Lazy Model Expansion by Incremental Grounding

Broes De Cat[1], **Marc Denecker**[2], **and Peter Stuckey**[3]

1 **Department of Computer Science, K.U.Leuven, Belgium**
`broes.decat@cs.kuleuven.be`

2 **Department of Computer Science, K.U.Leuven, Belgium**
`marc.denecker@cs.kuleuven.be`

3 **National ICT Australia, Victoria Laboratory,**
Department of Computing and Information Systems,
University of Melbourne, Australia
`peter.stuckey@nicta.com.au`

Abstract

Ground-and-solve methods used in state-of-the-art Answer Set Programming and model expansion systems proceed by rewriting the problem specification into a ground format and afterwards applying search. A disadvantage of such approaches is that the rewriting step blows up the original specification for large input domains and is unfeasible in case of infinite domains. In this paper we describe a *lazy* approach to model expansion in the context of first-order logic that can cope with large and infinite problem domains. The method interleaves grounding and search, incrementally extending the current partial grounding only when necessary. It often allows to solve the original problem without creating the full grounding and is hence more widely applicable than ground-and-solve. We report on an existing implementation within the IDP system and on experiments that show the promise of the method.

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.4 Knowledge Representation Formalisms and Methods, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Knowledge representation and reasoning, model generation, grounding, IDP framework, first-order logic

Digital Object Identifier 10.4230/LIPIcs.ICLP.2012.201

1 Introduction

Model expansion [8] is the task of generating models of a logical theory for a given universe of domain elements. It 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. Model expansion is related to answer set generation in Answer Set Programming [10] and to finding variable assignments satisfying sets of constraints in Constraint Programming [1]. One approach used to solve such inference tasks is the *ground-and-solve* paradigm. The problem specification is formulated in a high-level (user-friendly) language, which is then rewritten into a lower-level representation on which a search algorithm can be applied. This process is called *grounding* (also known as *unrolling*). Examples are the high-level language FO($\cdot$) [5], which is grounded to its propositional fragment PC($\cdot$); ASP is grounded to propositional ASP and MiniZinc [9] is unrolled into Flatzinc.

An important bottleneck to applying ground-and-solve is the size of the grounding. Grounding an FO theory results in a blow-up of the size of the theory, related to the nesting depth of quantifiers and the size of the domains of the original theory. There are lots of practical problems in which the propositional theory is too large to generate.

In this paper, we present a novel approach to remedy this bottleneck, based on rewriting the theory *lazily* instead of up-front. The two main ideas are *placeholder introduction* and *$\forall$-instantiation*. Placeholders which represent non-ground formulas are introduced in the grounding. During search, they are "grounded further" depending on the interpretation. For sentences of (universally quantified) disjunctions, conditions are derived under which those sentences can certainly be satisfied by some extension of the current interpretation. As long as the interpretation satisfies those conditions, the sentence is not added to the grounding. Consider for example a disjunctive sentence: as long as one disjunct is true in the model, the value of the others is irrelevant.

It is also shown that the approach becomes even stronger when we are only interested in a subset of the full solution, as long as it can certainly be extended to a model. For example for planning problems, we are often only interested in the actions necessary to achieve the goal, independent of the full (possibly infinite) time frame and any additional relationships.

The approach is presented for theories in function-free first-order logic. Without loss of generality, any FO theory can be transformed into one not containing functions [18]. In section 3, it is shown how to represent partly ground theories and their properties. Section 4 shows which formulas to delay and the lazy model expansion algorithm is presented in section 4.4. Experimental results are provided in section 5, related work in 6.

2 Preliminaries

In this section, we first present syntax and semantics of FO, used throughout the paper, followed an introduction to the inference tasks *modelexpansion* and *grounding*.

2.1 FO

We assume familiarity with classical logic. A vocabulary Σ consists of a set of predicate and function symbols. Propositional symbols and constants are 0-ary predicate symbols, respectively function symbols. 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 ($\forall$, $\exists$).

Each variable x is assumed to have an associated set of domain elements t over which the variable ranges, denoted as $x[\![t]\!]$. We sometimes refer to such a variable as a *typed* variable and formulae containing only typed variables as typed formulae. Given a formula φ with a free variable x, substitution of x with domain element d is denoted as $\varphi[x/d]$.

Throughout the paper, A and L are used to refer to an atom, respectively a literal. A *ground sentence* is a sentence without variables (hence also without quantifiers). A *ground theory* is a theory consisting of ground sentences.

In this paper we deal with three-valued interpretations I which allow us to adequately represent the partial structure within a search algorithm. We will sometimes write an interpretation I as the set of all domain literals which are true in it. For example given a set of domain atoms $\{P, Q, R\}$, then $I = \{P, \neg Q\}$, denotes the interpretation in which P is true, Q is false and R is unknown.

The interpretation of a sentence under an interpretation I, denoted $I(\varphi)$, is defined as usual except for quantified formulae, as we assume each variable is typed. For existential quantification, $I(\exists x[\![t]\!](\varphi))$ is true iff there is a $d \in t$ such that $I(\varphi[x/d]) = \mathbf{t}$; and false iff for all $d \in t$ we have $I(\varphi[x/d]) = \mathbf{f}$. (Typed) universal quantification is defined similarly.

The interpretation of formulae containing the shorthands $\Rightarrow, \Leftarrow$ and $\equiv$ is taken to be the interpretation of the formulae they represent.

A (three-valued) interpretation I is a *model* of an FO sentence if and only if the sentence is true under the interpretation. It is a model of an FO 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' ($I' \subseteq I$). Two interpretations I and J *agree on shared symbols* if there is no proposition P where $\{P, \neg P\} \subseteq I \cup J$.

An occurrence of a subformula φ in $\mathcal{T}$ is called *monotone* if it is not in the scope of a negation. It is *anti-monotone* if it is in the scope of a negation. If it occurs as a subformula of an equivalence, it is called *non-monotone*. This reflects the well-known property that increasing the truth value of an atom with only monotone occurrences, increases the truth value of formulas. If the atom has only anti-monotone occurrences, then increasing its truth value decreases the value of formulae.

With a slight abuse of notation, given a theory $\mathcal{T}$, $\mathcal{T}$ is used both to refer to the set and the conjunction of the sentences it contains.

2.2 Model expansion

Model generation is the inference task of, given a vocabulary Σ, a theory $\mathcal{T}$ over Σ and a (partial) interpretation $\mathcal{S}_{in}$ of Σ, finding models M which satisfy $\mathcal{T}$ and are more precise than $\mathcal{S}_{in}$. If the universe of domain elements is part of the input structure, the inference task is called *model expansion*, denoted as MX$\langle \mathcal{T}, \mathcal{S}_{in} \rangle$. 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 extending the structure[8].

In this paper we consider an instance of model expansion where also an "output" vocabulary σ_{out} is given, a subset of Σ. The idea is then to generate interpretations I which are two-valued on σ_{out} and for which an extension exists which is a model of the theory $\mathcal{T}$ and is more precise than $\mathcal{S}_{in}$. Conceptually, this comes down to problems where we are only interested in some part of the solution, as long as we are guaranteed that a complete solution exists. This task generalizes both satisfiability checking (where σ_{out} is emtpy) and model expansion (where $\sigma_{out} = \Sigma$). It is denoted as MX$\langle \mathcal{T}, \mathcal{S}_{in}, \sigma_{out} \rangle$.

In the next sections, we present an approach for model expansion over an empty output vocabulary. In section 4.4, the approach is extended (in a straightforward way) to non-empty output vocabularies.

2.3 Grounding

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

More intelligent grounding techniques exist which reduce the size of the grounding, such as grounding with bounds [16].

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

Table 1 The definition of the full grounding $G_{full}(\psi)$ of an FO formula ψ not containing free variables.

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

remainder. Conditions on the partial interpretation are derived which govern whether additional grounding is necessary. We call such conditions *delays*.

▶ **Example 1.** Consider the sentence $(\exists x[\![t]\!] : P(t))$ with t ranging from 1 to n. As long as $P(1)$ is not false, it can still be assigned a value (true) such that the formula becomes satisfied. Therefore, we can delay the remaining instantiations by replacing them with a new Tseitin symbol T, resulting in the ground clause $P(1) \vee T$ and the non-ground "delayed" formula $T \equiv \exists x[\![t \backslash 1]\!]$. The latter formula is only grounded further if T becomes true.

Such a condition on the partial structure I, $I(T) \neq \mathbf{t}$, is called a *delay*. As long as it is satisfied, no additional grounding is performed on the associated sentence. The result is a theory which is equivalent to the original one (as Tseitin transformation was used) and if a partial interpretation can be found which satisfies the ground portion and the delay, it can certainly be extended to a full model (setting $I(T)$ to $I(\exists x[\![t \backslash 1]\!])$). ☐

▶ **Example 2** (Continued). Consider the theory consisting of the previous sentence and the additional sentence $(\forall x[\![t]\!] : P(x) \equiv \varphi(x))$, with φ a general formula not containing P. As long as $P(d)$ has not been assigned a value, $P(d)$ can still be assigned a value such that $P(d) \equiv \varphi[x/d]$ is consistent (namely $I(P(d)) = I(\varphi[x/d])$). Consequently, grounding only has to be applied for instantations of x with domain elements d for which $I(P(d))$ is not unknown. The delay is then the condition $I(\exists x[\![t]\!] : \neg P(d)) \neq \mathbf{u}$. ☐

Delayed grounding can lead to a significant reduction in the size of the grounding. In the case of $(\exists x[\![t]\!] : P(t))$, quantifier instantiation is only done partially. In the case of $(\forall x[\![t]\!] : P(x) \equiv \varphi(x))$, it is even completely avoided as long as the delay is satisfied.

3.1 Delays on formulae

A *delayed sentence* has the form $(\varphi)^\delta$ where φ is a sentence and δ is a delay condition (in short, a *delay*). A *delayed theory* $\mathcal{T}_d$ is a theory consisting of a ground theory $\mathcal{G}$ and a residual (non-ground) theory $\mathcal{D}$ consisting of *delayed sentences*. We will often denote it by $\langle \mathcal{G}, \mathcal{D} \rangle$. Such a delayed theory will be obtained by partially grounding the theory, resulting in $\mathcal{G}$, and partially delaying the grounding, resulting in $\mathcal{D}$.

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 $I(\varphi) = \mathbf{u}$.

We say that a (partial) interpretation I satisfies (is a model of) a delayed sentence $(\varphi)^\delta$ if I satisfies φ. We say that I *weakly* satisfies $(\varphi)^\delta$ if I satisfies the delay or I satisfies φ. By extension, an interpretation weakly satisfies a (delayed) theory iff it satisfies all its sentences.

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

The lazy grounding algorithm will iteratively reduce delayed theories into "more ground" delayed theories. The main invariant of the algorithm is that any such delayed theory is a *partial grounding* of $\mathcal{T}$: 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 interpretation that weakly satisfies $\mathcal{T}_d$ has a two-valued extension that satisfies $\mathcal{T}$.

▶ **Example 3** (Continued)**.** The delayed theory introduced in example 2 is a partial grounding of its original theory. It consists of a ground theory $P(1) \lor T$ and of the delayed sentences

$$(T \equiv \exists x[\![t \setminus 1]\!])^{T \neq \mathbf{t}} \qquad\qquad (\forall x[\![t]\!] : P(x) \equiv \varphi(x))^{\exists x[\![t]\!] : \neg P(d) = \mathbf{u}}$$

The next section shows which delays can be safely introduced to guarantee this invariant.

4 Introducing delayed sentences

The lazy grounding component of the lazy mx algorithm is responsible for the grounding of an active delayed theory into a more ground, inactive one.[1] To this end, delayed sentences are replaced by a combination of ground and delayed sentences. This is either achieved with Tseitin introduction (section 4.1) or $\forall$-instantiation (section 4.2). The lazy mx algorithm itself is then presented in section 4.4.

4.1 Tseitin introduction

Recall from example 1 that $\exists x[\![t]\!] : P(x)$ was partially grounded to the ground formula $P(1) \lor T$ and a delayed sentence $((T \equiv (\exists x[\![t \setminus 1]\!] : P(x)))^{T \neq \mathbf{t}}$. We here describe this operation.

▶ **Definition 4** (Tseitin introduction)**.** Given a delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$ and a set of occurrences of a formula φ in sentences ψ with $(\psi)^{\delta} \in \mathcal{T}_d$, the Tseitin introduction for φ in $\mathcal{T}_d$ is the delayed theory $\mathcal{T}_d$'=$\langle \mathcal{G}, \mathcal{D}' \rangle$ where $\mathcal{D}$' is obtained from $\mathcal{D}$ by

- substituting each selected occurrence of φ in $\mathcal{D}$ with the new propositional symbol T_{φ}
- adding a new delayed sentence $(T_{\varphi} \equiv \varphi)^{\delta'}$ where δ' is determined as follows:
  - If all selected occurrences of φ 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, $\delta' = (T_{\varphi} = \mathbf{u})$. ☐

Applying Tseitin introduction to any partial grounding of a theory $\mathcal{T}$ results in a partial grounding of $\mathcal{T}$.

▶ **Example 5.** Consider the theory $\mathcal{T} = P \equiv \forall x[\![t]\!] : Q(x)$. Applying Tseitin introduction to $\forall x[\![t \setminus 1]\!] : Q(x)$ results in the ground theory $P \equiv Q(1) \land T$ and the delayed sentence $(T \equiv \forall x[\![t \setminus 1]\!] : Q(x))^{T = \mathbf{u}}$. This delayed theory is a partial grounding of $\mathcal{T}$.

[1] 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 an empty ground theory and all its formulae active.

4.2 $\forall$-instantiation

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 6.** Consider the definite clause $\forall \overline{x}[\![\overline{t}]\!] : P_1(\overline{x}) \land \ldots \land P_n(\overline{x}) \Rightarrow Q(\overline{x})$. 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 extension in which all heads are true. Consequently, only instantiations with domain elements $\overline{d}$ for which $I(Q(\overline{d}))$ is false need to be grounded. The remaining instantiations can then be delayed on the falsity of their heads. Assume for example that only $I(Q(\overline{d}_1)) = \mathbf{f}$ for some $\overline{d}_1 \in \overline{t}$. The delayed theory $\mathcal{T}_d$

$$\mathcal{T}_d = \left\langle \phi[\overline{x}/\overline{d}_1] \Rightarrow Q(\overline{d}_1),\ \{(\forall \overline{x}[\![\overline{t} \backslash \overline{d}_1]\!] : \phi(\overline{x}) \Rightarrow Q(\overline{x}))^{\exists \overline{x}[\![\overline{t}\backslash \overline{d}_1]\!] : \neg Q(\overline{x})\ \neq\ \mathbf{t}}\} \right\rangle$$

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

Below, it is shown formally how to delay instantiation for universally quantified disjunctive sentences based on their satisfiability. Delaying those is not captured by Tseitin introduction and they represent a class of formulae which occur often in practice. At the end of the section, it is shown how the approach can be extended to other classes of formulae.

▶ **Definition 7** ($\forall$-instantiation)**.** Consider a delayed theory $\mathcal{T}_d = \langle \mathcal{G}, \mathcal{D} \rangle$, a partial grounding of $\mathcal{T}$, and an interpretation I. Assume a sentence $\psi = \forall \overline{x}[\![\overline{t}]\!] : \bigvee_{i \in [1,m]} \varphi_i$ with $(\psi)^\delta \in \mathcal{D}$.[2] Applying $\forall$-instantiation to ψ 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 of which the symbol does not occur with opposite sign in any delay in $\mathcal{T}_d$. Assume nd denotes the set of tuples of domain elements which falsify all formulae in S_d under I (so they cannot be delayed). Then $\langle \mathcal{G}_{rem}, \mathcal{D}_{rem} \rangle$ is the grounding of the sentence $\forall \overline{x}[\![nd]\!] \bigvee_{i \in [1,m]} \varphi_i$. The remaining instantiations are delayed by the delay condition $\chi = \exists \overline{x}[\![\overline{t} \backslash nd]\!] : \bigwedge_{\varphi_i \in S_d} \neg \varphi_i \neq \mathbf{t}$. The result is the delayed theory $\mathcal{T}'_d = \left\langle \mathcal{G} \land \mathcal{G}_{rem}, \mathcal{D} - \{\psi\} \cup \mathcal{D}_{rem} \cup (\forall \overline{x}[\![\overline{t} \backslash nd]\!] \bigvee_{i \in [1,m]} \varphi_i)^\chi \right\rangle$. □

As for Tseitin introduction, it can be shown that applying $\forall$-instantiation to any partial grounding of a theory $\mathcal{T}$ results in a partial grounding of $\mathcal{T}$.

It should be noted that whether such a delay can contain a literal over a symbol P depends on occurrences of P in existing delays. If multiple delays are watching different truth assignments to the same symbol, inconsistencies might not be detected.

$\forall$-instantiation can be extended to other classes of formulae such as equivalences and non-monotone occurrences of quantifiers. For sentences of the form $\forall \overline{x}[\![\overline{t}]\!] : L(\overline{x}) \equiv \varphi(\overline{x})$ for example, the strategy is as outlined above except that χ becomes a known-delay. In the same fashion, the approach can be extended to *inductive definitions*[5], sets of rules of the form $(\forall \overline{x} : L(\overline{x}) \leftarrow \phi)$ evaluated by the well-founded semantics[13]. Furthermore, the exact delays used by $\forall$-instantiation allow us to trade-off propagation versus grounding size and towards solving query tasks. Details are out of the scope of this paper, but results of these ideas are included in the prototype implementation used in the experiments.

4.3 Delayed grounding algorithm

With these techniques, we can now give an informal (due to lack of space) presentation of the delayed grounding algorithm for_del_gnd. The algorithm takes as arguments an

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

interpretation I, a theory $\mathcal{T}$ and the set of delays of all currently delayed sentences. It returns a delayed theory $\mathcal{T}_d$ inactive in I which is a partial grounding of $\mathcal{T}$.

We assume a standard top-down reduced grounding algorithm, such as in e.g. [15], which recursively visits the theory top-down to ground it. The algorithm keeps track of the set of ground and delayed sentences and of the context ((anti/non)-monotone).

Lazy grounding is applied in two different scenarios. Firstly, if a universally quantified disjunctive sentence is encountered for which a set of subformulas can be selected according to the above conditions, the sentence is delayed using $\forall$-instantiation (recursively grounding non-delayable instantiations). Secondly, for an existential quantification or disjunction, a non-false subformula is selected randomly[3] and grounded recursively; the remainder of the formula is delayed by Tseitin introduction. The second approach is also applied for non-monotone occurrences of universal quantifiers and conjunctions.

4.3.1 Incremental delayed grounding algorithm

Initially, for_del_gnd is applied to I and $\mathcal{T}$ (no delayed sentences yet), resulting in an initial delayed theory $\mathcal{T}_d$. 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 is applied to the associated delayed sentences. This is achieved by iterating over all delayed sentences in $\mathcal{T}_d$ and *incrementally* applying for_del_gnd to each active delayed sentence. Each new ground sentence is added to the ground theory and the original delayed sentence is replaced by new delayed sentences (if any). This algorithm is denoted as inc_del_gnd. It takes as input an interpretation I and a delayed theory $\mathcal{T}$ and the result is a partial grounding of $\mathcal{T}$ which is inactive under I.

▶ **Example 8.** Consider $T = \exists x[\![t]\!] : (P(x) \land R(x)) \lor (\forall y[\![t']\!] : Q(x, y))$. Delayed grounding of this sentence is achieved by selecting a domain element $d \in t$ and applying for_del_gnd to $(P(d) \land R(d)) \lor (\forall y[\![t']\!] : Q(d, y))$ while applying Tseitin introduction to the residual subformula $\phi = \exists x[\![t \backslash d]\!] : (P(x) \land R(x)) \lor (\forall y[\![t']\!] : Q(x, y))$.

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

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

□

4.4 Lazy model expansion

The complete lazy model expansion algorithm, denoted lazy_mx and shown below, interleaves grounding and search based on a standard (incremental) CDCL search algorithm. The algorithm (shown below) gets as input a theory $\mathcal{T}$ and a pre-interpretation S_{in} and maintains the current delayed theory $\langle \mathcal{G}, \mathcal{D} \rangle$. The (current) ground theory provides the constraints used during search. If a conflict at root level is encountered, 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, grounding is performed to construct a new delayed theory which is a partial grounding of $\mathcal{T}$. If all delays are inactive and the search algorithm detects that I satisfies all constraints in $\mathcal{G}$, I is a weak model of $\langle \mathcal{G}, \mathcal{D} \rangle$. As $\langle \mathcal{G}, \mathcal{D} \rangle$ is a partial grounding of $\mathcal{T}$, $\mathcal{T}$ is satisfiable.

[3] Better heuristics are part of future work.

```
lazy_mx ($\mathcal{T}$, $I$)
    $\langle\mathcal{G},\mathcal{D}\rangle$ := for_del_gnd($\mathcal{T}$, $I$, $\emptyset$)
    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
```

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

Deciding atoms occurring in known-delays when I is already a weak model of $\mathcal{T}$ will obviously cause unnecessary grounding. As standard search algorithms decide all literals in the ground theory, we use a modified algorithm which tracks satisfaction of constraints in an efficient way without deciding all literals in the ground theory.

To handle non-empty output vocabularies σ_{out}, we modified the algorithm to always return models which are two-valued on σ_{out}. This is achieved by forcing the search algorithm to decide all atoms in the set of domain atoms of σ_{out} under $\mathcal{S}_{in}$.

5 Experiments

A prototype implementation was created within the IDP-3 system, a knowledge base system based on extensions of first-order logic. The IDP system is a state-of-the-art model expansion system, based on the ground-and-solve paradigm [14], [3].

Experiments were conducted with three setups: basic model expansion (denoted IDP), lazy model expansion by Tseitin introduction (IDP$_T$) and by Tseitin introduction and by $\forall$-instantiation (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 existing benchmarks are problems with a feasible grounding and difficult search part, we also created new instances which are computationally easier but have a very large grounding. This combination will allow to assess the strengths and weaknesses of the approach.

For each benchmark instance, runtime and grounding size are measured for each setup. Grounding size is measured as the number of literals over the input vocabulary. The grounding size of all setups is compared to the (theoretical) grounding size of the full grounding (see table 1). [4] 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. For most benchmarks, the grounding size is reduced orders of magnitude over both the full grounding and the reduced grounding as

[4] For the full grounding, the input structure is not taken into account. Consequently, even the grounding of the IDP setup can be smaller than the full grounding as IDP constructs a reduced grounding.

Table 2 Experimental results of applying lazy model expansion (IDP$_T$ and IDP$_{T,S}$) compared to default model expansion (IDP). Grounding time (in seconds) is denoted as t, grounding size as G, $**$ denotes ASP competition instances. A timeout of 1000 seconds was used and a memory limit of 3 Gb, — indicates timeout or memory overflow. All experiments were run on an Intel Core 2 Machine (dual 2.40Ghz) running Ubuntu 10.4. The version of the IDP system used in the experiments and all data files are available from `http://dtai.cs.kuleuven.be/krr/research/experiments`.

Benchmark	G_{full}	G_{IDP}	G_{IDP_T}	$G_{\text{IDP}_{T,S}}$	t_{IDP}	t_{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	∞	—	—	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	∞	—	—	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

done by IDP. Running times on many benchmarks go from untractable to solvable within seconds. Runtime is only worse for the sokoban 18 problem.

Tseitin introduction proves to be an advantage in benchmarks such as encoding functional dependencies and planning: problems which are generally solved by selecting an appropriate (small) subset of literals to assign, even if this choice is unguided. Indeed, it is enough to select one domain elelement for each function range or only actions for a small timeframe in many planning problems. On the other hand sokoban 18 shows that in hard planning problems, the incremental approach has an adverse effect on the search (introduction of large number of Tseitin literals). As expected, it has few effects on problems with a universal quantifications over large domains, such as packing 2, reachability and func 2.

Delaying using ∀-instantiation was expected to have a positive effect on most benchmarks, unless the loss in propagation is too significant (such as for sokoban 18). On all other benchmarks, it has an outspoken positive effect:

- Even for bounded N-Queens, a hard search problem, the grounding size is reduced and the performance increased, because non-propagating implications are not grounded.
- Problems with an infinite full grounding can be solved, such as planning problems over infinite times. The conjunction with Tseitin introduction is crucial to delay both existential and universal quantifiers.
- It acts as a kind of *dynamic dependency analysis*, selecting the parts of the theory which (hopefully) contribute to finding a model. This can be observed in particular in reachability (reach-*), in fact a query task generally solved by (static) dependency analysis. Additionally, the dynamic character of our approach is both at least as powerful and more general, during search only grounding what becomes relevant.

6 Related work

Within logic programming and ASP, static dependency analysis is applied as a means to reduce the size of the grounding up-front by detecting non-relevant parts of the theory. It has been implemented for example in [7] and [17]. Furthermore, lazy grounding techniques have been researched within ASP [6], [12] and CP [11]. Such techniques usually work on delaying grounding of specific constraints as long as they do not cause propagation, for example for all-different constraints, aggregates and equality reasoning. Our lazy mx approach is more general, as it performs dependency analysis dynamically and is able to delay grounding even when propagation is possible, but might be less powerful for constraints for which specific algorithms exist. Comparing those techniques in-depth is part of future work.

The model generation theorem prover Paradox [4] uses a grounding technique based on lazily extending the domain of the quantifiers. It first chooses a domain (all domain elements are symmetrical) and constructs the full grounding. If no model is found, it increases the domain size, until a model is found or a bound on the size is hit (if one could be derived). Such an approach stands orthogonal to the work presented in this paper and it is part of future work to combine the advantages of both approaches.

As part of future work, we will investigate the relation with techniques used to delay the grounding of quantifiers such as *skolemisation*, used e.g. in theorem proving, and congruence closure algorithms which reason on equality of terms, from the domain of SAT-Modulo-Theories. Another promising topic is that of undoing grounding on backtracking. In effect, it might be possible to track whether delays have become inactive again and to remove the associated constraints again. Such a strategy would allow to reduce the size of the grounding again when a different part of the search space comes under investigation.

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

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

Research supported by Research Foundation-Flanders (FWO-Vlaanderen).

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

References

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

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

3 Bart Bogaerts, Broes De Cat, Stef De Pooter, and Marc Denecker. The IDP framework reference manual. `http://dtai.cs.kuleuven.be/krr/software/idp3/documentation`, 2012.

4 Koen Claessen and Niklas Sörensson. New techniques that improve MACE-style model finding. In *MODEL*, 2003.

5 Marc Denecker and Eugenia Ternovska. A logic of nonmonotone inductive definitions. volume 9, 2008.

6 Claire Lefèvre and Pascal Nicolas. The first version of a new ASP solver : ASPeRiX. In *LPNMR*, pages 522–527, 2009.

7 Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV system for knowledge representation and reasoning. volume 7, pages 499–562, 2006.

8 David G. Mitchell and Eugenia Ternovska. A framework for representing and solving NP search problems. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 430–435. AAAI Press / The MIT Press, 2005.

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

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

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

12 Alessandro Dal Palù, Agostino Dovier, Enrico Pontelli, and Gianfranco Rossi. Answer set programming with constraints using lazy grounding. In Patricia M. Hill and David Scott Warren, editors, *ICLP*, volume 5649 of *LNCS*, pages 115–129. Springer, 2009.

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

14 Johan Wittocx, Maarten Mariën, and Marc Denecker. The IDP system: a model expansion system for an extension of classical logic. In Marc Denecker, editor, *LaSh*, pages 153–165, 2008.

15 Johan Wittocx. *Finite Domain and Symbolic Inference Methods for Extensions of First-Order Logic.* PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, May 2010.

16 Johan Wittocx, Maarten Mariën, and Marc Denecker. Grounding FO and FO(ID) with bounds. *Journal of Artificial Intelligence Research*, 38:223–269, 2010.

17 M. Gebser and R. Kaminski and A. König and T. Schaub. Advances in *gringo* Series 3. In James P. Delgrande and Wolfgang Faber, editors, *LPNMR*, volume 6645 of *LNCS*, pages 345-351. Springer, 2011.

18 Heinz-Dieter Ebbinghaus and Jörg Flum and Wolfgang Thomas. Mathematical logic (2. ed.). Springer, 1994.