symb. Comput. 2(3) (1986)
23. Quimper, C., Walsh, T.: Decomposing global grammar constraints. In: Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. pp. 590–604 (2007)
24. Srinivasan, A., Ham, T., Malik, S., Brayton, R.: Algorithms for discrete function manipulation. In: Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on. pp. 92–95 (1990)
25. Tseitin, G.: On the complexity of derivation in propositional calculus. Studies in Constructive Mathematics and Mathematical Logic Part 2, 115–125 (1968)
26. del Val, A.: Tractable databases: How to make propositional unit resolution complete through compilation. In: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994. pp. 551–561 (1994)
27. Walsh, T.: SAT v CSP. In: Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings. pp. 441–456 (2000)

Appendix A: Proofs

Theorem 2. Unit propagation on the clauses (B2), (B4), (B6), $\neg\mathcal{F}$, ρ for a BDD maintains consistency.

Proof. The proof is by induction. We show for any BDD node ν rooting a BDD of height n representing formula ϕ given an assignment A if $A \models \neg\phi$ the $\neg\nu \in UP(A)$.

The base case is trivial since $A \not\models \neg\mathcal{T}$ and $\neg\mathcal{F} \in UP(A)$. Given the result holds for $k < n$ we consider a BDD node $\nu = (x, t, f)$ representing $\phi = (x \wedge t) \vee (\neg x \wedge f)$ of height n.

Suppose $A \models \neg\phi$ then either $A \models x \wedge \neg\phi_t$ or $A \models \neg x \wedge \neg\phi_f$, or $A \models \neg\phi_t \wedge \neg\phi_f$. By induction either $\{x, \neg t\} \in UP(A)$ in which case $\neg\nu \in UP(A)$ by clauses (B2), or $\{\neg x, \neg f\} \in UP(A)$ in which case $\neg\nu \in UP(A)$ by clauses (B4), or $\{\neg t, \neg f\} \in UP(A)$ in which case $\neg\nu \in UP(A)$ by clauses (B6).

Applying the induction hypothesis to the root: if $A \models \neg\phi_\rho$ then $\neg\rho \in UP(A)$ and propagation detects the inconsitency. □

Proposition 1. Let $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ be a complete assignment on variables x satisfying the MDD $\mathcal{M}$. Then, there exists a complete assignment $B \supset A$ over the variables x, ν satisfying clauses $\mathsf{GenMiniSAT}$.

Proof. We can define B as follows: given $\nu \in \mathcal{M}$ at level i, $\nu \in B$ if and only if $\{[\![x_i = v_i]\!], [\![x_{i+1} = v_{i+1}]\!], \ldots, [\![x_n = v_n]\!]\}$ satisfies the MDD rooted at ν, i.e., the path from ν defined by $x_i = v_i, x_{i+1} = v_{i+1}, \ldots, x_n = v_n$ ends at $\mathcal{T}$; and $\neg\nu \in B$ otherwise.

Since A satisfies $\mathcal{M}$, $\rho \in B$ so ρ is satisfied. Obviously, $\mathcal{T}, \neg\mathcal{F} \in B$, so $\mathcal{T}$ and $\neg\mathcal{F}$ are also satisfied.

Given a node $\nu \in B$, clauses M2 and M3 are satisfied. For $j = v_i$, $\nu_j \in B$ by construction, so clause M1 is satisfied. If $j \neq v_i$, then $\neg[\![x_i = j]\!] \in B$, so clause M1 holds. Finally, since $\nu_{v_i} \in B$, clause M4 holds.

Given a node $\nu \notin B$, $\neg\nu \in B$, so clauses M1 and M4 are satisfied. For $j = v_i$, $\neg\nu_j \in B$ by construction, so clause M2 is satisfied. If $j \neq v_i$, then $\neg[\![x_i = j]\!] \in B$, so clause M2 holds. Finally, since $\neg\nu_{v_i} \in B$, clause M3 holds. □

Proposition 2. Let $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ be a complete assignment on variables x not satisfying the MDD $\mathcal{M}$, then clauses ρ and M1 propagate $\mathcal{F}$.

Proof. By induction on n. If $n = 0$, $\rho = \mathcal{F}$ so clause ρ propagates $\mathcal{F}$. Let us prove the general case.

If $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ does not satisy $\mathcal{M}$, then $\{[\![x_i = v_i]\!] \mid 2 \leq i \leq n\}$ does not satisfy ν, where ν is the v_1-th child of ρ. Clause ρ and M1 propagate ν. By induction hypothesis, ν and M1 propagate $\mathcal{F}$. □

Proposition 3. Let $A = \{[\![x_i = v_i]\!] \mid 1 \leq i \leq n\}$ be a complete assignment on variables x satisfying the MDD $\mathcal{M}$. Then, there exists a complete assignment $B \supset A$ over the variables x, ν, ε satisfying clauses $\mathsf{Tseitin}$.

Proof. We can define B as follows: given $\nu \in \mathcal{M}$ at level i, $\nu \in B$ if and only if $\{[\![x_i = v_i]\!], [\![x_{i+1} = v_{i+1}]\!], \ldots, [\![x_n = v_n]\!]\}$ satisfies the MDD rooted at ν, i.e., the path from ν defined by $x_i = v_i, x_{i+1} = v_{i+1}, \ldots, x_n = v_n$ ends at $\mathcal{T}$; and $\neg\nu \in B$ otherwise. Given $\varepsilon = \mathrm{edge}(\nu, \nu_j, [\![x_i = j]\!])$, $\varepsilon \in B$ if $[\![x_i = j]\!], \nu_j \in B$; and $\neg\varepsilon \in B$ otherwise.

Since A satisfies $\mathcal{M}$, $\rho \in B$ so ρ is satisfied. Obviously, $\mathcal{T}, \neg\mathcal{F} \in B$, so $\mathcal{T}$ and $\neg\mathcal{F}$ are also satisfied.

Clauses T3, T4 and T5 are satisfied by construction of B: $\varepsilon \in B$ if and only if $[\![x_i = j]\!], \nu_j \in B$.

If $\nu \in B$, then $\nu_{v_i} \in B$. Since $[\![x_i = v_i]\!] \in B$, $\mathrm{edge}(\nu, \nu_{v_i}, [\![x_i = v_i]\!]) \in B$. Therefore, clause T1 is satisfied.

If $\varepsilon = \mathrm{edge}(\nu, \nu_j, [\![x_i = j]\!]) \in B$, then $\nu_j \in B$ and $[\![x_i = j]\!] \in B$, this is, $j = v_i$. If the path defined by A goes from ν_j to $\mathcal{T}$, then it goes from ν to $\mathcal{T}$, so $\nu \in B$. Therefore, clause T2 is satisfied. $\square$

Proposition 4. Let A be a partial assignment on variables $\{x_i, x_{i+1}, \ldots, x_n\}$, and let ν be a node of $\mathcal{M}$ at level i. Assume that there is no completion A' of A satisfying the MDD rooted at ν. Then, unit propagation on clauses $\mathsf{Tseitin}$ and A enforces $\neg\nu$.

Proof. By induction on $n + 1 - i$. If $i = n + 1$, $\nu = \mathcal{F}$ and clause $\neg\mathcal{F}$ propagates $\neg\nu$. Let us prove the general case.

For every j in the domain of x_i, let ν_j be the j-th child of ν and $\varepsilon = \mathrm{edge}(\nu, \nu_j, [\![x_i = j]\!])$.

If $[\![x_i = j]\!] \in A$, since A has no completion satisfying ν, then there is no completion of A satisfying ν_j. By induction hypothesis, $\neg\nu_j$ is propagated. Therefore, by (T3), $\neg\varepsilon_j$ is propagated. For every $j' \neq j$, $\neg[\![x_i = j']\!] \in A$, so clause T4 propagates $\neg\varepsilon_{j'}$. Therefore, clause T1 propagates $\neg\nu$.

Assume now that $[\![x_i = j]\!] \notin A$ for any j. Then, for every j, there is no completion of $A \cup \{[\![x_i = j]\!]\}$ satisfying ν; so there is no completion of A satisfying ν_j. By induction hypothesis, $\neg\nu_j$ is propagated, so clause T3 propagates $\neg\varepsilon_j$. Therefore, $\neg\nu$ is propagated by (T1). $\square$

Proposition 5. Given a complete assignment A on the variables x compatible with $\mathcal{M}$, there exists a complete assignment $B \supset A$ over the variables x, ν, ε satisfying clauses $\mathsf{CompletePath}$.

Proof. A defines a path in $\mathcal{M}$. Let $\nu_1 = \rho$, $\varepsilon_1 = \mathrm{edge}(\nu_1, \nu_2, [\![x_1 = j_1]\!])$, ν_2, $\varepsilon_2 = \mathrm{edge}(\nu_2, \nu_3, [\![x_2 = j_2]\!])$, …, $\nu_{n+1} = \mathcal{T}$ be that path.

We define B as

$$B := A \cup \{\nu_i, \varepsilon_i\} \cup \{\neg\nu \mid \nu \neq \nu_i\} \cup \{\neg\varepsilon \mid \varepsilon \neq \varepsilon_i\}.$$

B is obviously a complete assignment, and $B \supset A$. We only have to show that B satisfies all the clauses of $\mathsf{CompletePath}$.

Clause ρ is satisfied since $\nu_1 = \rho$. Clause $\mathcal{T}$ is satisfied since $\nu_{n+1} = \mathcal{T}$. Clause $\neg\mathcal{F}$ is satisfied since the path does not contain $\mathcal{F}$.

It is easy to check that clauses P1, P2 and T1 are satisfied: they are obviously true if $\nu \neq \nu_i$, and, by construction of the path, they are true if $\nu = \nu_i$. The same happens with clauses T2–T4.

P4 holds since the path contains exactly one node on each level. P3 obviously holds for $[\![x_i = j]\!] \notin A$. For $[\![x_i = j]\!] \in A$, ε_i is true, so P3 also holds. □

Proposition 6. Let A be a partial assignment on variables x. Let $UP(A)$ be the set of propagated literals with $\mathsf{BasicPath}$. Let ν be a node of $\mathcal{M}$, and ε be an edge of $\mathcal{M}$. Then:

- $\neg\nu \in UP(A)$ if $A \wedge \nu \models \neg\mathcal{M}$.
- $\neg\varepsilon \in UP(A)$ if $A \wedge \varepsilon \models \neg\mathcal{M}$.

Proof. – Let us assume that $\neg\nu \notin UP(A)$, and $\mathrm{Level}(\nu) = i$. Let us call $\nu_i := \nu$. Since $\neg\nu_i \notin UP(A)$, either $i = 1$ or, by (P2), ν_i has an incoming edge ε_{i-1} such that $\neg\varepsilon_{i-1} \notin UP(A)$. Let us define ν_{i-1} and j_{i-1} such as

$$\varepsilon_{i-1} = \mathrm{edge}(\nu_{i-1}, \nu_i, [\![x_{i-1} = j_{i-1}]\!]).$$

Therefore, since $\neg\varepsilon_{i-1} \notin UP(A)$, by clause T4 $\neg[\![x_{i-1} = j_{i-1}]\!] \notin UP(A)$ and by (T2) $\neg\nu_{i-1} \notin UP(A)$.
Again, since $\neg\nu_{i-1} \notin UP(A)$, either $i-1 = 1$ or there exists an incoming edge $\varepsilon_{i-2} = \mathrm{edge}(\nu_{i-2}, \nu_{i-1}, [\![x_{i-2} = j_{i-2}]\!])$ with $\neg\varepsilon_{i-2}, \neg[\![x_{i-2} = j_{i-2}]\!], \neg\nu_{i-2} \notin UP(A)$. In the same way, we can define $\nu_{i-3}, \varepsilon_{i-3}, \nu_{i-4}, \varepsilon_{i-4}, \ldots, \nu_1$. Since $\mathrm{Level}(\nu_1) = 1$, $\nu_1 = \rho$.
Similarly, since $\neg\nu_i \notin UP(A)$, either $i = n+1$ or, by (T1), ν_i has an outgoing edge ε_i such that $\neg\varepsilon_i \notin UP(A)$. As before, we define ν_{i+1} and j_i such that $\varepsilon_i = \mathrm{edge}(\nu_i, \nu_{i+1}, [\![x_i = j_i]\!])$. By clauses (T3) and (T4), $\neg[\![x_i = j_i]\!], \neg\nu_{i+1} \notin UP(A)$. Therefore, we can repeat the process: either $i + 1 = n + 1$ or there exists an outgoing edge $\varepsilon_{i+1} = \mathrm{edge}(\nu_{i+1}, \nu_{i+2}, [\![x_{i+1} = j_{i+1}]\!])$ with $\neg\varepsilon_{i+1}, \neg[\![x_{i+1} = j_{i+1}]\!], \neg\nu_{i+2} \notin UP(A)$. Again, we repeat the process and define $\varepsilon_{i+2}, \nu_{i+3}, \ldots, \nu_{n+1}$. Since $\mathrm{Level}(\nu_{n+1}) = n + 1$, $\nu_{n+1} \in \{\mathcal{T}, \mathcal{F}\}$. However, $\neg\mathcal{F} \in UP(A)$ (since $\neg\mathcal{F}$ is a clause of $\mathsf{BasicPath}$), and $\neg\nu_{n+1} \notin UP(A)$ by construction. Therefore, $\nu_{n+1} = \mathcal{T}$.
Therefore, we have constructed a path $\nu_1 = \rho, \varepsilon_1, \nu_2, \ldots, \nu_{n+1} = \mathcal{T}$ such that, for all $k \in \{1, \ldots, n\}$:

- $\varepsilon_k = \mathrm{edge}(\nu_k, \nu_{k+1}, [\![x_k = j_k]\!])$.
- $\neg\varepsilon_k \notin UP(A)$.
- $\neg\nu_k \notin UP(A)$.
- $\neg[\![x_k = j_k]\!] \notin UP(A)$.

Let B be a complete assignment over the variables of $\mathsf{BasicPath}$, with:

$$B := \{[\![x_k = j_k]\!], \varepsilon_k, \nu_k \mid 1 \le k \le n\} \cup \{\neg l \mid l \neq [\![x_k = j_k]\!], \varepsilon_k, \nu_k\}.$$

Obviously, $B \models \mathcal{M}$ and $\nu \in B$. Therefore, we only have to prove that $B \supset A$.
Assume that $B \not\supset A$. This means that either there exists $[\![x_k = j]\!] \in A \setminus B$; or there exists $[\![x_k = j]\!]$ with $\neg[\![x_k = j]\!] \in A \setminus B$.

If $[\![x_k = j]\!] \in A \setminus B$, then $j \neq j_k$. However, since A is closed under unit propagation of $\mathrm{EO}\{[\![x_k = j]\!] \mid a_k \leq j \leq b_k\}$, this would mean that $\neg[\![x_k = j_k]\!] \in A$. That contradicts that $\neg[\![x_k = j_k]\!] \notin UP(A)$.
If $\neg[\![x_k = j]\!] \in A \setminus B$, then $j = j_k$. But that means that $\neg[\![x_k = j_k]\!] \in A$, which is again a contradiction.

- Let $\varepsilon = \mathrm{edge}(\nu, \mu, [\![x_i = j]\!])$, and assume $A \wedge \varepsilon \models \neg\mathcal{M}$. This means that there is no completation of A satisfying $\mathcal{M}$ with ε in its path.
First, let us assume that $\neg[\![x_i = j]\!] \notin A$ and there are A_1, A_2 completions of A, satisfying $\mathcal{M}$, with ν in the path of A_1 and μ in the path of A_2.
Let us define
$$A_3 := \Big(A_1 \cap \{[\![x_{i'} = j']\!], i' < i\}\Big) \cup \{[\![x_i = j]\!]\} \cup \Big(A_2 \cap \{[\![x_{i'} = j']\!], i' > i\}\Big),$$
and $A' := A_3 \cup \{\neg[\![x_{i'} = j']\!] \mid [\![x_{i'} = j']\!] \notin A_3\}$. It is easy to see that A' is a completion of A satisfying $\mathcal{M}$ with ε in the path defined by it.
Therefore, either $\neg[\![x_i = j]\!] \in A$, so $\neg\varepsilon$ is propagated by (T4); there is no A_1, so, by the first claim of this Proposition, $\neg\nu$ is propagated and $\neg\varepsilon$ is propagated by (T2); or there is no A_2, so $\neg\mu$ is propagated and (T3) propagates $\neg\varepsilon$.

$\square$

Theorem 4. BasicPath maintains consistency by unit propagation.

Proof. Given a partial assignment A that cannot be extended into a complete assignment satisfying $\mathcal{M}$, by the previous Proposition, $\neg\rho$ is propagated. Therefore, since ρ is a clause of BasicPath, a conflict is found. $\square$

Theorem 6. NNFPath maintains domain consistency by unit propagation.

Proof.
- Let A be a partial assignment such that $A \cup \{[\![x_i = j]\!]\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. Then, given any $\varepsilon = \mathrm{edge}(\nu, \mu, [\![x_i = j]\!])$, $A \wedge \varepsilon \models \neg\mathcal{M}$. By Proposition 6, clauses from BasicPath propagate $\neg\varepsilon$.
Therefore, clause P3 propagates $\neg[\![x_i = j]\!]$.
- Let A be a partial assignment such that $A \cup \{\neg[\![x_i = j]\!]\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. Due to constraint $\mathrm{EO}\{[\![x_i = j']\!] \mid a_i \leq j' \leq b_i\}$, for each $j' \neq j$, $A \cup \{x_i^{j'}\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. Therefore, as previously seen, $\neg[\![x_i = j']\!]$ is propagated. Therefore, $\mathrm{EO}\{[\![x_i = j']\!] \mid a_i \leq j' \leq b_i\}$ propagates $[\![x_i = j]\!]$.

$\square$

Proposition 7. Let B be a partial assignment on all the variables. Let $UP(B)$ be the set of propagated literals with LevelPath. Let ν be a node of $\mathcal{M}$, and ε be an edge of $\mathcal{M}$. Then:

1. $\neg\nu \in UP(B)$ if $B \wedge \nu \models \neg\mathcal{M}$.
2. $\neg\varepsilon \in UP(B)$ if $B \wedge \varepsilon \models \neg\mathcal{M}$.

3. $\nu \in UP(B)$ if $B \wedge \neg\nu \models \neg\mathcal{M}$.
4. $\varepsilon \in UP(B)$ if $B \wedge \neg\varepsilon \models \neg\mathcal{M}$.

Proof. 1. Let us assume that $\neg\nu \notin UP(B)$, and $\text{Level}(\nu) = i$. Let us call $\nu_i := \nu$. Using the same argument as in the proof of Proposition 6, we can build a path $\nu_1 = \rho, \varepsilon_1, \nu_2, \ldots, \nu_{n+1} = \mathcal{T}$ such that:
   - $\varepsilon_k = \text{edge}(\nu_k, \nu_{k+1}, [\![x_k = j_k]\!])$.
   - $\neg\varepsilon_k \notin UP(B)$.
   - $\neg\nu_k \notin UP(B)$.
   - $\neg[\![x_k = j_k]\!] \notin UP(B)$.

   Let us define B' as follows:

$$B' := \{[\![x_k = j_k]\!], \nu_k, \varepsilon_k \mid 1 \le k \le n\} \cup \{\nu_{n+1}\} \cup \\ \cup \{\neg[\![x_k = j]\!] \mid j \ne j_k\} \cup \{\neg\eta \mid \eta \ne \nu_k\} \cup \{\neg\varepsilon \mid \varepsilon \ne \varepsilon_k\}.$$

   We just have to prove that $B' \supset B$, $B' \models \mathcal{M}$ and $\nu \in B'$.
   The path defined by B' is obviously $\nu_1 = \rho, \varepsilon_1, \nu_2, \ldots, \nu_{n+1} = \mathcal{T}$; therefore, B' satisfies $\mathcal{M}$ and $\nu \in B'$. Therefore, we only have to prove that B' is a completion of B, this is, $B' \supset B$.
   Assume that $B' \not\supset B$. Then, one of the following cases holds:
   - There exists $[\![x_k = j]\!] \in B \setminus B'$: In this case, $j \ne j_k$. However, since B is closed under unit propagation of $\text{EO}\{[\![x_k = j]\!] \mid a_k \le j \le b_k\}$, this means that $\neg[\![x_k = j_k]\!] \in B$. That contradicts that $\neg[\![x_k = j_k]\!] \notin UP(B)$.
   - There exists $[\![x_k = j]\!]$ with $\neg[\![x_k = j]\!] \in B \setminus B'$: In this case, $j = k_j$. But that means that $\neg[\![x_k = j_k]\!] \in B \subset UP(B)$, which is again a contradiction.
   - There exists $\mu \in B \setminus B'$. Let k be the level of μ. Since $\mu \notin B'$, $\mu \ne \nu_k$. Therefore, since $\mu \in B$, by (P4), $\neg\nu_k \in UP(B)$, which is a contradiction.
   - There exists μ with $\neg\mu \in B \setminus B'$. Let k be the level of μ. Since $\neg\mu \notin B'$, $\mu = \nu_k$. This means $\neg\nu_k \in B \subset UP(B)$, which is a contradiction.
   - There exists $\varepsilon \in B \setminus B'$. Let us define η, μ, k, j such that $\varepsilon = \text{edge}(\eta, \mu, [\![x_k = j]\!])$. Since $\varepsilon \in B$, by (T2) $\eta \in UP(B)$. Since $\neg\nu_k \notin UP(B)$, by (P4) $\eta = \nu_k$. Similarly, since $\varepsilon \in B$, by T4 $[\![x_k = j]\!] \in UP(B)$. Since $\neg[\![x_k = j_k]\!] \notin UP(B)$ and $UP(B)$ is closed under unit propagation of $\text{EO}\{[\![x_k = j']\!] \mid a_k \le j' \le b_k\}$, we can deduce $[\![x_k = j]\!] = [\![x_k = j_k]\!]$.
     Since $\eta = \nu_k$ and $[\![x_k = j]\!] = [\![x_k = j_k]\!]$, $\varepsilon = \varepsilon_k$. This contradicts that $\varepsilon \in B \setminus B'$.
   - There exists ε with $\neg\varepsilon \in B \setminus B'$. Let k be the level of ε. Since $\neg\varepsilon \notin B'$, $\varepsilon = \varepsilon_k$. This means $\neg\varepsilon_k \in B \subset UP(B)$, which is a contradiction.

   Therefore, B' is a completion of B, it satisfies $\mathcal{M}$ and $\nu \in B'$.
2. The proof is identical as the proof of the second claim of Proposition 6.
3. If $B \wedge \neg\nu \models \neg\mathcal{M}$, then for every ν' with $\text{Level}(\nu) = \text{Level}(\nu')$, $B \wedge \nu' \models \neg\mathcal{M}$. Therefore, $\neg\nu' \in UP(B)$. Therefore, by (P4), $\nu \in UP(B)$.
4. Let $\varepsilon = \text{edge}(\nu, \mu, [\![x_i = j]\!])$. If every completion of B satisfying $\mathcal{M}$ contains ε, then they all also contains ν. Therefore, $\nu \in UP(B)$. Moreover, they cannot contain the edges $\text{edge}(\nu, \mu, [\![x_i = j']\!])$ with $j' \ne j$, so $\neg\,\text{edge}(\nu, \mu, [\![x_i = j']\!]) \in UP(B)$. Therefore, by (T1), $\varepsilon \in UP(B)$.

$\square$

Theorem 7. $\mathsf{LevelPath}$ is unit refutation complete.

Proof. Given a partial assignment B that cannot be extended into a complete assignment satisfying $\mathcal{M}$, by the previous Proposition, $\neg\rho$ is propagated. $\square$

Theorem 8. $\mathsf{CompletePath}$ is propagation complete.

Proof.
- Let B be a partial assignment such that $B \cup \{[\![x_i = j]\!]\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. Then, given any $\varepsilon = \text{edge}(\nu, \mu, [\![x_i = j]\!])$, there is no completion B' of B satisfying $\mathcal{M}$ such that ε is on the path defined by B'. By Proposition 7, $\neg\varepsilon$ is propagated.
  Therefore, clause (P3) propagates $\neg[\![x_i = j]\!]$.
- Let B be a partial assignment such that $B \cup \{\neg[\![x_i = j]\!]\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. Due to constraint $\text{EO}\{[\![x_i = j']\!] \mid a_i \leq j' \leq b_i\}$, this means that, for each $j' \neq j$, $B \cup \{[\![x_i = j']\!]\}$ cannot be extended to a complete assignment satisfying $\mathcal{M}$. Therefore, as previously seen, $\neg[\![x_i = j']\!]$ is propagated. Therefore, $\text{EO}\{[\![x_i = j']\!] \mid a_i \leq j' \leq b_i\}$ propagates $[\![x_i = j]\!]\}$.

Other cases are direct consequences of Proposition 7. $\square$

Lemma 1. *Given a NNF $\mathcal{N}$ rooted at ν representing formula ϕ and assignment A on the $vars(\phi)$. If $A \models \neg\phi$ then unit propagation on A using* $\mathsf{BaseNNF}$ *propagates $\neg\nu$ if A is complete, or $\mathcal{N}$ is decomposable.*

Proof. The proof is induction on the height of the NNF. We show that given complete assignment A on original variables that if $A \models \neg\phi$ where ϕ is the formula rooted at ν with height h then unit propagation on $\mathsf{BaseNNF}$ propagates $\neg\nu$.

The base case are nodes of height 1. A node labelled l is only false if $\neg l \in A$ hence the condition holds.

Suppose the result holds for nodes of height less thatn h. Given a node ν of height h then its children $\nu_1, \ldots, \nu_k$ are height less than h.

If ν is an $\vee$-node then $A \models \neg\phi$ hence $A \models \neg(\phi_1 \vee \cdots \vee \phi_k)$ where ϕ_i is the formula rooted at ν_i. Hence $A \models \neg\phi_i, 1 \leq i \leq k$. By induction $\neg\nu_i \in UP(A), 1 \leq i \leq k$, and hence using $\nu \rightarrow \nu_1 \vee \cdots \vee \nu_k$, we have $\neg\nu \in UP(A)$.

If ν is an $\wedge$-node then $A \models \neg\phi$ hence $A \models \neg(\phi_1 \wedge \cdots \wedge \phi_k)$ where ϕ_i is the formula rooted at ν_i. If A is complete then $A \models \neg\phi_i$ for some $1 \leq i \leq k$. Similarly if A is decomposable, then since the variables in each ϕ_i are distinct, $A \models \neg(\phi_1 \wedge \cdots \wedge \phi_k)$ implies that $A \models \neg\phi_i$ for some $1 \leq i \leq k$. By induction $\neg\nu_i \in UP(A)$, and hence using $\nu \rightarrow \nu_i$, we have $\neg\nu \in UP(A)$. $\square$

Theorem 9. Given an NNF $\mathcal{N}$ then $\mathsf{BaseNNF}$ is correct

Proof. Lemma 1 shows that any complete assignment A where $A \models \neg\phi$ has no extension satisfying $\mathsf{BaseNNF}$. Suppose that $A \models \phi$ we claim that $B = \{\nu \mid \neg\nu \notin UP(A)\} \cup UP(A)$ gives a complete model of $\mathsf{BaseNNF}$.

The clause ρ is satisfied since $A \not\models \neg\mathcal{N}$ so $\rho \in B$. Clearly the clauses for nodes ν where $A \models \neg\phi$ are satisfied since $\neg\nu \in B$. Consider a clause for node ν

where this does not hold. Then $\nu \in B$. If ν is an $\vee$ node, there must exist i such that $\neg\nu_i \notin UP(A)$ otherwise $\neg\nu \in UP(A)$. Hence $\nu_i \in B$ and hence the clause for ν is satisfied. If ν is an $\wedge$ node, then forall $1 \leq i \leq k$ $\neg\nu_i \notin UP(A)$ otherwise $\neg\nu \in UP(A)$. Hence $\nu_1 \in B, 1 \leq i \leq k$ and hence all clauses for ν are satisfied. $\square$

Theorem 10. Given a smooth decomposable NNF $\mathcal{N}$ then $\mathsf{FullNNF}$ is a correct encoding.

Proof. Let ϕ be a smooth decomposable NNF, and F_ϕ be $\mathsf{FullNNF}_\phi$.

$\mathsf{BaseNNF}_\phi$ is a correct encoding of ϕ (by Theorem 9) and is a subset of F_ϕ. Therefore any complete assignment A such that $A \models \neg\phi$ has no extension satisfying F_ϕ.

Consider some complete assignment A such that $A \models \phi$. We now claim some complete assignment B over $vars(F_\phi)$ exists which is consistent with A and satisfies F_ϕ. The argument proceeds by structural induction on ϕ.

If ϕ is some leaf l, $F_\phi = \{l\}$. $A \models \phi$, so A satisfies F_ϕ.

Consider the case where ϕ is either a $\vee$ or a $\wedge$ node. Assume the hypothesis holds for all subformulae of ϕ.

Assume ϕ is a $\vee$ node. Then there must be some child c of ϕ such that $A \models c$. We partition $vars(F_\phi)$ into $\phi \cup V_c \cup V_{\neg c}$, where V_c is all nodes which have c as an ancestor (and c itself). We then set c to true, and all variables in $V_{\neg c}$ to false. This satisfies N1-N3 for all $V_{\neg c}$, N1 for ϕ and N3 for c. As ϕ is smooth, $vars(c) = vars(\phi)$ so no leaf clauses are made false. Removing satisfied clauses and false literals, the remaining clauses are exactly F_c. By the induction hypothesis, some assignment $B \supset A$ over $vars(F_c)$ satisfies F_c. Thus $B \cup \{\phi\} \cup \{\neg\nu \mid \nu \in V_{\neg c}\}$ satisfies F_ϕ, so the hypothesis holds for ϕ.

Now assume ϕ is a $\wedge$ node $(c_1 \wedge \ldots \wedge c_m)$. $A \models \phi$, so $\forall j.\ A \models c_j$. ϕ is decomposable, so for each pair c_i, c_j $vars(c_i) \cap vars(c_j) = \emptyset$. To satisfy N2 for ϕ, we must set $B \supseteq \{c_1, \ldots, c_m\}$. This also satisfies N3 for $c_1, \ldots, c_m$. Removing satisfied clauses, we are left with $F_{c_1} \cup \ldots \cup F_{c_m}$. By the induction hypothesis, there is a set of assignments $B_1, \ldots, B_m$, each consistent with A, satisfying each $F_{c_1}, \ldots, F_{c_m}$. These assignments are over disjoint sets of variables, so $B_1 \cup \ldots B_m$ satisfies $F_{c_1} \cup \ldots \cup F_{c_m}$. Thus $\{phi, c_1, \ldots, c_m\} \cup B_1 \cup \ldots \cup B_m$ satisfies F_ϕ and is consistent with A, so the hypothesis holds for ϕ.

Thus, F_ϕ is a correct encoding of ϕ.

Theorem 11. Given a smooth decomposable NNF $\mathcal{N}$, then unit propagation on $\mathsf{FullNNF}$ enforces domain consistency. $\square$

Proof. Let F be $\mathsf{FullNNF}_\mathcal{N}$, A a partial assignment over $vars(N)$, and l some literal on $vars(\mathcal{N})$ such that $l \notin UP_F(A)$.

The terminal l exists and has at least one parent p such that $\neg p \notin UP(A)$ (otherwise clause N3 or N4 would have propagated $\neg l$). p is either the root r, or itself has some parent p', $\neg p' \notin UP(A)$. Thus, there is a chain of ancestors $[l = p_1, p_2, \ldots, p_k = r]$ such that $\neg p_i \notin UP(A)$.

We now show that for each p_i there is some partial assignment $asg(p_i)$ over $vars(p_i)$, which is consistent with $A \cup \{l\}$ and satisfies p_i. We proceed via induction. Clearly $A \cup \{l\}$ satisfies $l = p_1$. Now consider p_i, $i \geq 2$. By the induction hypothesis, there is some $asg(p_{i-1})$ over $vars(p_{i-1})$ satisfying p_{i-1}. p_i is either a $\vee$ or a $\wedge$ node. If p_i is a $\vee$ node, $asg(p_{i-1})$ satisfies p_i. As $\mathcal{N}$ is smooth, $vars(p_i) = vars(p_{i-1})$. Therefore $asg(p_i) = asg(p_{i-1})$ satisfies the induction hypothesis for p_i. If p_i is a $\wedge$ node, p_i has some set of children $p_{i-1}, c_1, \ldots, c_m$. As $\neg p_i \notin UP(A)$, $\neg c_j \notin UP(A)$ for each c_j (otherwise clause N2 would have propagated). F is a superset of $\mathsf{BaseNNF}_{\mathcal{N}}$, so by Lemma 1 there must be some assignment $asg(c_j)$ to $vars(c_j)$ which is consistent with A and satisfies c_j. $\mathcal{N}$ is decomposable, so the children of p_i share no variables. Therefore $asg(p_i) = asg(p_{i-1}) \cup asg(c_1) \cup \ldots \cup asg(c_m)$ satisfies p_i, and is over $vars(p_i)$. As each $asg(c_j)$ is consistent with A, $asg(p_{i-1})$ is consistent with A and $l \in asg(p_{i-1})$, $asg(p_i)$ satisfies the induction hypothesis for p_i. Therefore the induction hypothesis holds for p_i.

Thus, there is some assignment $asg(p_k)$ over $vars(p_k)$ consistent with $A \cup \{l\}$ which satisfies p_k. Since $p_k = \mathcal{N}$, $asg(p_k)$ is an total assignment over $vars(calN)$ which satisfies $\mathcal{N}$ and contains l.

Therefore $\mathsf{FullNNF}$ enforces domain consistency. $\square$