# *Lazy model expansion by incremental grounding*

BROES DE CAT and MARC DENECKER
*Department of Computer Science, K.U.Leuven, Belgium*
(*e-mail:* {broes.decat, marc.denecker}@cs.kuleuven.be)

PETER STUCKEY
*National ICT Australia, Victoria Laboratory,*
*Department of Computing and Information Systems,*
*University of Melbourne, Australia*
(*e-mail:* peter.stuckey@nicta.com.au)

*submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003*

**Abstract**

Model expansion is a widely accepted way to solve a range of problems. It is achieved by encoding the problem in a declarative (logic) language such that structures which satisfy the specification are solutions to the problem. Model expansion is researched in domains like Knowledge Representation and Answer Set Programming. The *ground-and-solve* methodology tackles model expansion by converting the predicate descriptions of the problem together with the problem instance data to propositional clauses and afterwards applying a model generator for propositional logic. Ground-and-solve is currently a state-of-the-art approach to model expansion, but a weakness of the approach is that the grounding step is only applicable when the problem instance is finite, and even then the grounding can be exponentially larger than the original description. In this paper we describe a *lazy* approach to grounding of problems specified in *first-order logic* that can cope with large and infinite problem domains. It interleaves grounding and propositional model expansion, and by keeping track whether models found for a subset of the full grounding solve the problem, it can forgo determining all propositions in the grounding. The approach is hence both more widely applicable and more efficient for problems where the grounding is large.

*KEYWORDS*: First-order logic, Model expansion, Lazy grounding

## 1 Introduction

Model expansion is a widely accepted way to solve a range of problems, by encoding the problem in a declarative (logic) language such that structures which satisfy the specification are solutions to the problem. Such inference techniques are researched within the domains of Constraint Programming (Apt 2003), Answer Set Programming (Niemelä 2006) and Knowledge Representation (Baral 2003).

A shared property of a number of such techniques is a duality between representation and model expansion. The problem specification on the one hand is formulated in a high-level (user-friendly) language. The search algorithm on the other hand usually only supports a more basic language. The link between the two is a process called *grounding* (also known as *unrolling*) which converts the high-level specification into low-level input. The high-level language FO(·) (Denecker and Ternovska 2008) for example is associated with its propositional fragment PC(·); accordingly, MiniZinc (Nethercote et al. 2007) is associated to Flatzinc, ASP to propositional ASP and First Order logic (FO) to propositional logic. Such an approach to model expansion is denoted as *ground-and-solve*, a state-of-the-art way to solve model expansion problems.

An important bottleneck to applying ground-and-solve is the size of the grounding. For FO for example, the size of the propositional theory is exponential in the nesting depth of quantifiers and polynomial in the size of the domain of the original theory. There are lots of practical problems in which it is intractable to generate the propositional theory.

Is it necessary to always perform grounding completely? Obviously the answer is no, as shown by a simple example: if a subset of the specification is inconsistent, it is not necessary to ground the remainder before the search algorithm can return unsatisfiable. Furthermore, there are a number of problem classes for which it is known that the full grounding is not required. An example are planning problems, which usually are modelled over some notion of time. Time is conceptually infinite, but only a finite plan is required to satisfy some goal. For general model expansion, we might not be interested in the full model, but only in some small subset, so again the full grounding might not be necessary to find such models. On the other hand, we need the guarantee that found solutions are consistent on the whole specification.

In this paper, we present a novel approach to remedy this bottleneck, namely to generate this grounding *lazily* (or *on-the-fly*) instead of up-front. The approach is presented for the language FO(TYPE), FO extended with a type system, and results are reported on an implementation in the IDP system, a knowledge base system supporting efficient model expansion.

## 2 Preliminaries

While FO is very expressive, from a knowledge representation point of view it has some well-known limitations, such as not being to naturally express transitive closure or the cardinality of a set of domain elements. To this end, FO(·) (Denecker and Ternovska 2008) denotes a broad class of extensions of FO, intended to be better suited for knowledge representation. A concrete instantiation of "·" denotes a particular extension; *FO(Agg)* for example extends FO with aggregate expressions.

The IDP system is a knowledge base system (Denecker and Vennekens 2008): it allows one to specify knowledge in a declarative way and to use a range of inference mechanisms to apply various reasoning tasks on the knowledge base. The language supported by IDP is an instantiation of FO(·) which extends FO with (among others) *types*, *arithmetic*, *aggregates* and *inductive definitions*. In this paper, we focus on the language FO(TYPE), a subset of that language, which extends FO with a type system. More information on the other extensions can be found in (Wittocx et al. 2008) and (Bogaerts et al. 2012).

## *2.1 Syntax and semantics*

We assume familiarity with classical logic. A vocabulary $\Sigma$ consists of a set of predicate and function symbols. Propositional symbols and constants are 0-ary predicate symbols, respectively function symbols. The Herbrand universe $\mathcal{HU}$ of atoms is defined as usual. FO terms and formulae are defined as usual, and are built inductively from variables, constant and function symbols, logical connectives ($\neg$, $\wedge$, $\vee$) and quantifiers ($\forall$, $\exists$).

A *ground sentence* is a sentence without quantifiers or variables. A *ground theory* is a theory consisting of ground sentences.

A vocabulary also contains *types*, which are interpreted in a structure as set of domain elements. Each variable has an associated type, as well as each argument of a predicate symbol and each argument and the range of a function symbol. As an example, consider the types $Machine$ and $Dollar$ and a function $cost(Machine) : Dollar$ which maps a machine to its cost in dollars. A variable $x$ with associated type $T$ is denoted by $x[\![T]\!]$.

Given a formula $\varphi$ with free variable $x$, substitution of $x$ with domain element $d$ is denoted as $\varphi[x/d]$. We only consider substitutions which replace a variable with a domain element belonging to its type.

Throughout the paper, $A$ and $L$ are used to refer to atoms, respectively literals.

The semantics of FO(Type) are an integration of standard FO semantics and the semantics of the type extension.

In this paper we deal with three-valued interpretations $I$ over some set of atoms $\mathcal{HU}$. We will sometimes write an interpretation $I$ as the set of literals over $\mathcal{HU}$ which are true in it. For example $I = \{P, \neg Q\}$, with $\mathcal{HU} = \{P, Q, R\}$, denotes the interpretation in which $P$ is true, $Q$ is false and $R$ is unknown.

Let $I(\varphi)$ denote the interpretation of a sentence $\varphi$ under $I$. Hence if $I \models \varphi$ we have $I(\varphi) = \mathbf{t}$, if $I \models \neg\varphi$ we have $I(\varphi) = \mathbf{f}$ and if neither $I \models \varphi$ nor $I \models \neg\varphi$ we have $I(\varphi) = \mathbf{u}$, that is $\varphi$ is unknown w.r.t $I$. The interpretation of a sentence under an interpretation $I$, denoted $I(\varphi)$ is defined as usual except that for existential quantification $I(\exists x[\![t]\!](\varphi)) = \mathbf{t}$ if and only if there is a $d \in t$ such that $I(\varphi[x/d]) = \mathbf{t}$; and $I(\exists x[\![t]\!](\varphi)) = \mathbf{f}$ if and only if for all $d \in t$ we have $I(\varphi[x/d]) = \mathbf{f}$. (Typed) universal quantification is defined similarly. For $\Rightarrow, \Leftarrow$ and $\equiv$ we take their interpretation to be the weak interpretation, which (as they are shorthands) coincides with the interpretation of their associated logical formula. Consequently, a formula $\varphi \Rightarrow \psi$ is true if $I(\varphi) = \mathbf{f}$ or $I(\psi) = \mathbf{t}$, false if $I(\varphi \wedge \psi) = \mathbf{t}$ and unknown otherwise; $\varphi \Leftarrow \psi$ is interpreted as $\psi \Rightarrow \varphi$; $\varphi \equiv \psi$ as $(\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$.

A (three-valued) interpretation $I$ is a *model* of an FO(Type) sentence if and only if the sentence is true under the interpretation. It is a model of an FO(Type) theory $\mathcal{T}$ if and only if it is a model of each of the sentences in $\mathcal{T}$.

An interpretation $I$ is *more precise* than an interpretation $I$' if and only if $I$ is identical to $I$' except on symbols which are unknown in $I$'. Using the set based view that is $I' \subseteq I$. Two interpretations $I$ and $J$ *agree on shared symbols* if there is no proposition $P$ where $\{P, \neg P\} \subseteq I \cup J$.

With a slight abuse of notation, given a theory $\mathcal{T}$, $\mathcal{T}$ is also used to refer to the conjunction of the sentences it contains (as opposed to the set of sentences).

### *2.2 Model expansion*

*Model generation* is the inference task of, given a vocabulary $\sigma_{in}$, a theory $\mathcal{T}$ over $\sigma_{in}$ and a (partial) interpretation $\mathcal{S}_{in}$ of $\sigma_{in}$, finding models $M$ which satisfy $\mathcal{T}$ and are more precise than $\mathcal{S}_{in}$. If $\mathcal{S}_{in}$ interprets all types, the inference task is *model expansion*, which will be denoted by $\langle \mathcal{T}, \mathcal{S}_{in} \rangle$. Model expansion can be used to solve problems by modelling them as a logical theory (and structure) such that solutions to the problem are models of the theory (Mitchell et al. 2006).

In this paper we consider *bounded* model expansion where also an "output" vocabulary $\sigma_{out}$ is given, a subset of $\sigma_{in}$. The idea is then to generate interpretations $I$ which are two-valued on $\sigma_{out}$ and can be extended to a model $M$ of the theory $\mathcal{T}$ more precise than $\mathcal{S}_{in}$. Conceptually, this comes down to problems where we are only interested in a (small) part of the solution, as long as we are guaranteed that a solution exists. Examples are satisfiability checking, where we are only interested in whether the problem has a solution at all (empty output vocabulary), or a planning problem where we are only interested in the first $n$ steps, as long as we are guaranteed some full plan exists (with the same $n$ steps) in which we can reach the goal. Obviously, model expansion can be cast as a bounded model expansion problem by setting $\sigma_{out}$ to $\sigma_{in}$. Bounded model expansion is denoted as $\langle \mathcal{T}, \mathcal{S}_{in}, \sigma_{out} \rangle$.

In the next sections, we present an approach for bounded model expansion over an empty output vocabulary. In section 4.5, the approach is extended (in a straightforward way) to non-empty output vocabularies.

In the rest of the paper, we assume the input theory $\mathcal{T}$ is build up from the basic connectives $\wedge, \vee, \equiv, \neg, \forall, \exists$. We assume that $\mathcal{T}$ is function-free and negations only occur explicitly in front of atoms.

An occurrence of a subformula $\varphi$ in $\mathcal{T}$ is called monotone if it is not in the scope of a negation and not in an equivalence. It is anti-monotone if in the scope of a negation (hence, must be an atom) and not in an equivalence. It is non-monotone if in an equivalence. This reflects the well-known property that increasing the truth value of an atom with only monotone occurrences, increases the truth value of formulae. If the atom has only anti-monotone occurrences, then increasing its truth value, decreases the value of formulae, and if it has both, or non-monotone occurrences, then it can be either way.

### *2.3 Grounding*

Basically, *grounding* is the process of instantiating all variables with domain elements to obtain a propositional theory. The *full grounding* of an FO(Type) formula $\psi$, $G_{full}(\psi)$, is defined by Table 1. The size of the full grounding is exponential in the nesting depth of quantifiers and polynomial in the size of the domains.

More intelligent grounding techniques exist, such as symbolic propagation (Wittocx 2010) and grounding with bounds (Wittocx et al. 2010).

| Original formula $\psi$ | Full grounding $G_{full}(\psi)$ |
|---|---|
| $P(\overline{d})$ | $P(\overline{d})$ |
| $\neg P(\overline{d})$ | $\neg P(\overline{d})$ |
| $\bigwedge_{i\in[1,n]} \varphi_i$ | $\bigwedge_{i\in[1,n]} G_{full}(\varphi_i)$ |
| $\bigvee_{i\in[1,n]} \varphi_i$ | $\bigvee_{i\in[1,n]} G_{full}(\varphi_i)$ |
| $\forall y[\![t]\!] : \varphi$ | $\bigwedge_{d\in t} G_{full}(\varphi[y/d])$ |
| $\exists y[\![t]\!] : \varphi$ | $\bigvee_{d\in t} G_{full}(\varphi[y/d])$ |
| $\varphi \equiv \varphi'$ | $G_{full}(\varphi) \equiv G_{full}(\varphi')$ |

Table 1. The definition of the full grounding $G_{full}(\psi)$.

## 3 Delayed theories

*Lazy grounding* (lazy mx) is an approach to interleave grounding and search. The key idea of our approach is to partly ground the input theory and to delay grounding of the remainder as long as certain conditions on the partial interpretation are satisfied. We call such conditions *delays.*

*Example 1*
Consider the formula $L_1 \vee \varphi$ where $L_1$ is a ground literal and $\varphi$ a formula. As long as $L_1$ is not made false during search, $L_1$ may still become true in the computed solution and the formula will be satisfied. Therefore, we can delay the grounding of $\varphi$ by replacing it with the Tseitin symbol $T$ for $\varphi$, resulting in a propositional clause $L_1 \vee T$ and the (non-propositional) formula $T \equiv \varphi$. The latter formula then only needs to be grounded when $T$ becomes true, for example by unit propagation when $L_1$ is made false. This condition on the partial structure $I$, $I(T) \neq \mathbf{t}$, is called the *delay.* As long as it satisfied, $T \equiv \varphi$ need not be grounded. □

Delayed grounding can can lead to exponential reduction in the size of the grounding. If for example $\varphi$ is the formula $\forall \overline{x} : \chi(\overline{x})$, quantifier instantiation can be completely avoided if $L_1$ is not false.

### *3.1 Delays on formulae*

A *delayed sentence* has the form $(\varphi)^\delta$ where $\varphi$ is a sentence and $\delta$ is a delay condition (briefly, a delay). Two types of delays are considered:

- A *true-delay*, denoted $\varphi \neq \mathbf{t}$, is satisfied in an interpretation $I$ iff $I(\varphi) \neq \mathbf{t}$.
- A *known-delay*, denoted $\varphi = \mathbf{u}$, is satisfied in an interpretation $I$ iff $I(\varphi) = \mathbf{u}$.

As a notational convenience, we write $\varphi \models \delta$ for delay condition $\delta$ as an obvious shorthand for $I(\varphi) \neq \mathbf{t}$ or $I(\varphi) = \mathbf{u}$.

A *delayed theory* $\mathcal{T}_d$ is a theory consisting of a ground theory $\mathcal{G}$ and a set $\mathcal{D}$ of *delayed sentences.* We will often denote it by $\langle \mathcal{G}, \mathcal{D} \rangle$.

The delayed theory $\mathcal{T}_d$ is constructed by a grounding algorithm which partially grounds the theory, resulting in $\mathcal{G}$, and delays the grounding of certain (sub)formulae, resulting in $\mathcal{D}$, a residual non-ground theory.

We say that a partial interpretation $I$ satisfies (is a model of) a delayed sentence $(\varphi)^{\delta}$ if $I$ satisfies $\varphi$. We say that $I$ weakly satisfies $(\varphi)^{\delta}$ if $I \models \delta$ or else, $I \models \varphi$.

We will say that a delayed sentence $(\phi)^{\delta}$ is *active* in $I$ if its condition $\delta$ is not satisfied, otherwise it is *inactive*. Conceptually, grounding will be triggered when a sentence becomes active.

*Definition 1* (*Weak satisfaction*)
A partial interpretation $I$ *weakly satisfies* (is a *weak model* of) a delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$ iff it satisfies each sentence in $\mathcal{G}$ and weakly satisfies each sentence in $\mathcal{D}$.

The lazy model expansion algorithm iteratively reduces delayed theories into "more ground" delayed theories. The following definition expresses the main invariant of our algorithm.

*Definition 2* (*Partial grounding*)
A delayed theory $\mathcal{T}_d$ is a *partial grounding* of a theory $\mathcal{T}$ iff

- $\mathcal{T}$ and $\mathcal{T}_d$ are "logically equivalent", in the sense that each 2-valued model $M$ of $\mathcal{T}$ can be extended to a model of $\mathcal{T}_d$ and vice versa, each 2-valued model of $\mathcal{T}_d$ satisfies $\mathcal{T}$.
- Each partial interpretation $I$ that weakly satisfies $\mathcal{T}_d$ has a two-valued extension $M$ that satisfies $\mathcal{T}$.

*Example 2*
Consider again the sentence $\mathcal{T} = L_1 \lor \varphi$. The delayed theory $\mathcal{T}_d$ consisting of the ground proposition $L_1 \lor T$ and the delayed sentence $(T \equiv \varphi)^{T \neq \mathbf{t}}$ is a partial grounding of $\mathcal{T}$. Indeed, (1) the two theories are obviously equivalent modulo the difference in vocabulary. (2) Any partial interpretation $I$ that weakly satisfies $\mathcal{T}_d$, satisfies either $L_1$ or $T$. In the first case, any extension of $I$ satisfies $\mathcal{T}$. In the second case, $I$ satisfies $T$ and the active sentence $T \equiv \varphi$, hence also $\varphi$ and therefore $\mathcal{T}$. $\square$

## 4 Introducing delayed sentences

The lazy grounding algorithm described in section 4.5 iteratively transforms an initial theory $\mathcal{T}$ into a sequence of partial groundings with a monotonically increasing ground theory. Note that an initial theory $\mathcal{T}$ trivially corresponds to the delayed theory $\left\langle \emptyset, \{(\varphi)^{\mathbf{t} \neq \mathbf{t}} \mid \varphi \in \mathcal{T}\} \right\rangle$ with empty ground theory and all its formulae active. The process is driven by delayed sentences becoming active in a partial interpretation $I$. It may stop when a solution is found, or when unsatisfiability is discovered, or, in case of infinite domains, it may run forever.

### *4.1 Tseitin introduction*

Recall from the initial example that $L_1 \lor \varphi$ was partially grounded to the ground formula $L_1 \lor T$ and a delayed sentence $((T \equiv (\varphi)))^{T \neq \mathbf{t}}$. We here describe this operation.

*Definition 3* (*Tseitin introduction*)
Given a delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$ and a set of occurrences of a formula $\varphi$ in sentences $\psi$ with $(\psi)^\delta \in \mathcal{T}_d$, the Tseitin introduction for $\varphi$ in $\mathcal{T}_d$ is the delayed theory $\mathcal{T}_d$'=$\langle \mathcal{G}, \mathcal{D}' \rangle$ where $\mathcal{D}$' is obtained from $\mathcal{D}$ as follows:

- by substituting a new Tseitin propositional symbol $T_\varphi$ for each of the selected occurrences of $\varphi$ in $\mathcal{D}$, and
- by adding a new delayed sentence $(T_\varphi \equiv \varphi)^\delta$. Here $\delta$ is determined as follows:
  - If all selected occurrences of $\varphi$ are monotone in $\mathcal{D}$, then $\delta = (T_\varphi \neq \mathbf{t})$.
  - If all are anti-monotone, then $\delta = (\neg T_\varphi \neq \mathbf{t})$.
  - Otherwise, $\delta = (T_\varphi = \mathbf{u})$.

*Lemma 1*
Let $\mathcal{T}_d$ be a partial grounding of $\mathcal{T}$. Let $\mathcal{T}_d'$ be the Tseitin introduction for a set of occurrences of $\varphi$ in sentences $\psi$ with $(\psi)^\delta \in \mathcal{T}_d$. Then (A) $\mathcal{T}_d$' is a partial grounding of $\mathcal{T}$. (B) If $(\psi)^\delta \in \mathcal{T}_d$ is transformed in $(\psi')^\delta$ in $\mathcal{T}_d$', then $T_\varphi \equiv \varphi \models \psi \equiv \psi'$.

*Proof*
(A) Condition 1, that $\mathcal{T}$ and $\mathcal{T}_d$' are equivalent in the shared vocabulary (i.e., in all symbols except $T_\varphi$), is a well-known property of the Tseitin transformation. The fact that any delayed sentence $\psi$ in $\mathcal{T}_d$ is only transformed into a delayed sentence $\psi$' in $\mathcal{T}_d$' by applying Tseitin transformation, also proves (B).

(A) For condition 2, let $I$ be a partial interpretation weakly satisfying $\mathcal{T}_d$'. Notice that if $(\psi')^\delta \in \mathcal{T}_d'$ was derived from $(\psi)^\delta \in \mathcal{T}_d$ by Tseitin introduction, then if one is active in $I$ so is the other. We show that $I$', the restriction of $I$ to symbols different than $T_\varphi$, weakly satisfies $\mathcal{T}_d$. Hence, it can be extended to a 2-valued model $M$ of $\mathcal{T}$. If $I(T_\varphi) \neq \mathbf{u}$, we then set $M(T_\varphi) := I(T_\varphi)$; if $I(T_\varphi) = \mathbf{u}$, we set $M(T_\varphi) := I(\varphi)$. Either way, the property that $I$ is a model of $\mathcal{T}$ is maintained and $M$ extends $I$.

If $(T_\varphi \equiv \varphi)^\delta$ is active and hence satisfied in $I$, it is obvious that $I$ weakly satisfies $\mathcal{T}_d$ since $I(T_\varphi) = I(\varphi) \neq \mathbf{u}$. Let us consider the three cases where the delay is inactive.

Assume that $I(T_\varphi) = \mathbf{u}$. If $(\psi')^\delta \in \mathcal{T}_d'$ is active and was derived from $(\psi)^\delta \in \mathcal{T}_d$, then $\psi'$ is obtained from $\psi$ by replacing certain subformulae by one that is $\mathbf{u}$ in $I$. It follows that $I \models \psi$. Thus, $I$ weakly satisfies $\mathcal{T}_d$, and it can be extended to a 2-valued model $M$ of $\mathcal{T}_d$.

The other cases are when $I(T_\varphi) = \mathbf{f}$ and only monotone occurrences of $\varphi$ were replaced, or when $I(T_\varphi) = \mathbf{t}$ and only anti-monotone occurrences were replaced. Both cases are similar, so let's consider only the first one. Because $T_\varphi$ is false and has only monotone occurrences in $\mathcal{T}_d$, $I$' satisfies all active sentences of $\mathcal{T}_d$ (since $I(T_\varphi) \leq I(\varphi)$). Hence, $I$' can be extended to a model $M$ of $\mathcal{T}$. □

One can prove that if Tseitin introduction is applied only to monotone occurrences of $\varphi$, then introduced delayed sentence can be weakened to $(T_\varphi \Rightarrow \varphi)^{T_\varphi \neq \mathbf{t}}$. If it is only applied to anti-monotone occurrences, we can use $(T_\varphi \Leftarrow \varphi)^{\neg T_\varphi \neq \mathbf{t}}$. We will not detail this.

### *4.2 Delayed grounding algorithm*

We now present the delayed grounding algorithm $\mathsf{for\_del\_gnd}$. Given a theory $\mathcal{T}$, the result is a delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$ which is a partial grounding of $T$.

The delayed grounding recursively examines formulae in $\mathcal{T}$, choosing disjuncts and existentially quantified formulae to apply Tseitin introduction on. The algorithm also keeps track of the *context* of the formula, which is monotone (`mono`) or non-monotone (`nm`).[1] The context is used to build an appropriate delay using the helper function $\mathsf{kind}$ defined as:

$$\mathsf{kind}(\mathtt{mono}, \phi) = (\phi \neq \mathbf{t})$$
$$\mathsf{kind}(\mathtt{nm}, \phi) = (\phi = \mathbf{u})$$

The algorithm constructs a ground theory in general propositional logic, but many practical solvers take input in clausal normal form (CNF) instead. Adapting $\mathsf{for\_del\_gnd}$ to return a CNF is straightforward, by applying the Tseitin transformation on the ground sentences (on-the-fly or as a post-processing step).

The algorithm makes use of a *current* interpretation $I$ and of $I_{entail}$, the set of literals true in $\mathcal{S}_{in}$ or by unit propagation on $\mathcal{G}$ starting from $\mathcal{S}_{in}$ (that is literals that are known to be globally true independent of decisions in search). $I_{entail}$ is used to simplify formulae which are certainly true/false, $I$ allows to first select subformulae which are not currently known, preventing immediate activation of newly introduced Tseitin literals (this condition can be relaxed if it is too expensive to check).

We define the conjunction $\langle \mathcal{G}, \mathcal{D} \rangle \wedge \langle \mathcal{G}', \mathcal{D}' \rangle$ of two delayed theories $\langle \mathcal{G}, \mathcal{D} \rangle$ and $\langle \mathcal{G}', \mathcal{D}' \rangle$ as $\langle \mathcal{G} \wedge \mathcal{G}', \mathcal{D} \cup \mathcal{D}' \rangle$.

```
for_del_gnd($\phi$, $I$, $ctxt$)
    if ($I_{entail}(\phi) = \mathbf{t}$) return $\langle true, \emptyset \rangle$
    if ($I_{entail}(\phi) = \mathbf{f}$) return $\langle false, \emptyset \rangle$
    switch $\phi$
        $P(\bar{d})$ :                    return $\langle P(\bar{d}), \emptyset \rangle$
        $\neg P(\bar{d})$ :               return $\langle \neg P(\bar{d}), \emptyset \rangle$
        $\bigvee_{i \in [1,m]} \varphi_i$ : select $j \in [1,m]$ (with $I(\varphi_j) \neq \mathbf{f}$ if possible)
                                          $\psi := \bigvee_{i \in [1,m] \setminus j} \varphi_i$
                                          $\langle \mathcal{G}_j, \mathcal{D}_j \rangle$ = for_del_gnd($\varphi_j$, $I$, $ctxt$)
                                          return $\langle \mathcal{G}_j \vee T_\phi, \mathcal{D}_j \cup \{(T_\psi \equiv \psi)^{\mathsf{kind}(ctxt, T_\phi)}\} \rangle$
        $\exists y[\![t]\!] : \varphi$ :  select $d \in t$ (with $I(\varphi[y/d]) \neq \mathbf{f}$ if possible)
                                          $\psi := \exists y[\![t \setminus d]\!] : \varphi$
                                          $\langle \mathcal{G}_d, \mathcal{D}_d \rangle$ = for_del_gnd($\varphi[y/d]$, $I$, $ctxt$)
                                          return $\langle \mathcal{G}_d \vee T_\phi, \mathcal{D}_d \cup \{(T_\phi \equiv \psi)^{\mathsf{kind}(ctxt, T_\psi)}\} \rangle$
        $\bigwedge_{i \in [1,m]} \varphi_i$ : return $\langle true, \emptyset \rangle$ $\bigwedge_{i \in [1,m]}$ for_del_gnd($\varphi_i$, $I$, $ctxt$)
        $\forall y[\![t]\!] : \varphi$ :  return $\langle true, \emptyset \rangle$ $\bigwedge_{d \in t}$ for_del_gnd($\varphi[y/d]$, $I$, $ctxt$)
        $\varphi_l \equiv \varphi_r$:     $\langle \mathcal{G}_l, \mathcal{D}_l \rangle$ := for_del_gnd($\varphi_l$, $I$, nm)
                                          $\langle \mathcal{G}_r, \mathcal{D}_r \rangle$ := for_del_gnd($\varphi_r$, $I$, nm)
```

[1] ←Recall, formulae (except atoms) only occur monotonically or non-monotone.

```
        **return** $\langle \mathcal{G}_l \equiv \mathcal{G}_r, \mathcal{D}_l \cup \mathcal{D}_r \rangle$
```

The result of for_del_gnd is guaranteed to be a correct partial grounding of $\mathcal{T}$.

*Theorem 1*
Let $\mathcal{T}$ be a theory and $\mathcal{T}_d$ = for_del_gnd ($\mathcal{T}$, $I$, `mono`). Then $\mathcal{T}_d$ is a partial grounding of $\mathcal{T}$ and all delays in $\mathcal{T}_d$ are inactive under $I$.

*Proof*
Follows from Proposition 1 and the fact that all introduced literals $T_\phi$ are new and hence $I(T_\phi) = \mathbf{u}$. □

*Example 3*
Consider $T = \exists x[\![t]\!] : (P(x) \wedge R(x)) \vee (\forall y[\![t']\!] : Q(x, y))$. Delayed grounding of this sentence is achieved by selecting a domain element $d \in t$ and applying for_del_gnd to $(P(d) \wedge R(d)) \vee (\forall y[\![t']\!] : Q(d, y))$ while applying Tseitin introduction to the residual subformula $\phi = \exists x[\![t \backslash d]\!] : (P(x) \wedge R(x)) \vee (\forall y[\![t']\!] : Q(x, y))$.

Applying for_del_gnd to $(P(d) \wedge R(d)) \vee (\forall y[\![t']\!] : Q(d, y))$ recursively calls for_del_gnd on $P(d) \wedge R(d)$ and Tseitin introduction on the other disjunct $\psi = (\forall y[\![t']\!] : Q(d, y))$. The resulting delayed theory consists of the ground sentence $(P(d) \wedge R(d)) \vee T_\psi \vee T_\phi$ and the true-delayed sentences:

$$(T_\psi \equiv (\forall y[\![t']\!] : Q(d, y)))^{T_\psi \neq \mathbf{t}}$$
$$(T_\phi \equiv \exists x[\![t \backslash d]\!] : (P(x) \wedge R(x)) \vee (\forall y[\![t']\!] : Q(x, y)))^{T_\phi \neq \mathbf{t}}$$

True-delays suffice as the associated formulae occur monotonically in $\mathcal{T}$. □

### *4.3 Incremental delayed grounding*

In order to construct a weak model for a delayed theory $\mathcal{T}_d$, search and grounding are interleaved. When, during search, an interpretation $I$ is constructed where some delays in $\mathcal{T}_d$ are active, further grounding needs to be applied to the associated delayed sentences.

The function inc_del_gnd takes a delayed theory $\mathcal{T}_d = \langle \mathcal{G}_0, \mathcal{D}_0 \rangle$ which is a partial grounding of $\mathcal{T}$ and an interpretation $I$, and returns a new delayed theory $\mathcal{T}_d$', also a partial grounding of $\mathcal{T}$, for which all delays are inactive in $I$. The function iterates over the delayed sentences and applies delayed grounding to those with an active delay. The delay condition itself is used to determine if its Tseitin literal appears monotonically or not.

```
inc_del_gnd ($\langle \mathcal{G}_0, \mathcal{D}_0 \rangle$,$I$)
    $\mathcal{G} := \mathcal{G}_0$; $\mathcal{D} := \emptyset$
    **for each** $(T_\phi \equiv \phi)^\delta \in \mathcal{D}_0$
        **if** ($\delta$ is inactive in $I$) $\mathcal{D} := \mathcal{D} \cup \{(T_\phi \equiv \phi)^\delta\}$
        **else**  **if** ($\delta = T_\phi \neq \mathbf{t}$) $ctxt$ := mono **else** $ctxt$ := nm
              $\langle \mathcal{G}_n, \mathcal{D}_n \rangle$ := for_del_gnd($\phi, I, ctxt$)
        $\mathcal{G} := \mathcal{G} \wedge (T_\phi \equiv \mathcal{G}_n)$; $\mathcal{D} := \mathcal{D} \cup \mathcal{D}_n$
    **endfor**
    **return** $\langle \mathcal{G}, \mathcal{D} \rangle$
```

## *4.4 Satisfiability delays*

Another approach to introducing delays applies to sentences of which a condition on their satisfiability can be derived. For some classes of formulae such conditions are well-known:

*Example 4*
Consider the definite clause $\forall \overline{x}[\![\overline{t}]\!] : \phi[\overline{x}] \Rightarrow Q(\overline{x})$ where $\phi[\overline{x}]$ is the body of the clause (negations have not been pushed inside for clarity). Any interpretation $I$ in which none of the (ground) *heads* $Q(\overline{d})$ are false can be extended to an interpretation which satisfies all clauses, namely the interpretation in which all heads are true. Consequently, only instantiations of which the head is false in $I$ need to be ground. All remaining heads can then be delayed as they can still be satisfied.

Assume that only $I(Q(\overline{d})) = \mathbf{f}$ for some $\overline{d} \in \overline{t}$. The delayed theory $\mathcal{T}_d$

$$\left\langle \phi[\overline{x}/\overline{d}] \Rightarrow Q(\overline{d}),\ \{(\forall \overline{x}[\![\overline{t} \backslash \overline{d}]\!] : \phi(\overline{x}) \Rightarrow Q(\overline{x}))^{\exists \overline{x}[\![\overline{t}\backslash\overline{d}]\!] : \neg Q(\overline{x}) \neq \mathbf{t}}\} \right\rangle$$

is then a partial grounding of the definite clause under $I$. □

In this section, we present an approach to delay sentences of universally quantified disjunctions based on their satisfiability. Delaying universally quantified disjunctions is not captured by Tseitin introduction and is a class of formulae which occur often in practice. Extending the approach to other and more general classes of formulae is part of future work.

*Definition 4* (*Satisfiability delaying*)
Consider a delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$, a partial grounding of $\mathcal{T}$, and an interpretation $I$. Assume $(\psi)^\delta \in \mathcal{D}$, with $\psi = \forall \overline{x}[\![\overline{t}]\!] : \bigvee_{i \in [1,m]} \varphi_i$.[2] Satisfiability delaying of $\psi$ for $\mathcal{T}_d$ under $I$ consists of selecting a subset $S_d$ of $\bigcup_{i \in [1,m]} \varphi_i$ such that

- Each formula in $S_d$ is a literal.
- No delay in $\mathcal{T}_d$ contains a literal over the same symbol with the opposite sign as any literal in $S_d$.

The set *nd* of *non-delayable instantiations* is the set of tuples of domain elements $\overline{d} \in \overline{t}$ for which all formulae in $S_d$ are false in $I$. The delay condition $\chi$ is then is $\chi = \exists \overline{x}[\![\overline{t} \backslash nd]\!] : \bigwedge_{\varphi_i \in S_d} \neg \varphi_i \neq \mathbf{t}$.

Assume the grounding of the non-delayable instantiations $\forall \overline{x}[\![nd]\!] \bigvee_{i \in [1,m]} \varphi_i$ results in the delayed theory $\langle \mathcal{G}_{rem}, \mathcal{D}_{rem} \rangle$. The delayed theory $\mathcal{T}'_d$ is then constructed as $\left\langle \mathcal{G} \wedge \mathcal{G}_{rem}, \mathcal{D} - \{\psi\} \cup \mathcal{D}_{rem} \cup (\forall \overline{x}[\![\overline{t} \backslash nd]\!] \bigvee_{i \in [1,m]} \varphi_i)^\chi \right\rangle$. We then say that $\mathcal{T}'_d$ is *satisfiability delayed* on $\psi$.

It should be noted that whether a satisfiability delay can contain a literal over a symbol $P$ depends on occurrences of $P$ in existing delays.

[2] ←A delayed sentence $(\varphi \wedge \varphi')^\delta$ can be seen as the union of delayed sentences $(\varphi)^\delta$ and $(\varphi')^\delta$.

*Example 5*
Consider the theory consisting of the sentences $\forall\overline{x}[\![\overline{t}]\!] : P(\overline{x}) \lor Q(\overline{x})$ and $\forall\overline{x}[\overline{t}] : \neg P(\overline{x})$. Delaying the former on $\exists\overline{x}[\![\overline{t}]\!] : \neg P(\overline{x})$ is not correct if the latter is delayed on $\exists\overline{x}[\![\overline{t}]\!] : P(\overline{x})$. Indeed, consider an interpretation $I$ in which all literals over $P$ are unknown and some literal over $Q$ is false. In that case, $I$ is a weak model but cannot be extended to a model of the theory. $\square$

*Proposition 1*
Let $\mathcal{T}_d$ be a delayed theory on $\mathcal{T}$ which contains the delayed sentence $(\psi)^\delta$ such that a subset exists for $\psi$ satisfying the above-described conditions. Let $\mathcal{T}_d$' be the delayed theory resulting from satisfiability delaying $\mathcal{T}_d$ on $\psi$ under $I$. Then (A) $\mathcal{T}_d$' is a partial grounding of $\mathcal{T}$. Moreover (B) if $(\psi)^\delta \in \mathcal{T}_d$ was transformed into $(\psi')^\delta \in \mathcal{T}_d'$ then $\psi \models \psi'$.

*Proof*
(A) Condition 1, equivalence of $\mathcal{T}_d$' and $\mathcal{T}$, directly follows from the fact that the conjunction $\psi$ is split and each conjunct is either added as a sentence in $\mathcal{G}$ or as a new delayed sentence.

(A) Condition 2: Assume $(\psi)^\delta$ is a delayed sentence in $\mathcal{T}_d = \langle\mathcal{G}, \mathcal{D}\rangle$ and is satisfiability delayed by $\psi$. Take $(\psi')^{\delta'}$ to be the delayed sentence introduced by this process to create $\mathcal{T}_d' = \langle\mathcal{G}', \mathcal{D}'\rangle$. Then $\mathcal{D}' - \{(\psi')^{\delta'}\} = \mathcal{D} - \{(\psi)^\delta\}$ and $\mathcal{G}' \geq \mathcal{G}$.

Now consider any weak model $M$ of $\mathcal{T}_d$'. If $\delta$ is inactive in $M$, all delayed sentences $\mathcal{T}_d$ are weakly satisfied in $M$, so $M$ is a weak model of $\mathcal{T}_d$. If $\delta$ is active in $M$, the proof is split up. Case 1: $\delta$' is active. $M$ then satisfies $\psi$, so all delayed sentences in $\mathcal{T}_d$ are weakly satisfied; consequently $M$ is a weak model of $\mathcal{T}_d$. Case 2: $\delta$' is inactive. In this case, $\psi$ might not be satisfied in $M$. Take $(\psi')^{\delta'}$ to be $(\forall\overline{x}[\overline{t}] : \bigvee_{i\in[1,m]} \varphi_i)^{\exists\overline{x}[\overline{t}']:\bigwedge_{i\in S}\neg\varphi_i}$. For each $\overline{d}$in $\overline{t}$, either $\bigvee_{i\in[1,m]} \varphi_i[\overline{x}/\overline{d}]$ is satisfied in $M$ or some $\varphi_j[\overline{x}/\overline{d}]$, $j \in S$, is not false in $M$ (otherwise $\delta$' would be active). $\varphi_j[\overline{x}/\overline{d}]$ is a ground literal such that no literal on the same symbol and with the opposite sign occurs in any delay in $\mathcal{T}_d$. Construct $M$' as identical to $M$ except that $\varphi_j[\overline{x}/\overline{d}]$ is true. No delay not active in $M$ can be active in $M$' as the negation of the literal does not occur in any delay. Applying this idea to all relevant $\overline{d}$, $M$" is constructed which satisfies $\psi$, so $M$" and $M$ are a weak models of $\mathcal{T}_d$.

(B) follows trivially from the way the new delayed sentence is constructed. $\square$

Satisfiability delaying can be extended to other classes of formulae, such as known-delaying formulae of the form $\forall\overline{x}[\![\overline{t}]\!] : L(\overline{x}) \equiv \varphi(\overline{x})$. It can also be extended to *inductive definitions* (Denecker and Ternovska 2008), sets of rules of the form $\forall\overline{x} : P(\overline{x}) \leftarrow \phi$ evaluated by the well-founded semantics(Van Gelder 1993), by known-delaying their heads. Delaying of inductive definitions opens up possibilities towards for example transitive closure and more importantly towards Answer Set Programming. The formalisation is out of the scope of this paper, but the techniques are included in the prototype implementation used in the experiments.

### *4.4.1 Updated delayed grounding algorithms*

To include satisfiability delaying in the grounding algorithms, information on the full set of delays is required. To this end, all calls to for_del_gnd, both those in inc_del_gnd and the recursive ones in for_del_gnd have the current set of delayed sentences $\mathcal{D}$ as an additional parameter.[3] The incremental algorithm is adapted to also handle satisfiability delayed sentences in a straightforward way. Furthermore, an additional case is added to the switch statement of for_del_gnd, before the step which grounds universal quantifications.

```
$\forall \overline{y}[\![\overline{t}]\!] : \bigvee_{i \in [1,m]} \varphi_i$ :
    **if** ($\varphi$ can be satisfiability delayed under $\mathcal{D}$ and $I$ by set $S \subseteq [1,m]$)
        $nd := \{\overline{d} \mid \overline{d} \in \overline{t}, I(\bigwedge_{j \in S} \neg\varphi_j[\overline{y}/\overline{d}]) = \mathbf{t}\}$
        $\mathcal{D} := \mathcal{D} \cup (\forall \overline{y}[\![\overline{t} \backslash nd]\!] : \bigvee_{i \in [1,m]} \varphi_i)^{\exists \overline{y}[\![\overline{t} \backslash nd]\!] : \bigwedge_{j \in S} \neg\varphi_j \neq \mathbf{t}}$
        **return** $\langle true, \mathcal{D} \rangle \ \bigwedge_{\overline{d} \in nd}$ for_del_gnd($\varphi_i[\overline{y}/\overline{d}]$, $I$, $ctxt$, $\mathcal{D}$)
    **else** continue to original universal quantification case
```

## *4.5 Lazy model expansion*

Lazy model expansion interleaves grounding and search. Given delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$, the ground theory $\mathcal{G}$ is added to the search algorithm as constraints. During the standard search algorithm, changes to the interpretation are checked against the delays and grounding can be applied in the middle of the search. Note that *crucially* lazy grounding steps are not undone on backtracking. The delayed theory becomes more and more ground as execution proceeds.

The algorithm gets as input a theory $\mathcal{T}$ and a pre-interpretation $S_{in}$, and constructs the initial delayed theory $\langle \mathcal{G}, \mathcal{D} \rangle$, a partial grounding of $\mathcal{T}$. The algorithm proceeds by performing unit propagation on the ground part of the delayed theory. If a conflict is detected it backjumps as usual, adding a conflict clause. If the conflict is at the root level then $\mathcal{G}$ has no model and hence neither does $\mathcal{T}$ since $\langle \mathcal{G}, \mathcal{D} \rangle$ is a partial grounding of $\mathcal{T}$. If a delay is active, it performs grounding to construct a new delayed theory which is a partial grounding of $\mathcal{T}$. If the search algorithm detects the current interpretation certainly satisfies all constraints in $\mathcal{G}$, a weak model for the current delayed theory $\langle \mathcal{G}, \mathcal{D} \rangle$ has been found. As it is a partial grounding of $\mathcal{T}$, we have proven the satisfiability of $\mathcal{T}$. The strength of such detection depends on the search algorithm used; the standard approach halts when all literals in the grounding are true or false, but stronger methods can empower e.g. known-delays.[4] Otherwise we make a new search choice and continue.

[3] ←For conjunctions and universal quantifications, the delayed sentences returned by one recursive call are used as input to the next recursive call.

[4] ←In the standard approach, known-delays are useless as they are certainly decided (unless they do not occur in the ground theory). Our own implementation uses a mechanism based on reducing the number of decision literals.

```
lazy_mx ($\mathcal{T}$, $I$)
    $\langle \mathcal{G}, \mathcal{D} \rangle$ := for_del_gnd($\mathcal{T}$, $I$, mono)
    while $true$ do
        $I$ := unit_propagation($\mathcal{G}$,$I$)
        if (conflict detected)
            if (at root level) return $false$
            $\mathcal{G}$ := $\mathcal{G} \wedge$ conflict clause
            $I$ := $I$ at state of backjump point
        else if (some delay in $\mathcal{D}$ is active in $I$)
            $\langle \mathcal{G}, \mathcal{D} \rangle$ := inc_del_gnd ($\langle \mathcal{G}, \mathcal{D} \rangle$,$I$)
        else if (satisfaction of $\mathcal{G}$ in $I$ is detected) return $true$
        else $I := I \cup \{l\}$ with $l$ a search choice
    endwhile
```

*Theorem 2* (*Correctness and termination*)
If algorithm lazy_mx returns $true$, $\mathcal{T}$ has a model which is more precise than $I$. If the algorithm returns $false$, no interpretation exists which is more precise than $I$ and satisfies $\mathcal{T}$. Algorithm lazy_mx terminates if $\mathcal{T}$ and $I$ are finite. If $\mathcal{T}$ has a finite number of sentences, termination is possible but not guaranteed.

Proofs are omitted due to lack of space.

The algorithm can be adapted to handle non-empty output vocabularies $\sigma_{out}$. A post-condition on the algorithm is that any model found is two-valued on $\sigma_{out}$. Such a property can, for example, be ensured by adding the disjunction $A \vee \neg A$ to the initial ground theory for each atom $A$ in $\sigma_{out}$.

## 5 Experiments

A prototype implementation was created within the new IDP-3 system and experiments were conducted with three different setups: basic model expansion (denoted IDP), lazy model expansion by Tseitin introduction ($\text{IDP}_T$) and by Tseitin introduction and satisfiability delaying ($\text{IDP}_{T,S}$).

The considered benchmarks represent a diverse set of problems, both existing benchmarks (e.g. from previous ASP competitions) and newly constructed ones. As most instances of existing benchmarks are hard problems geared towards grounding systems and propagation, we also used new instances geared towards "easier" problems with a very large grounding. It will show the strength and applicability of lazy mx, while at the same time clarifying that the technique is less well suited if propagation is crucial to solve it efficiently.

A timeout of 1000 seconds was used and a memory limit of 3 Gb; — is used to indicate that an instance could not be solved within those bounds. [5]

For each benchmark instance, the size of the full grounding ($g(full)$) and of the

[5] ←All experiments were run on an Intel Core 2 Machine (dual 2.40Ghz) running Ubuntu 10.4. The tested version of the IDP system and all data files are available from `http://dtai.cs.kuleuven.be/krr/research/experiments`.

| Benchmark | $g(full)$ | $g(\text{IDP})$ | $g(\text{IDP}_T)$ | $g(\text{IDP}_{T,S})$ | $t(\text{IDP})$ | $t(\text{IDP}_T)$ | $t(\text{IDP}_{T,S})$ |
|---|---|---|---|---|---|---|---|
| func-1 | $8.0 * 10^7$ | $8.0 * 10^7$ | $1.6 * 10^5$ | 540 | 99.03 | 4.07 | 0.1 |
| func-2 | $\infty$ | — | — | 1370 | — | — | 0.1 |
| bnq** | $1.4 * 10^8$ | $1.1 * 10^5$ | $1.1 * 10^5$ | $6.8 * 10^4$ | 2.56 | 2.56 | 1.96 |
| packing-1 | $1.0 * 10^{10}$ | $1.2 * 10^8$ | $1.1 * 10^8$ | $1.0 * 10^6$ | 171 | 172 | 5.0 |
| packing-2 | $3.1 * 10^{12}$ | — | — | $2.2 * 10^7$ | — | — | 27.0 |
| agentK | $5.0 * 10^6$ | — | — | 626 | — | — | 0.02 |
| planning1 | $\infty$ | — | — | 385 | — | — | 0.29 |
| planning2-1 | $3.0 * 10^8$ | $2.0 * 10^8$ | $.05 * 10^6$ | $4.3 * 10^4$ | 139.02 | 5.96 | 0.46 |
| planning2-2 | $3.0 * 10^{10}$ | — | $5.1 * 10^8$ | $2.5 * 10^6$ | — | 455.02 | 31.05 |
| soko-18** | $1.6 * 10^8$ | $8.3 * 10^7$ | — | — | 247.5 | — | — |
| soko-L | $3.7 * 10^8$ | — | $1.5 * 10^6$ | $4.0 * 10^5$ | — | 16.0 | 6.0 |
| reach-08** | $2.3 * 10^{18}$ | — | — | 60 | — | — | 26.05 |
| reach-14** | $6.2 * 10^{14}$ | — | — | $1.7 * 10^5$ | — | — | 3.36 |

Table 2. Experimental results of applying lazy model expansion. $**$ denotes ASP competition instances.

created groundings ($g(\text{IDP})$, $g(\text{IDP}_T)$ and $g(\text{IDP}_{T,S})$) is measured as the number of literals over the input vocabulary they contain. As the input structure is not taken into account in the measure, even the grounding of the IDP setup can be smaller than the full grounding. The running times ($t(\text{IDP})$, $t(\text{IDP}_T)$ and $t(\text{IDP}_{T,S})$) are also measured (in seconds). The results are shown in table 2.

The experiments show that, for a range of benchmarks and instances, lazy mx by incremental grounding can be very beneficial. One reason is that lazy mx acts as a kind of *dynamic dependency analysis*, selecting the parts of the theory which (hopefully) contribute to finding a model. The effect is extremely outspoken in reachability (reach-*), a benchmark generally solved by static dependency analysis. The dynamic character of our approach is both at least as powerful and more general. For some benchmarks, such as soko and bnq, the reduction in grounding size is counteracted by the adverse effect on propagation and search.

## 6 Related work

Within logic programming and ASP, research has been conducted towards reducing the size of the grounding by static dependency analysis, implemented in for example (Leone et al. 2006) and bottom up computation evaluation. Propagation techniques on the first-order level, delaying grounding until propagation ensues, has been researched within ASP (Lefèvre and Nicolas 2009), (Palù et al. 2009) and CP (Ohrimenko et al. 2009). Such techniques can be used in conjunction with lazy mx to reduce the size of the grounding considered and to lazily ground specific types of constraints not handled by the presented framework.

The model generation theorem prover Paradox (Claessen and Sörensson 2003) employs an incremental model generation algorithm. For each type in the vocabulary, an initial domain size is chosen (all domain elements are symmetrical) and the full grounding is constructed. If no model is found, the domain sizes are increased until a model is found or a bound on the size is hit (if one could be derived). This can be combined with lazy grounding to reduce the size of the domains considered.

The well-known technique *skolemisation*, used for example in theorem proving and Sat-Modulo-Theory algorithms, allows to deal directly with existential quantifiers by introducing function symbols. Reasoning on consistency can for example be achieved by congruence closure algorithms, capable of deriving consistency without effectively assigning an interpretation to the function symbols. Comparing with an approach using skolemisation and congruence closure is part of future work.

Another topic for future work is to investigate undoing grounding on backtracking, which would prevent the grounding becoming too large to handle.

## 7 Conclusion

Lazy model expansion is an approach to model expansion that interleaves solving and search. It can be highly beneficial when the original theory has a large (or infinite) grounding, because it tries to introduce just enough grounding to solve the problem. The disadvantages of lazy mx are that it provides less propagation than full grounding, and the order of grounding can effect search detrimentally. There remains much future work to improve lazy mx by incorporating ideas such as lifted unit propagation and devising better heuristics for controlling delay, but there are already examples where lazy mx is highly beneficial.

*Acknowledgements* 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. Broes De Cat is funded by the Institute for Science and Technology Flanders (IWT).

## References

Apt, K. R. 2003. *Principles of Constraint Programming.* Cambridge University Press.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge University Press.

Bogaerts, B., De Cat, B., De Pooter, S., and Denecker, M. 2012. The idp framework reference manual. `http://dtai.cs.kuleuven.be/krr/software/idp3/documentation`.

Claessen, K. and Sörensson, N. 2003. New techniques that improve MACE-style model finding. In *MODEL*.

Denecker, M. and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log. 9,* 2.

Denecker, M. and Vennekens, J. 2008. Building a knowledge base system for an integration of logic programming and classical logic. In *ICLP*. 71–76.

Lefèvre, C. and Nicolas, P. 2009. The first version of a new ASP solver : ASPeRiX. In *LPNMR*. 522–527.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic (TOCL) 7,* 3, 499–562.

Mitchell, D. G., Ternovska, E., Hach, F., and Mohebali, R. 2006. Model expansion as a framework for modelling and solving search problems. Tech. Rep. TR 2006-24, Simon Fraser University, Canada.

Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G., and Tack, G. 2007. Minizinc: Towards a standard CP modelling language. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, C. Bessiere, Ed. LNCS, vol. 4741. Springer-Verlag, 529–543.

Niemelä, I. 2006. Answer set programming: A declarative approach to solving search problems. In *JELIA*. 15–18. Invited talk.

Ohrimenko, O., Stuckey, P., and Codish, M. 2009. Propagation via lazy clause generation. *Constraints 14,* 3, 357–391.

Palù, A. D., Dovier, A., Pontelli, E., and Rossi, G. 2009. Answer set programming with constraints using lazy grounding. In *ICLP*, P. M. Hill and D. S. Warren, Eds. LNCS, vol. 5649. Springer, 115–129.

Van Gelder, A. 1993. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences 47,* 1, 185–221.

Wittocx, J. 2010. Finite domain and symbolic inference methods for extensions of first-order logic. Ph.D. thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium.

Wittocx, J., Mariën, M., and Denecker, M. 2008. The idp system: a model expansion system for an extension of classical logic. In *LaSh*, M. Denecker, Ed. 153–165.

Wittocx, J., Mariën, M., and Denecker, M. 2010. Grounding FO and FO(ID) with bounds. *Journal of Artificial Intelligence Research 38*, 223–269.