# 1 Relevancy Propagation

**Problem.** We are given a set of clauses $\mathcal{C} = \{C_1, \ldots, C_n\}$ and we are building an incremental partial model $\mathcal{M}$ of $\mathcal{C}$. Given such a partial interpretation, determine which clauses and variables can we safely ignore while keeping satisfiability.

**Example 1.** *Consider the following formula*

$$\phi \equiv (x \vee (y \leftrightarrow z)) \wedge (\neg x \vee \neg y \vee z) \ . \tag{1}$$

*Now, assume the partial model has an assignment for $x$, i.e. $x^{\mathcal{M}} = \top$. This assignment alone is enough to satisfy the first conjunct, and so it becomes irrelevant in further reasoning. Choosing values for $y$ and $z$, we can now safely ignore the first conjunct. If we now choose $y^{\mathcal{M}} = \bot$, we will also satisfy the second conjunct, making the choice of $z$ completely irrelevant.*

Previous example shows that partial models can help us eliminate from consideration variables and sub-formulas of the formula we are trying to satisfy. This can be very important when applied in the SMT context, where every abstract boolean assertion corresponds to a theory literal. Each asserted literal this adds to the computational weight of the decision procedure used to decide the underlying theory. Being able to ignore such "irrelevant" literals could thus be of significant value.

**Previous work.** This idea has already been explored in the literature [14, 18, 11]. While [14] proves complexity results, [18] shows how to use relevancy to empower a non-clausal SAT solver to an advantage. In the SMT context, [11] the relevancy of literals in the definitional CNF encoding [19] is determined by keeping the original non-clausal representation available. This is reported to be useful in expensive theories (bit-vectors and quantified theories).

## 1.1 Preliminaries

We call a variable $x$ *irrelevant* for $\phi$, in a partial model $\mathcal{M}$, if for each extension of $\mathcal{M}' \supseteq \mathcal{M}$ we have that

$$\mathcal{M}' \cup \{x\} \vDash \phi \Longleftrightarrow \mathcal{M}'\{\neg x\} \vDash \phi \ ,$$

otherwise the variable $x$ is *relevant.*

When transforming a formula $\phi$ to CNF, the transformation algorithm might introduce new variables that represent nodes in original formula. The algorithm will produce a set of clause, and we will identify two different types of clauses: regural clauses, which we denote with $C$, and definitinoal clauses $D^x$, where $x \in D^x$ in the subscript denotes that the clause is used to define the fresh variable $x$. Given a formula $\phi$, we denote with $\mathsf{CNF}(\phi)$ the result of such a transformation to conjunctive normal form, i.e.

$$\mathsf{CNF}(\phi) = \{C_1, \ldots, C_m, D_1^{x_1}, \ldots, D_n^{x_n}\} \ .$$

**Example 2.** *Consider the formula* (1) *again, and consider a Tseitin-style* CNF *transformation*

$$\mathsf{CNF}(\phi) = \{\overbrace{(x \vee w)}^{C_1}, \overbrace{(\neg x \vee \neg y \vee \neg z)}^{C_2}, \overbrace{(\neg y \vee z \vee \neg w)}^{D_1^w}\} \cup \tag{2}$$

$$\{\underbrace{(y \vee \neg z \vee \neg w)}_{D_2^w}, \underbrace{(\neg y \vee \neg z \vee w)}_{D_3^w}, \underbrace{(y \vee z \vee w)}_{D_4^w}\} \tag{3}$$

*In the clauses above, the fresh variables $w$ is used to define the subterm $(x \leftrightarrow y)$, i.e. $w \leftrightarrow (x \leftrightarrow y)$.*

For a variable $x$, we will keep $\mathsf{rel}_\phi(x, \mathcal{M})$ as the number of relevant clauses $C$, that don't define $x$, such that $x$ appears in $C$. As we are building the partial model we note the following rules for updating the relevancy of clauses:

1. in empty model all the clauses and variables are relevant;
2. if a clause $C$ is satisfied in $\mathcal{M}$, then $C$ becomes irelevant;
3. if $\mathsf{rel}_\phi(w, \mathcal{M}) = 0$ then all the clauses $D^w \in \mathsf{CNF}(\phi)$ become irellevant.

**Lemma 1.** *If all the rules above are applied and $\mathsf{rel}_\phi(x, \mathcal{M}) = 0$, then $x$ is irellevant.*

**Example 3.** *The following table shows which clauses become irellevant during the construction of the model $\mathcal{M}$ for $\phi$.*

| *rule* | $\mathcal{M}$ | $C_1$ | $C_2$ | $D_1^w$ | $D_2^w$ | $D_3^w$ | $D_4^w$ | $x$ | $y$ | $z$ | $w$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *rel1* | $\emptyset$ | $\top$ | $\top$ | $\top$ | $\top$ | $\top$ | $\top$ | *2* | *5* | *5* | *1* |
| *decide* | $\{x\}$ | $\top$ | $\top$ | $\top$ | $\top$ | $\top$ | $\top$ | ***2*** | *5* | *5* | *1* |
| *rel2* | $\{x\}$ | $\bot$ | $\top$ | $\top$ | $\top$ | $\top$ | $\top$ | *1* | *5* | *5* | ***0*** |
| *rel3* | $\{x\}$ | $\bot$ | $\top$ | $\bot$ | $\bot$ | $\bot$ | $\bot$ | *1* | *1* | *1* | *0* |
| *decide* | $\{x, \neg y\}$ | $\bot$ | $\top$ | $\bot$ | $\bot$ | $\bot$ | $\bot$ | *1* | ***1*** | *1* | *0* |
| *rel2* | $\{x, \neg y\}$ | $\bot$ | $\bot$ | $\bot$ | $\bot$ | $\bot$ | $\bot$ | *0* | *0* | *0* | *0* |

*Note that when we decided the variable $y$, it was only appearing in one relevant clauses, so the choice of $\neg y$ was obvious. In general, we can keep separate $\mathsf{rel}_\phi^+$ and $\mathsf{rel}_\phi^-$ for a variable and, if a variable only apears in one polarity, we choose that one first.*[1]

## 1.2 Implementation

In order to implement the above relevancy scheme into a modern SAT solver, we must have

- keep a list of definition clauses $D^x$ for each variable $x$;
- have a mechanizm for detecting satisfied clauses;
- keep up-to-date $\mathsf{rel}_\phi$ information on the variables;
- have all of the above backtrackable.

We then use this information to only branch on relevant variables and propagate only on relevant clauses. Notice that all of the above could be an approximation.

[1] If the varibale is not a theory atom, this decision is sufficient but, in general, it is only a heuristic

## References


[1] S. Bardin, P. Herrmann, and F. Perroud. An alternative to SAT-based approaches for bit-vectors. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 84–98, 2010.

[2] C.W. Barrett, D.L. Dill, and J.R. Levitt. A decision procedure for bit-vector arithmetic. In *Proceedings of the 35th annual Design Automation Conference*, pages 522–527. ACM, 1998.

[3] N. Bjørner, A. Blass, Y. Gurevich, and M. Musuvathi. Modular difference logic is hard. *Arxiv preprint arXiv:0811.0987*, 2008.

[4] N.S. Bjørner and M.C. Pichora. Deciding fixed and non-fixed size bit-vectors. *Tools and Algorithms for the Construction and Analysis of Systems*, page 376, 1998.

[5] R. Brinkmann and R. Drechsler. RTL-datapath verification using integer linear programming. In *Design Automation Conference, 2002. Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design. Proceedings.*, pages 741–746. IEEE, 2002.

[6] R. Brummayer and A. Biere. Local two-level and-inverter graph minimization without blowup. *Proc. MEMICS*, 6:57, 2006.

[7] R. Brummayer and A. Biere. Boolector: An efficient SMT solver for bit-vectors and arrays. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 174–177, 2009.

[8] R. Bruttomesso, A. Cimatti, A. Franzén, A. Griggio, Z. Hanna, A. Nadel, A. Palti, and R. Sebastiani. A lazy and layered SMT (BV) solver for hard industrial verification problems. In *Proceedings of the 19th international conference on Computer aided verification*, pages 547–560. Springer-Verlag, 2007.

[9] R. Bruttomesso and N. Sharygina. A scalable decision procedure for fixed-width bit-vectors. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, pages 13–20. ACM, 2009.

[10] D. Cyrluk, O. Möller, and H. Rueß. An efficient decision procedure for the theory of fixed-sized bit-vectors. In *Computer Aided Verification*, pages 60–71. Springer, 1997.

[11] L. De Moura and N. Bjørner. Relevancy propagation. Technical Report MSR-TR-2007-140, Microsoft Research, 2007.

[12] P. Jackson and D. Sheridan. Clause form conversions for Boolean circuits. In *Theory and Applications of Satisfiability Testing*, pages 183–198. Springer, 2005.

[13] S. Jha, R. Limaye, and S. Seshia. Beaver: Engineering an efficient SMT solver for bit-vector arithmetic. In *Computer Aided Verification*, pages 668–674. Springer, 2009.

[14] J. Lang and P. Marquis. Complexity results for independence and definability in propositional logic. In *Pprinciples of Knowlege Representation and Reasoning*, pages 356–367. Morgan Kaufmann Publishers, 1998.

[15] P. Manolios and D. Vroon. Efficient circuit to CNF conversion. *Theory and Applications of Satisfiability Testing–SAT 2007*, pages 4–9, 2007.

[16] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(5):29, 2007.

[17] G. Parthasarathy, MK Iyer, K.T. Cheng, and L.C. Wang. An efficient finite-domain constraint solver for circuits. 2004.

[18] C. Thiffault, F. Bacchus, and T. Walsh. Solving non-clausal formulas with DPLL search. *Principles and Practice of Constraint Programming–CP 2004*, pages 663–678, 2004.

[19] G.S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in constructive mathematics and mathematical logic*, 2(115-125):10–13, 1968.

[20] M.N. Velev. Efficient translation of boolean formulas to CNF in formal verification of microprocessors. In *Proceedings of the 2004 Asia and South Pacific Design Automation Conference*, pages 310–315. IEEE Press, 2004.