Relevance for SAT(ID)

Joachim Jansen[★], **Bart Bogaerts**[♠], **Jo Devriendt**[★], **Gerda Janssens**[★] and **Marc Denecker**[★]
★ KU Leuven, Leuven, Belgium, firstname.lastname@kuleuven.be
♠ Aalto University, Espoo, Finland, bart.bogaerts@aalto.fi

Abstract

Inductive definitions and justifications are well-studied concepts. Solvers that support inductive definitions have been developed, but several of their computationally nice properties have never been exploited to improve these solvers. In this paper, we present a new notion called *relevance*. We determine a class of literals that are relevant for a given definition and partial interpretation, and show that choices on irrelevant atoms can never benefit the search for a model. We propose an *early stopping criterion* and a *modification of existing heuristics* that exploit relevance. We present a first implementation in MinisatID and experimentally evaluate our approach, and study how often existing solvers make choices on irrelevant atoms.

1 Introduction

Since the addition of conflict-driven clause learning [Marques-Silva and Sakallah, 1999], SAT solvers have made huge leaps forward. Now that these highly-performant SAT-solvers exist, research often stretches *beyond SAT* by extending the language supported by SAT with richer language constructs. Research fields such as SAT Modulo Theories (SMT) [Barrett *et al.*, 2009], Constraint Programming (CP) [Apt, 2003] in the form of lazy clause generation [Stuckey, 2010], or Answer Set Programming (ASP) [Marek and Truszczyński, 1999] could be seen as following this approach. In this paper, we focus on the logic PC(ID): the Propositional Calculus extended with Inductive Definitions [Mariën *et al.*, 2007]. The satisfiability problem for PC(ID) encodings is called SAT(ID) [Mariën *et al.*, 2008]. SAT(ID) can be formalised as SAT modulo a theory of inductive definitions and is closely related to answer set solving. In fact, all the work we introduce in this paper is also applicable to so-called generate-define-test answer set programs.

In this paper we introduce an alternative criterion to determine satisfiability of a PC(ID) theory. Instead of searching for a variable assignment that satisfies the PC(ID) theory, we search for a *partial* assignment that contains sufficient information to guarantee satisfiability. Our approach is based on the notion of *justifications* [Denecker and De Schreye, 1993; Denecker *et al.*, 2015]. As a small example, consider the following theory.

$$p_{\mathcal{T}}.$$
$$\left\{\begin{array}{ll} p_{\mathcal{T}} \leftarrow a \wedge b. \\ a \leftarrow d \vee \neg e \vee f. \\ b \leftarrow c \vee \neg g \vee h. \\ e \leftarrow f \vee \neg h \vee i. \end{array}\right\}$$

This theory contains one constraint, that $p_{\mathcal{T}}$ must hold, and a definition (between '{' and '}') of $p_{\mathcal{T}}$ in terms of variables a to i. One way to check satisfiability would be to generate an assignment of all variables that satisfies the above theory (this is the classical approach to solving such problems). What we do, on the other hand, is to search for a *partial* assignment to these variables such that $p_{\mathcal{T}}$ is *justified* in that partial assignment. Consider for example the partial assignment where $p_{\mathcal{T}}$, a, b, c and d are true and everything else is unknown. In this assignment, a and b are *justified* because d and c hold respectively; $p_{\mathcal{T}}$ is *justified* because both a and b are justified. This suffices to determine satisfiability of the theory, without considering the definition of e for instance.

We introduce the notion of *relevance*. Intuitively, a literal is *relevant* if it can contribute to justifying the theory. In the above example, as soon as d is assigned *true*, the variable e becomes *irrelevant*. From that point onwards, search should not take e's defining rule into account.

Based on this notion of relevance, we define two extensions of existing SAT(ID) solvers. The first is to modify the *decision heuristics*: we show that deciding on irrelevant literals *never* affects any possible justification for $p_{\mathcal{T}}$. Hence, we propose to only choose on relevant literals, otherwise leaving the heuristics unchanged. The second is to implement an *early stopping criterion* that allows a solver to decide the theory is satisfiable in a *partial* assignment.

The main contributions of this paper are **(1)** the formal identification of the set of relevant literals, **(2)** showing that assigning a value to an irrelevant literal does not affect satisfiability, **(3)** proving correctness of the new early stopping criterion, and **(4)** experimentally evaluating the proposed approach.

The rest of this paper is structured as follows. In Section 2 we present some necessary preliminaries. In Section 3, we present our new theory, essentially introducing relevance, the

new algorithms and the associated correctness theorems. We experimentally evaluate our proposed approach in Section 4 and conclude in Section 5.

2 Preliminaries

PC(ID)

In this section, we briefly recall the syntax and semantics of Propositional Calculus extended with Inductive Definitions (PC(ID)) [Mariën, 2009].

A truth value is one of $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$; $\mathbf{t}$ represents *true*, $\mathbf{f}$ *false* and $\mathbf{u}$ *unknown*. The truth order $\leq_t$ on truth values is given by $\mathbf{f} \leq_t \mathbf{u} \leq_t \mathbf{t}$, the precision order $\leq_p$ is given by $\mathbf{u} \leq_p \mathbf{f}$ and $\mathbf{u} \leq_p \mathbf{t}$. Let Σ be a finite set of symbols called *atoms*. A *literal* l is an atom p or its negation $\neg p$. In the former case, we call l *positive*, in the latter, we call l *negative*. We use $\overline{\Sigma}$ to denote the set of all literals over Σ. If l is a literal, we use $|l|$ to denote the atom of l, i.e., to denote p if $l = p$ or $l = \neg p$. We use $\sim l$ to denote the literal that is the negation of l, i.e., $\sim p = \neg p$ and $\sim\neg p = p$. A *partial interpretation* $\mathcal{I}$ is a mapping from Σ to truth values. We use the notation $\{p_1^{\mathbf{t}}, \ldots, p_n^{\mathbf{t}}, q_1^{\mathbf{f}}, \ldots, q_m^{\mathbf{f}}\}$ for the partial interpretation that maps the p_i to $\mathbf{t}$, the q_i to $\mathbf{f}$ and all other atoms to $\mathbf{u}$. We call a partial interpretation *two-valued* if it does not map any atom to $\mathbf{u}$. If $\mathcal{I}$ and $\mathcal{I}'$ are partial interpretations, we say that $\mathcal{I}$ is less precise than $\mathcal{I}'$ (notation $\mathcal{I} \leq_p \mathcal{I}'$) if for all $p \in \Sigma$, $\mathcal{I}(p) \leq_p \mathcal{I}'(p)$. If φ is a propositional formula, we use $\varphi^{\mathcal{I}}$ to denote the truth value ($\mathbf{t}$, $\mathbf{f}$ or $\mathbf{u}$) of φ in $\mathcal{I}$, based on the Kleene truth tables [Kleene, 1938]. If $\mathcal{I}$ is a partial interpretation and l a literal, we use $\mathcal{I}[l : \mathbf{t}]$ to denote the partial interpretation equal to $\mathcal{I}$, except that it interprets l as $\mathbf{t}$ (and similar for $\mathbf{f}$, $\mathbf{u}$). With σ a set of symbols, we use the notation $\mathcal{I}|_\sigma$ to indicate the *restriction* of $\mathcal{I}$ to symbols in σ. I.e., $\mathcal{I}|_\sigma(p) = \mathbf{u}$ if $p \notin \sigma$ and $\mathcal{I}|_\sigma(p) = \mathcal{I}(p)$ otherwise.

A two-valued *interpretation* I is a subset of Σ. We identify an interpretation I with the two-valued partial interpretation that maps $p \in I$ to $\mathbf{t}$ and $p \in \Sigma \setminus I$ to $\mathbf{f}$.

An inductive definition Δ over Σ is a finite set of rules of the form $p \leftarrow \varphi$ where $p \in \Sigma$ and φ is a propositional formula over Σ. We call p the head of the rule and φ the body of the rule. We call p *defined in* Δ if p occurs as the head of a rule in Δ. The set of all symbols defined in Δ is denoted by $\mathit{defs}(\Delta)$. All other symbols are called *open in* Δ. The set of open symbols in Δ is denoted $\mathit{opens}(\Delta)$. We say that a literal l is *defined in* Δ if $|l| \in \mathit{defs}(\Delta)$. We use the *parametrised well-founded semantics* for inductive definitions [Denecker and Vennekens, 2007]. That is, interpretation I is a model of Δ (denoted $I \models \Delta$) if I is the well-founded model of Δ in context $I|_{\mathit{opens}(\Delta)}$. We call an inductive definition *total* if for every interpretation I of the open symbols, the well-founded model in context I is a two-valued interpretation.

A *PC(ID)* theory $\mathcal{T}$ over Σ is a set of propositional formulas, called constraints, and inductive definitions over Σ. Interpretation I is a model of $\mathcal{T}$ if I is a model of all definitions and constraints in $\mathcal{T}$. Without loss of generality [Mariën, 2009], we assume that every PC(ID) theory is in the **DEFNF** normalform, where $\mathcal{T} = \{p_\mathcal{T}, \Delta\}$ and

- $p_\mathcal{T}$ is an atom,
- Δ is an inductive definition defining $p_\mathcal{T}$,
- every rule in Δ is of the form $p \leftarrow l_1 \odot \cdots \odot l_n$, where $\odot$ is either $\wedge$ or $\vee$, p is an atom, and each of the l_i are literals,
- every atom p is defined in at most one rule of Δ.

A rule in which $\odot$ is $\wedge$, respectively $\vee$ is called a *conjunctive*, respectively *disjunctive*, rule. The rules in a definition Δ impose a *direct dependency relation*, denoted dd_Δ, between literals, defined as follows. For literals $from$ and to, it holds that $(from, to) \in dd_\Delta$ in Δ if there is a rule $p \leftarrow l_1 \odot \cdots \odot l_n$ in Δ such that for some i, either $from = p$ and $to = l_i$ or $from = \sim p$ and $to = \sim l_i$. The *dependency graph* of Δ is the graph $G_\Delta = (\overline{\Sigma}, dd_\Delta)$. For the remainder of the paper, we assume that some PC(ID) theory $\mathcal{T} = \{p_\mathcal{T}, \Delta\}$ is fixed; hence, we will often omit Δ and/or $\mathcal{T}$ from the notations.

It has been argued many times before [Denecker, 1998; Denecker and Ternovska, 2008; Denecker and Vennekens, 2014] that all sensible definitions in mathematical texts are total definitions. Following these arguments, in the rest of this paper we assume Δ to be a total definition.

The satisfiability problem for PC(ID), i.e., deciding whether a PC(ID) theory has a model, is called *SAT(ID)*. This problem is NP-complete [Mariën *et al.*, 2008].

Justifications

Consider graph $G = (V, E)$, with V the set of nodes and E the set of edges. If the graph contains an edge from l to l' (i.e., $(l, l') \in E$), we say that l is a parent of l' in G and that l' is a child of l in G. A node l is called a *leaf* of G if it has no children in G; otherwise it is called *internal* in G. Let $G' = (V', E')$ be another graph. We define the union of two graphs (denoted $G \cup G'$) as the graph with vertices $V \cup V'$ that contains only edges that were already in G or G'.

Suppose l is a literal with $p = |l|$ and $p \in \mathit{defs}(\Delta)$ with defining rule $p \leftarrow l_1 \odot \cdots \odot l_n$. A set of literals J_d is a *direct justification* of l in Δ if one of the following holds:

- $l = p$, $\odot$ is $\wedge$, and $J_d = \{l_1, \ldots, l_n\}$,
- $l = p$, $\odot$ is $\vee$, and $J_d = \{l_i\}$ for some i,
- $l = \neg p$, $\odot$ is $\wedge$, and $J_d = \{\sim l_i\}$ for some i,
- $l = \neg p$, $\odot$ is $\vee$, and $J_d = \{\sim l_1, \ldots, \sim l_n\}$.

Note that a direct justification of a literal can only contain children of that literal in the dependency graph.

A *justification* [Denecker and De Schreye, 1993] J of a definition Δ is a subgraph of G_Δ, such that each internal node $l \in J$ is a defined literal and the set of its children is a direct justification of l in Δ. We say that J *contains* l if l occurs as node in J. A justification is *total* if none of its leaves are defined literals. A justification can contain *cycles*.[1] A cycle is called *positive* (resp. *negative*) if it contains only positive (resp. negative) literals. It is called a *mixed* cycle otherwise.

If J is a justification and $\mathcal{I}$ a (partial) interpretation, we define the value of J in $\mathcal{I}$, denoted $V_\mathcal{I}(J)$ as follows:

- $V_\mathcal{I}(J) = \mathbf{f}$ if J contains a leaf l with $l^\mathcal{I} = \mathbf{f}$ or a positive cycle (or both).
- $V_\mathcal{I}(J) = \mathbf{u}$ if $V_\mathcal{I}(J) \neq \mathbf{f}$ and J contains a leaf l with $l^\mathcal{I} = \mathbf{u}$ or a mixed cycle (or both).

[1] In this text, we assume that Δ is finite; in this case cycles are simply loops in the graph.The infinite case is a bit more subtle, and an adapted definition of cycle is required to maintain all results presented below.

- $V_{\mathcal{I}}(J) = \mathbf{t}$ otherwise (all leaves are $\mathbf{t}$ and cycles, if any, are negative).

A literal l is *justified* (in $\mathcal{I}$, for $\mathcal{T}$) if there exists a total justification J (of Δ) that contains l such that $V_{\mathcal{I}}(J) = \mathbf{t}$. In this case, we say that such a J *justifies* l (in $\mathcal{I}$, for $\mathcal{T}$). We say that J *minimally justifies* l if J justifies l and there exists no subgraph J' of J that also justifies l.

Denecker and De Schreye [1993] showed that many semantics of logic programs can be captured by justifications. We recall their major result on the well-founded semantics.

Theorem 2.1 (Denecker and De Schreye [1993])**.** *Let J be a justification of definition Δ.*

- *Suppose $\mathcal{I}$ and $\mathcal{I}'$ are partial interpretations. If $\mathcal{I} \leq_p \mathcal{I}'$ then $V_{\mathcal{I}}(J) \leq_p V_{\mathcal{I}'}(J)$.*
- *Suppose $\mathcal{I}$ is an $opens(\Delta)$-interpretation and $\mathcal{I}'$ is the well-founded model of Δ in context $\mathcal{I}$. For each defined literal l, it holds that*

$$l^{\mathcal{I}'} = \max_{\leq_t}\{V_{\mathcal{I}}(J) \mid J \text{ a total justification containing } l\}$$

3 Relevance

Observations

The central observation in this paper is the fact that classical SAT(ID) solvers such as for example MINISAT(ID) [Mariën *et al.*, 2008; De Cat *et al.*, 2013] or the related ASP systems such as clasp [Gebser *et al.*, 2012] or DLV [Leone *et al.*, 2006] fail to exploit an important property. Recall that a PC(ID) theory $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ is assumed to be fixed throughout the text. Systems such as MINISAT(ID) search for an interpretation I such that $I \models \mathcal{T}$, while in fact they could search for a *partial* interpretation $\mathcal{I}$ and a *justification* J that justifies $p_{\mathcal{T}}$ in $\mathcal{I}$. Our claim is that even though in theory both tasks are of the same complexity, for practical applications, the latter task possesses some important advantages. Before discussing these, we provide the formal basis for our theory.

Theorem 3.1. *$\mathcal{T}$ is satisfiable if and only if there exists a partial interpretation $\mathcal{I}$ and a J that justifies $p_{\mathcal{T}}$ in $\mathcal{I}$.*

Proof. First assume that $\mathcal{T}$ is satisfiable. Then there exists an interpretation I such that ${p_{\mathcal{T}}}^I = \mathbf{t}$ and $I \models \Delta$. Theorem 2.1 (2) then yields that $\mathbf{t} = \max_{\leq_t}\{V_I(J) \mid J \text{ is a total justification that justifies } p_{\mathcal{T}}\}$. Hence, there must exist a justification J that contains $p_{\mathcal{T}}$ for which $V_I(J) = \mathbf{t}$, i.e. J justifies $p_{\mathcal{T}}$ in I. The result then follows by taking $\mathcal{I} = I$ and using J as justification.

On the other hand assume that there exists a partial interpretation $\mathcal{I}$ and a justification J such that J justifies $p_{\mathcal{T}}$ in $\mathcal{I}$. Now, let $\mathcal{I}'$ be any partial interpretation such that $\mathcal{I}' \geq_p \mathcal{I}$ and $\mathcal{I}'$ is two-valued in $opens(\Delta)$. From Theorem 2.1 (1) follows that $V_{\mathcal{I}'}(J) \geq_p V_{\mathcal{I}}(J)$, since $\mathcal{I}' \geq_p \mathcal{I}$. Because J justifies $p_{\mathcal{T}}$, we also know $V_{\mathcal{I}}(J) = \mathbf{t}$, which implies $V_{\mathcal{I}'}(J) = \mathbf{t}$. Further, $V_{\mathcal{I}'|_{opens(\Delta)}}(J) = V_{\mathcal{I}'}(J)$ since the value of a justification only depends on the edge relations in J (unchanged) and the values of open atoms (also unchanged). Let I' denote the well-founded model of Δ in context $\mathcal{I}'|_{opens(\Delta)}$. I' exists because we assume Δ to be a total definition. From Theorem 2.1 (2) we know that ${p_{\mathcal{T}}}^{I'} = \mathbf{t}$, because justification J already maps to the maximal value in the $\leq_t$ order, thus the value of the set expression in the theorem is fixed. Hence $\mathcal{T}$ is indeed satisfiable: I' is a model of $\mathcal{T}$. □

We will now identify which literals are *relevant*.

Definition 3.2 (Relevant)**.** Given a PC(ID) theory $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ and a partial interpretation $\mathcal{I}$, we define the set of relevant literals, denoted $\mathcal{R}_{\mathcal{T}}(\mathcal{I})$, as follows

- $p_{\mathcal{T}}$ is relevant if $p_{\mathcal{T}}$ is not justified,
- if $l \in \mathcal{R}_{\mathcal{T}}(\mathcal{I})$, $(l, l') \in dd_{\Delta}$ and l' is not justified, then l' is relevant.

Intuitively, a literal is relevant if making it true can help justify $p_{\mathcal{T}}$. If a partial structure is made more precise, literals may become irrelevant because they can no longer contribute to any justification that justifies $p_{\mathcal{T}}$. Often, we assume $\mathcal{T}$ is clear from the context and simply state that l *is relevant* in $\mathcal{I}$. We define the set of relevant *literals* and not the set of relevant *atoms* because, in a further stadium, one can exploit the information that e.g., a literal l is relevant, but $\sim l$ is not.

Using relevance, we aim to obtain three advantages over classical SAT(ID) solvers.

(1) We can avoid irrelevant parts of the search space.
(2) We can stop searching once a *partial* interpretation is found in which $p_{\mathcal{T}}$ is justified, instead of searching for a total interpretation.
(3) We can make solvers more robust for wrong choices.

We illustrate each of these three advantages in the following example.

Example 3.3. Let $\mathcal{T} = \{p_{\mathcal{T}}, \Delta\}$ denote the theory where

$$\Delta = \left\{ \begin{array}{ll} p_{\mathcal{T}} & \leftarrow a \wedge b. \\ a & \leftarrow d \vee \neg e \vee f. \\ b & \leftarrow \neg h \vee j. \\ d & \leftarrow c \wedge \neg g. \\ e & \leftarrow i \vee h. \\ h & \leftarrow \neg i. \end{array} \right\}$$

Let $\mathcal{I}_1$ be the partial interpretation $\{{p_{\mathcal{T}}}^{\mathbf{t}}, a^{\mathbf{t}}, b^{\mathbf{t}}, d^{\mathbf{t}}, c^{\mathbf{t}}, g^{\mathbf{f}}\}$. In this case, d is justified in $\mathcal{I}_1$, hence so is a. This means that the value of e and f cannot influence whether or not a is justified. Hence, giving a value to e or to f cannot help justifying $p_{\mathcal{T}}$, illustrating advantage **(1)**.

Let $\mathcal{I}_2$ be $\mathcal{I}_1[j : \mathbf{t}]$. In this case $p_{\mathcal{T}}$ is justified in $\mathcal{I}_2$, hence Theorem 3.1 yields that $\mathcal{T}$ is satisfiable and we do not need to search an assignment for the remaining (irrelevant) atoms, illustrating advantage **(2)**.

Let $\mathcal{I}_3$ be $\mathcal{I}_2[e : \mathbf{f}]$. It can be seen that there exists no model of $\mathcal{T}$ that is more precise than $\mathcal{I}_3$. Indeed, e is true in every model of $\mathcal{T}$ because i as well as $\neg i$ make e true. It is possible that the solver make the choice $e^{\mathbf{f}}$ early on. Theorem 3.1 shows that since $p_{\mathcal{T}}$ is justified in $\mathcal{I}_3$ a model must exist (even though the current interpretation is incompatible with that model), illustrating advantage **(3)**.

Example 3.4 (Example 3.3 continued)**.** The set of relevant literals for $\mathcal{I}_1$ is $\mathcal{R}_{\mathcal{T}}(\mathcal{I}_1) = \{p_{\mathcal{T}}, b, \neg h, j, i\}$. $p_{\mathcal{T}}$ is relevant in $\mathcal{I}_1$ because it is not justified. b is relevant in $\mathcal{I}_1$ since it is not justified and since $p_{\mathcal{T}}$, which is not justified, depends on

it. $\neg h$ and j are relevant in $\mathcal{I}_1$ since they are not justified and potentially useful to justify b. i is relevant in $\mathcal{I}_1$ since it is not justified and might be used to justify $\neg h$. $p_\mathcal{T}$ is justified in $\mathcal{I}_2$ and $\mathcal{I}_3$, which means there are no relevant literals in these partial interpretations.

Using these observations, we show how to exploit relevance to reduce the search space.

Exploiting Relevance

In order to exploit relevance, we assume that some search algorithm for SAT(ID) is given; we assume this algorithm searches for an interpretation I such that $I \models \mathcal{T}$. We implemented our techniques in a conflict-driven clause learning DPLL solver. However, it deserves to be stressed that all ideas developed here are independent of the choice of search strategy or heuristic. We propose the following modification to such a solver: choose only on relevant literals and stop search early if there are no unassigned relevant literals in the current search state. Note that if there are no unassigned relevant literals left, $p_\mathcal{T}$ is justified if and only if there is a model (according to Theorem 3.1). In order to prove correctness of our modification, we will use the following result.

Theorem 3.5. *Let $\mathcal{T} = \{p_\mathcal{T}, \Delta\}$ be a PC(ID) theory. Suppose $\mathcal{I}$ is a partial interpretation and l^{irr} a literal such that $\mathcal{I}(|l^{irr}|) = \mathbf{u}$ and l^{irr} is not relevant in $\mathcal{I}$. If $p_\mathcal{T}$ is justified in some partial interpretation $\mathcal{I}'$ more precise than $\mathcal{I}$, then $p_\mathcal{T}$ is also justified in $\mathcal{I}'[l^{irr} : \mathbf{f}]$ and in $\mathcal{I}'[l^{irr} : \mathbf{t}]$.*

Proof. Let J be a justification that minimally justifies $p_\mathcal{T}$ in $\mathcal{I}'$. Note that leaves in J are open and true in $\mathcal{I}'$; cycles, if any, are negative.

J_1 is derived from J as follows: for each defined literal x in J that is justified in $\mathcal{I}$: remove the edges from x to its children. Finally, remove all parts not reachable from $p_\mathcal{T}$. By construction, the leafs of J' are either open literals, or defined literals justified in $\mathcal{I}$.

Let J_2 be a justification that contains only literals justified in $\mathcal{I}$ and that justifies all these literals. Now, define J' as $J_1 \cup J_2$. Since J_1's internal nodes are not justified in $\mathcal{I}$, and all literals in J_2 are justified in $\mathcal{I}$, this union introduces no new loops not already in J_1 or in J_2. Additionally, J' only contains open literals already in J_1 or in J_2. This means J' is a justification that justifies $p_\mathcal{T}$ in $\mathcal{I}'$, since J' contains $p_\mathcal{T}$, leaves in J' are open and true in $\mathcal{I}'$; cycles, if any, are negative.

J' cannot contain l^{irr} in the part that originated from J_1, because those are all literals that were relevant in $\mathcal{I}$. Any occurence of l^{irr} in any part that originated from J_2 has to be an internal node, since $V_\mathcal{I}(J_2) = \mathbf{t}$, which demands that all leafs are true. Hence, any occurrence of l^{irr} cannot be a leaf in J', which means that changing its interpretation does not affect the value of J' in $\mathcal{I}'$. Therefore, $p_\mathcal{T}$ is also justified in $\mathcal{I}'[l^{irr} : \mathbf{f}]$ and $\mathcal{I}'[l^{irr} : \mathbf{t}]$. □

Theorem 3.5 shows that any search algorithm that can arrive in a state in which $p_\mathcal{T}$ is justified by deciding on a literal l that is irrelevant in its current partial interpretation, can also arrive in such a state *without* deciding on l. Hence, if a literal l is irrelevant, it is useless to choose on that literal if the goal is to justify $p_\mathcal{T}$. This is exactly what our proposed solver modification does; we restrict the choices of a search algorithm to the set of relevant literals.

4 Experimental evaluation

In order to empirically evaluate our proposed approach, we adjusted the IDP3 system [De Cat *et al.*, 2016] and its underlying solver MINISAT(ID) [De Cat *et al.*, 2013] to take relevance into account. Integrating relevance into the search process is simple: it is a non-intrusive modification to the search heuristic to not choose on certain literals. However, calculating which literals are relevant requires a tight integration with the solver being adapted. Detailed information about the solver state, such as the dependency graph and the justification status for literals, are required in order to calculate which literals are relevant. For the purpose of this paper, we opted for a simple and non-intrusive implementation that had the drawback of significant overhead. Therefore, our experiments are based on search space metrics rather than absolute solving time. This performance overhead is not inherent to maintaining relevance. Large parts of the bookkeeping we do now is discovering information that is already present somewhere in the solver internally. However, extracting all the necessary information is an engineering task we did not complete yet. In this section, we will answer the following questions to evaluate whether it is worth investigating relevance further:

(Q1) How often does the VSIDS, the current state-of-the-art heuristic for SAT, make irrelevant decisions?

(Q2) Can we improve the performance of SAT(ID) solvers using relevance?

The complete set of experiments and information on how to run them can be found at `https://dtai.cs.kuleuven.be/static/krr/files/experiments/idp_relevance_experiments.tar.gz`.

For these experiments we selected problems from previous ASP competitions that could be encoded without the use of aggregates and functions, since we do not yet support these language constructs. We ran these problems on an **Intel(R) Xeon(R) CPU E5645 @ 2.40GHz** CPU, using a time limit of 7200 seconds and a memory limit of 8GiB.

Problem	**#**	μ_{irr^d}	$\sigma^2_{irr^d}$	μ_{irr^c}	$\sigma^2_{irr^c}$
GG	0/30	-	-	-	-
HP	102/102	27.37%	2.87%	36.99%	7.88%
NQueens	14/29	22.55%	0.11%	0.43%	0.00%
PPM	13/30	22.93%	5.10%	4.98%	0.00%
RR	0/30	-	-	-	-
Sokoban	4/30	48.20%	7.62%	0.96%	0.01%
Solitaire	17/27	13.32%	0.13%	3.95%	0.19%
SM	27/30	96.40%	0.13%	0.01%	0.00%
Visit All	19/30	15.02%	2.16%	16.45%	3.42%

Table 1: Statistics per problem: the columns represent number of instances solved, percentage of irrelevant decisions (mean μ and variance σ^2), and percentage of irrelevant decisions in conflicts (mean μ and variance σ^2). GG = Graceful Graphs, HP = Hamiltonian Path, PPM = Permutation Pattern Matching, RR = Ricochet Robots, SM = Stable Marriage.

To answer **(Q1)** we ran all the above problems and their instances with a solver configuration that uses the VSIDS heuristic while keeping track of relevance. We keep track of whether the decision made by VSIDS is relevant without actually preventing decisions on irrelevant literals (i.e., the search behaviour is **not** affected). Table 1 shows the problems and their number of successfully solved versus total number of instances (second column). In this table we show the mean (μ) and variance (σ^2) of **(1)** irr^d: the ratio between the irrelevant decisions made by VSIDS and the total number of decisions, and **(2)** irr^c: the ratio between the number of irrelevant decisions involved in conflicts and total number of decisions involved in conflicts. In order to obtain the latter statistic, we analyse the conflicts that occur during solving by applying full resolution on them. The resulting clause only contains decision literals. We then count the total number of decisions as well as the number of decision literals that were irrelevant at the time they were made.

Due to our significant performance overhead in keeping track of relevance, we were not able to solve a single instance of the Graceful Graphs and the Ricochet Robots problems. We observe that, on average, the VSIDS heuristics chooses a considerate amount of irrelevant literals. There is even an outlier in the Stable Marriage problem where more than 96% of the choices were irrelevant. Therefore we can say for **(Q1)** that, on average, VSIDS selects a significant amount of irrelevant choice literals on the classical benchmarks.

On the other hand, irr^c is generally significantly lower than irr^d, meaning that the irrelevant decisions that are made by VSIDS hardly ever lead to conflicts. In order to further inspect the behaviour of relevance we discuss cactusplots for the behaviour of the experimental runs in Table 1 for instances that were solved both by VSIDS (labeled "NR", for "No Relevance") and by our proposed solver modification (labeled "R", for "Relevance").

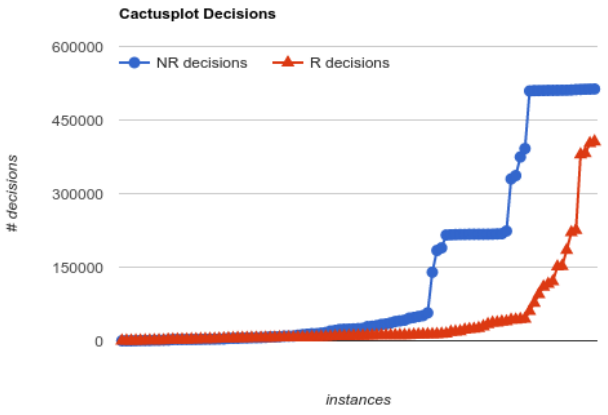

Figure 1: Cactusplot of # decisions

Figure 1 shows that we succeeded in reducing the number of decisions made, and Figure 2 shows that this did not affect the number of conflicts for these benchmarks. This initial observation is not encouraging, since the number of conflicts is often taken as a measure for the size of the search space

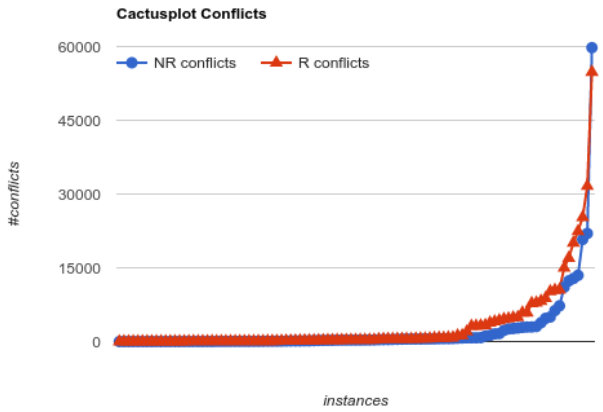

Figure 2: Cactusplot of # conflicts

traversed. In what follows, we

(1) argue that in certain applications, reducing the number of decisions is itself already a desirable property

(2) investigate why we observe no reduction in the number of conflicts.

Reducing decisions: a contribution on its own Even if we did not manage to significantly reduce the number of conflicts, reducing the number of decisions is already a significant achievement for certain applications. To illustrate this, we consider lazy model expansion [De Cat *et al.*, 2015]. The approach of lazy model expansion is to interleave the grounding and the search space. That is, a first-order theory is not translated to propositional logic a priori. Instead, depending on the search of a SAT(ID) solver, certain parts of the grounding are generated. This approach works (roughly) as follows. A PC(ID) theory $\mathcal{T}$ is initialised as $p_{\mathcal{T}}$. Each time a literal that has no definition in $\mathcal{T}$ is assigned a value, some external procedure is called and the definition of that literal is added to $\mathcal{T}$. This approach is particularly fruitful in applications with very large (possibly infinite) domains where it is simply infeasible to generate the entire grounding.

Adding more definitions to $\mathcal{T}$ is possibly a costly operation and should be avoided as much as possible. If we combine lazy grounding with our proposed relevance approach we will greatly benefit from the reduced number of decisions made, because avoiding non relevant decisions results in fewer variables that are assigned a value (also propagations that follow from irrelevant decisions!) and hence less grounding.

Analysing the conflict behaviour We noticed that, while VSIDS makes lots of choices on irrelevant literals, the number of conflicts did not increase significantly. One possible explanation for this behaviour is that in the examples we used, the irrelevant parts of the search space are not strongly constrained. One real-world example of a problem where irrelevant parts of the search space are still heavily constrained is a scheduling problem for a trucking company, such that each scheduled truck can solve a packing problem. Solutions to such problems are often hand-made in such a way that they

take relevance into account (i.e., first solving the scheduling and then only trying to solve the relevant packing problems), because the current generation of solvers cannot handle this problem directly. An instance of such a problem and its solution is given by Verstichel [2013]. In order to test the hypothesis that underconstrained problems are indeed at the root of this behaviour, we construct a small encoding in which we force irrelevant literals to represent that a combinatorially hard probem is satisfiable:

$$\forall x[1..n] : XOR(x) \Leftrightarrow (P(x) \Leftrightarrow \neg Q(x)).$$
$$\forall x[1..n] : XOR(x) \Rightarrow pigeon_{k,k}.$$
$$\forall x[1..n] : \neg XOR(x) \Rightarrow pigeon_{k,k+1}.$$

Figure 3: Hand-made encoding showing the use of relevance. For ease of reading, a first-order version of the encoding is presented.

Figure 3 encodes the following problem. Predicates P and Q can be chosen freely (they are opens of the underlying definition). For each domain element d, $XOR(d)$ holds if and only if exactly one of $P(d)$ and $Q(d)$ holds. Next, if $XOR(d)$ is $\mathbf{f}$, an encoding of an unsatisfiable pigeonhole must be satisfied. If $XOR(d)$ is $\mathbf{t}$, an encoding of a satisfiable pigeonhole problem must be satisfied. Thus, the problem can only be solved by making $XOR(d)$ $\mathbf{t}$ for all instances. At any point during search, VSIDS can make choices on the variables occurring in the encoding of the pigeonhole problems. As soon as XOR is decided, the relevance heuristic, on the other hand, only makes choices on variables in the relevant subproblem. If unlucky, VSIDS behaviour can lead to a great deal of time wasted and a great number of unnecessary conflicts during search.

In order to test the behaviour of VSIDS on this problem we used the same setup as in Table 1. This time we also measured the solving time and memory needed, as well as the total number of decisions and conflicts. We ran the above encoding with the domain of size $n = 250$ and $k = 9$. The results, presented Table 2, show that there are problems where taking relevance leads to a greatly reduced number of conflicts, which means a reduction of the search space. Increasing the domain size only widened the gap between VSIDS and relevance. Runtime statistics of such additional experiments are omitted here, for brevity concerns. These observations lead to a definite positive answer to **(Q2)**.

	VSIDS	**Relevance**
Running time (ms)	35691	**12523**
Memory (MB)	**192**	217.1
# decisions	10317218	**150851**
# conflicts	116434	**20900**
% irr^d	96.15%	0.00%
% irr^c	96.49%	0.00%

Table 2: Performance of VSIDS vs. Relevance on the handmade problem encoding shown in Figure 3

5 Conclusion

In this paper we formally identified a set of literals called *relevant*; we showed that irrelevant literals cannot influence the justification status of a PC(ID) theory and hence that making choices on irrelevant literals is useless with regards to proving the satisfiability of the given PC(ID) theory. We proposed two simple solver modifications: choosing only on relevant literals and stopping early. In this paper we provided a preliminary experimental evaluation using a simple and non-intrusive implementation of these proposed modifications. We compared our algorithms with the VSIDS heuristics, the current state-of-the-art heuristic for SAT-solvers.

Our conclusions are that, in the benchmarks that we ran, VSIDS was observed to choose on a significant amount of irrelevant literals; as such, our proposed solver modification to VSIDS successfully managed to decrease the number of decisions made. However, we were not able to significantly reduce the number of conflicts, which would mean a reduction in the search space. Our hypothesis as to why the number of conflicts did not decrease with the number of decisions was confirmed using a crafted example. Furthermore, we sketched situations in which the decrease in the number of decisions alone is significant enough to improve performance compared to the current state-of-the-art.

Our notion of irrelevance is closely related to "don't care atoms" in satisfiability solving [Fu *et al.*, 2005]. However, there is an important difference between don't cares and irrelevant literals. To complete a partial structure with don't cares, *any* value may be assigned to a don't care literal; to an irrelevant literal, on the other hand, we only know that *some* value can be found for it. The for irrelevant literals can be found as follows: first, *any* value can be assigned to the irrelevant atoms that are open in Δ. Given these values to the opens, the (parametrised) well-founded model of Δ can be computed in polynomial time. The value of any other irrelevant literal is its value in the well-founded model. This is exactly what happens in the proof of Theorem 3.1.

We believe that further research into relevance will be of great value and see several topics for future work. First of all, the current theory is limited to PC(ID); further language extensions such as aggregates and arithmetic are not yet supported. Second, our theory also applies to generate-define-test ASP programs; experimentally evaluating relevance in a native ASP solver can yield interesting results. Third, engineering a more efficient algorithm to keep track of relevant literals can shed light on the possible (time-wise) performance gains from using relevance. Fourth, experimentally evaluating relevance in the context of lazy grounding is needed to verify our hypothesis that relevance can mean great improvements there.

Acknowledgements

This research was supported by project GOA 13/010 of the Research Fund KU Leuven and projects G.0489.10, G.0357.12, and G.0922.13 of the Research Foundation - Flanders. Bart Bogaerts is supported by the Finnish Center of Excellence in Computational Inference Research (COIN) funded by the Academy of Finland (under grant #251170).

References

[Apt, 2003] Krzysztof R. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[Barrett *et al.*, 2009] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. IOS Press, 2009.

[De Cat *et al.*, 2013] Broes De Cat, Bart Bogaerts, Jo Devriendt, and Marc Denecker. Model expansion in the presence of function symbols using constraint programming. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, November 4-6, 2013*, pages 1068–1075. IEEE Computer Society, 2013.

[De Cat *et al.*, 2015] Broes De Cat, Marc Denecker, Maurice Bruynooghe, and Peter J. Stuckey. Lazy model expansion: Interleaving grounding with search. *J. Artif. Intell. Res. (JAIR)*, 52:235–286, 2015.

[De Cat *et al.*, 2016] Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, Gerda Janssens, and Marc Denecker. Predicate logic as a modelling language: The IDP system. *CoRR*, abs/1401.6312v2, 2016.

[Denecker and De Schreye, 1993] Marc Denecker and Danny De Schreye. Justification semantics: A unifying framework for the semantics of logic programs. In Luís Moniz Pereira and Anil Nerode, editors, *LPNMR*, pages 365–379. MIT Press, 1993.

[Denecker and Ternovska, 2008] Marc Denecker and Eugenia Ternovska. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log.*, 9(2):14:1–14:52, April 2008.

[Denecker and Vennekens, 2007] Marc Denecker and Joost Vennekens. Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In Chitta Baral, Gerhard Brewka, and John S. Schlipf, editors, *LPNMR*, volume 4483 of *Lecture Notes in Computer Science*, pages 84–96. Springer, 2007.

[Denecker and Vennekens, 2014] Marc Denecker and Joost Vennekens. The well-founded semantics is the principle of inductive definition, revisited. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *KR*, pages 1–10. AAAI Press, 2014.

[Denecker *et al.*, 2015] Marc Denecker, Gerhard Brewka, and Hannes Strass. A formal theory of justifications. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, volume 9345 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2015.

[Denecker, 1998] Marc Denecker. The well-founded semantics is the principle of inductive definition. In Jürgen Dix, Luis Fariñas del Cerro, and Ulrich Furbach, editors, *JELIA*, volume 1489 of *LNCS*, pages 1–16. Springer, 1998.

[Fu *et al.*, 2005] Zhaohui Fu, Yinlei Yu, and S. Malik. Considering circuit observability don't cares in cnf satisfiability. In *Design, Automation and Test in Europe*, pages 1108–1113 Vol. 2, March 2005.

[Gebser *et al.*, 2012] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.*, 187:52–89, 2012.

[Kleene, 1938] S. C. Kleene. On notation for ordinal numbers. *The Journal of Symbolic Logic*, 3(4):150–155, 1938.

[Leone *et al.*, 2006] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.*, 7(3):499–562, 2006.

[Marek and Truszczyński, 1999] Victor Marek and Mirosław Truszczyński. Stable models and an alternative logic programming paradigm. In Krzysztof R. Apt, Victor Marek, Mirosław Truszczyński, and David S. Warren, editors, *The Logic Programming Paradigm: A 25-Year Perspective*, pages 375–398. Springer-Verlag, 1999.

[Mariën *et al.*, 2007] Maarten Mariën, Johan Wittocx, and Marc Denecker. Integrating inductive definitions in SAT. In Nachum Derschowitz and Andrei Voronkov, editors, *LPAR*, volume 4790 of *LNCS*, pages 378–392. Springer, 2007.

[Mariën *et al.*, 2008] Maarten Mariën, Johan Wittocx, Marc Denecker, and Maurice Bruynooghe. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In Hans Kleine Büning and Xishun Zhao, editors, *SAT*, volume 4996 of *LNCS*, pages 211–224. Springer, 2008.

[Mariën, 2009] Maarten Mariën. *Model Generation for ID-Logic*. PhD thesis, Department of Computer Science, KU Leuven, Belgium, February 2009.

[Marques-Silva and Sakallah, 1999] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[Stuckey, 2010] Peter J. Stuckey. Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In *CPAIOR*, pages 5–9, 2010.

[Verstichel, 2013] Jannes Verstichel. *The Lock Scheduling Problem (Het sluisplanningsprobleem)*. PhD thesis, Science, Engineering and Technology Group, Campus Kulak Kortrijk, Faculty of Science, Campus Kulak Kortrijk, Faculty of Engineering Science, November 2013. De Causmaecker, Patrick and Vanden Berghe, Greet (supervisors).