Model revision inference for extensions of first order logic

Joachim Jansen

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

Abstract

I am Joachim Jansen and this is my research summary, part of my application to the Doctoral Consortium at ICLP'14. I am a PhD student in the Knowledge Representation and Reasoning (KRR) research group, a subgroup of the Declarative Languages and Airtificial Intelligence (DTAI) group at the department of Computer Science at KU Leuven. I started my PhD in September 2012. My promotor is prof. dr. ir. Gerda Janssens and my co-promotor is prof. dr. Marc Denecker.

I can be contacted at `joachim.jansen@cs.kuleuven.be` or at:

Room 01.167
Celestijnenlaan 200A
3001 Heverlee
Belgium

An extended abstract / full version of a paper accepted to be presented at the Doctoral Consortium of the 30th International Conference on Logic Programming (ICLP 2014), July 19-22, Vienna, Austria

1 Problem description

1.1 Introduction

The IDP system is a state-of-the-art system for declarative problem solving; complex search- and optimizationproblems are solved in an efficient and generic manner. As time passes on however, the found solution has to be *revised*: new information (e.g., changed circumstances) has to be taken into account. In this case it is desirable to start from the old (near-)solution and by perfroming a limited amount of changes transform it into a solution in which the new information is processed. At the moment there are no efficient, general solutions for these kind of problems; the only way this problem is currently solved is by writing special-purpose algorithms. During my thesis I would like to devise a general way to solve these problems using IDP as a system supporting an expressive modeling language.

The *Knowledge Base System* (KBS) (De Pooter et al. 2011) paradigm is a declarative approach in which one specifies *what* needs to be solved, instead of writing procedures that depict *how* to do this (Apt 2003; Gebser et al. 2012). A KBS represents the knowledge in its explicit form using an expressive modeling language and provides inferences to solve different kinds of problems. The expressive modeling language has as advantage that domains with a very complex or quickly changing knowledge can be expressed in a concise and clear way. Additionally, knowledge can be reused to solve different problems sharing the same scope. Because the inferences are domain independent, they can be reused across different scopes as well.

One of these inferences is model revision; the adaptation of an existing solution to new

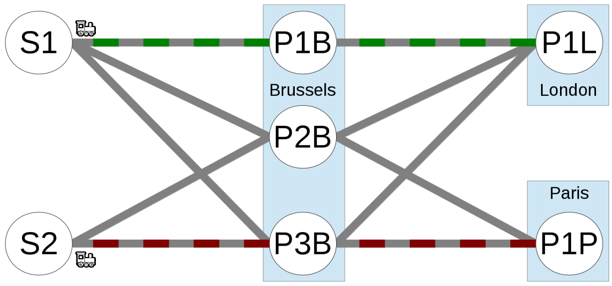


Figure 1. An example of a train routing situation

information. In a train dispatching toy-problem for example there are a plethora of unforeseen circumstances (e.g., track defects, delays, copper cable thefts) and the dispatching schedule needs to be adapted to new requirements. Model revision also tries to maintain as much as possible of the original solution (dispatching schedule) when processing the change. This is a consequence of the solution technique that is generally efficient (start from the 'old' solution and apply a limited amount of changes), but is also a desirable property of the computed solution. Indeed, when a train is delayed in Paris, it doesn't make a lot of sense to change the dispatching schedule of trains in London when this is not necessary.

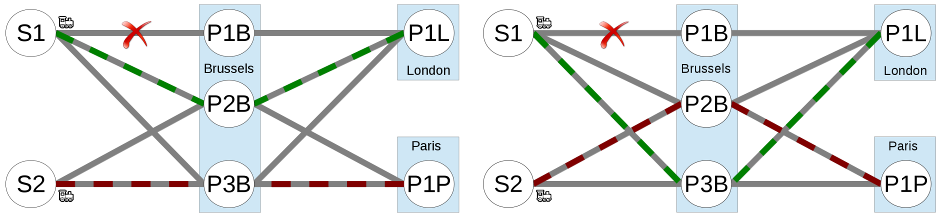
1.2 Model revision: a motivating example

Here we introduce a small motivating example of the model revision inference using the situation depicted in Figure 1.2. In this figure the train tracks are indicated using grey lines between nodes. The train starting in S1 (which we will also call Train1) has to go past stations Brussels and London and the train that starts in S2 ((which we will call Train2) has to visit stations Brussels and Paris. The dispatched route for this is indicated using a green dotted line, that of Train2 is indicated with a red dotted line. Imagine the train track between S1 (Shunting 1, shuntings are intermediary crossroads in train tracks where one can change direction) and P1B (Platform 1 in Brussels) is detected to have broken down. By using model revision we can construct a new route for Train2 in S1 that does not use any broken down train tracks. Figure 2(a) shows a high-quality revised model: a route has been found for Train1 without changing too much to the existing dispatching. Figure 2(b) shows a low-quality revised model: the route for Train1 is correct but an unnecessary change to the route of Train2 was made. The change to the route of Train2 was needed because there is a requirement that states that two trains cannot enter the same station on the same platform (at the same time).

1.3 Formal definition of model revision

The formal definition for the model revision problem is as follows (Wittocx et al. 2009):

Given a FO($\cdot$) theory T, a model M for this theory and a collection of domain atoms C. Henceforth C are called the *required changes*. In the example C is the usage of the tracks between S1 and P1B. Solving *model revision* for $\langle T, M, C\rangle$ means searching a new model M' of T such that all domain atoms in C all have a different value compaired to their old one in M. M' is also called the *revised model*. Figure 2 shows two possible revised models for the example; the broken down train tracks are not used in either case.



(a) A high-quality revised model (b) A low-quality revised model

Figure 2. Examples of high- and low-quality revised models

In addition to the required changes, one usually has to change other parts of the original solution as well to construct the revised model. We call these other changes between M and M' the *additional changes* and denote them with S. In the example the usage of the new route is the additional change.

Often it is not desirable that the entire original model M is be changed; some elements are *immutable*. In the example the structure of the train tracks is considered immutable: we are not interested in new solutions that would require us to build additional train tracks (e.g., one between London and Paris). These immutable elements in the problem domain are represented by the *limitation* G, a set of domain atoms whose value must remain fixed. The revision problem for $\langle T, M, C, G\rangle$ is the same as the revision problem for $\langle T, M, C\rangle$, except for the extra condition that the additional changes cannot include any of the limitations (i.e., S is disjoint with G).

2 Existing Literature

Model revision allows us to flexibly use with a computed solution by imposing new restrictions. Although this kind of flexible reasoning is essential to a KBS, there is no research for model revision (in its general sense) in the context of an expressive modeling language. Comparable research has been performed in areas of incremental constraint programming (Freeman-Benson et al. 1990) and reactive answer set programming (Gebser et al. 2011). In this research only a limited form of new requirements are supported: one takes into account specific forms of new types of knowledge, but e.g. there is no way to apply previously onforeseen changes. Recent research are also trying to tackle this problem on the SAT level (Abo et al. 2011). These SAT-level techniques are interesting for the implementation that will be provided eventually because IDP uses a SAT solver in its workflow, but do not work in the context of a complex modeling language. There has also been work on trying to construct the solution in such a way that it is 'robust' w.r.t. changes (Climent et al. 2014). For first order logic there is a basic algorithm that takes general changes into account (Wittocx et al. 2009). This will serve as a starting point for my thesis.

3 Background

This section contains a short introduction to the used terminology. The following concepts are introduced briefly: Knowledge Base System paradigm (De Pooter et al. 2011), FO($\cdot$) (Blockeel et al. 2013), and the IDP system (De Cat et al. 2013; IDP 2013).

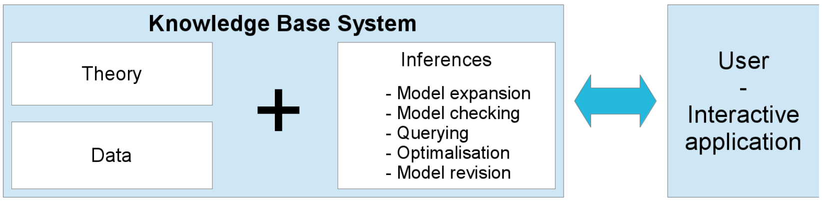


Figure 3. Een conceptuele voorstelling van een Knowledge Base System

3.1 The language: FO($\cdot$)

Each declarative system requires a language in which the problems are represented. This language is preferably expressive, so the problem domain can be intuitively expressed. The FO($\cdot$) family of languages has been developed at the KRR group for this purpose.

FO($\cdot$) is a family of expressive knowledge representation languages that extend classical First Order Logic (FO) with various concepts. Apart from the logical symbols ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$, $\exists$, $\forall$), FO($\cdot$) also contains:

Inductive definitions are represented as a set of defining rules.

Set expressions of the form $\{x\ y : p(x) \wedge q(y) \wedge r(x, y)\}$ represent the set of all combinations of x and y such that $p(x) \wedge q(y) \wedge r(x, y)$.

Aggregates express the result of an aggregate function of a set expression together with a cost function (for each element in the set). The following aggregate functions are supported: minimum, maximum, sum, product and cardinality.

Expressive quantifiers such as $\exists_{=1}$ (there exists exactly one), $\exists_{\geq 2}$ (there exists at least two) ...

Types and subtypes : each variable is typed in FO($\cdot$).

(Partial) functions . These are non-Herbrand functions.

Arithmetic operators such as $+$ $-$, $\times$, $\div$, $|x|$, and $\%$.

A problem specification in FO($\cdot$) consists of at least three parts: a *vocabulary* that depicts the domain ontology, a *theory* containing the constraints for this problem, and a *structure* that contains the known data about the problem.

For a more hands-on introduction to FO($\cdot$) and IDP, the reader is directed to our webpage of examples at `http://dtai.cs.kuleuven.be/krr/software/idp-examples`

3.2 Knowledge Base System

In a Knowledge Base System (KBS) the data and knowledge (expressed in the modeling language, e.g., FO($\cdot$)) are maintained in a *Knowledge Base*. A KBS then offers a variety of inferences to solve problems with the knowledge. A conceptual representation of a KBS is displayed in Figure 3.

Among these inferences are *model expansion* (extend a three-valued structure such that it satisfies a theory), *model checking* (verify whether a given structure satisfies a theory), *optimization* (extend a three-valued structure to a two-valued structure that satisfies a theory that has the least cost), and *model revision* (see Section 1.3).

3.3 The IDP *system*

The IDP system is a state-of-the-art implementation of the KBS paradigm using $\mathrm{FO}(\cdot)^{\mathtt{IDP}}$ as its modeling language. The workflow of the IDP system is as follows (De Cat et al. 2013). First the $\mathrm{FO}(\cdot)$ theory is ground into a low-level propositional representation. This representation is called "Extended CNF" or ECNF. It is an extension of CNF with concepts such as inductive definitions (that are ground). Next IDP uses a SAT-solver, MINISAT(ID), to generate solutions based on the grounding. IDP as well as MINISAT(ID) are open-source and available at `https://bitbucket.org/krr/idp` and respectively `https://bitbucket.org/krr/minisatid`.

The goal of my thesis is to provide support for model revision in the IDP system.

4 Goal of the research

The goal of my PhD thesis is to develop logic inference methods for different forms of model revision in the context of the $\mathrm{FO}(\cdot)$ modeling language.

In order for this to be possible, we need a mechanism to reason about changes propagating through a theory. To this end, the *approximating definition* for a theory (Wittocx 2010; Vlaeminck 2012; Vaezipoor et al. 2011) needs to be computed and used to propagate impact of a change to the solution throughout the theory of the problem. The theory behind this currently supports basic FO. For my thesis, I will extend the scope of approximating definition to theories containing more expressive constructs such as inductive definitions, aggregates...

Because the approximating definition is a definition that needs to be calculated, there need to be efficient techniques for doing so. It was proposed in (Wittocx 2010) that the definition can be evaluated using any external system that can evaluate definitions (or rules).

For model revision there are typically a multitude of possible revisions. There is a need for proper *criteria* that quantify the quality of a revision. I intend to construct criteria using a *domain independent* as well as a *domain dependent* approach. For the domain independent criteria some brute-force metrics such the number of changed domain atoms will be used. In order to properly support domain dependent criteria, a user needs to able to express which revisions are preferred over others. This can be done either by expressing them beforehand using some sort of cost function. For this the knowledge representation language needs to be extended. Another way to do this is to let the user interactively guide the search process for the revision, indicating which choices are preferred.

5 Current status of the research

The first part of my PhD consisted of constructing an interface between XSB and IDP for calculating definitions that can be completely evaluated. For this work, the inductive definitions are transformed into rules for tabled Prolog. This was published in TPLP (Jansen et al. 2013).

Further I extended IDP to compute the approximating definition using the existing theory concerning this topic. Additionally, IDP was also extended with the possibility to making the input structure as two-valued as possible before grounding using the approximating definition (De Cat et al. 2013) as an alternative approach to the "Ground With Bounds" (GWB) technique depicted in (Wittocx 2010; Vlaeminck 2012). Currently benchmarks are being run to compare the two approaches. According to (Vaezipoor et al. 2011) the new approach using approximating definition outperforms the classical GWB technique because it will always compute all possible unit propagation possible (at SAT-level) beforehand. GWB on the other hand sometimes performs cutoffs to increase performance. Preliminary results however contratict this claim.

Another claim from (Vaezipoor et al. 2011) is currently being investigated: a "smarter" grounding will affect the search tree as well. A smarter grounding can contain fewer introduced symbols (i.e., Tseitins) because it was detected beforehand that they need not be generated at all. Since these Tseitins are not removed by performing unit propagation at SAT-level, a smarter grounding thus contains (according to the above authors) possbily fewer "autarkies" - irrelevant parts of the search space in which the solver possbibly can waste time. Currently experiments are being run that compare the search behaviour of solver runs on smart, respectively "naive" groundings.

6 Preliminary results

Benchmarks over problems in the P complexity class that are generally solved by evaluating definitions for completely given structures show that a great speedup is achieved compared to the classical approach (Jansen et al. 2013).

Preliminary results (a complete study is being performed) suggest that making the input structure as two-valued as possible before grounding using approximating definitions is not superior to its counterpart the classic GWB workflow already implemented in IDP. Additionally, there were only very few problems where the grounding was smaller.

7 Open Issues

Tasks that still need addressing are the extension of the approximating definition for theories that contain more expressive constructs such as inductive definitions, aggregates... Additionally, the solver MINISAT(ID) will need to be adapted to support model revision. For support of interactively searching for a revision, the solver workflow also needs to be updated to work interactively with user input.

References

ABO, I., DETERS, M., NIEUWENHUIS, R., AND STUCKEY, P. 2011. Reducing chaos in sat-like search: Finding solutions close to a given one. In *Theory and Applications of Satisfiability Testing - SAT 2011*, K. Sakallah and L. Simon, Eds. Lecture Notes in Computer Science, vol. 6695. Springer Berlin Heidelberg, 273–286.

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

BLOCKEEL, H., BRUYNOOGHE, M., BART, B., DE CAT, B., DE POOTER, S., DENECKER, M., LABARRE, A., RAMON, J., AND VERWER, S. 2013. Predicate logic as a modeling language: Modeling and solving some machine learning and data mining problems with idp3. *CoRR abs/1309.6883*.

CLIMENT, L., WALLACE, R. J., SALIDO, M. A., AND BARBER, F. 2014. Robustness and stability in constraint programming under dynamism and uncertainty. *J. Artif. Intell. Res. (JAIR) 49*, 49–78.

DE CAT, B., JANSEN, J., AND JANSSENS, G. 2013. IDP3: Combining symbolic and ground reasoning for model generation. In *Workshop on Grounding and Transformations for Theories with Variables, La Coruña, 15 Sept 2013*.

DE POOTER, S., WITTOCX, J., AND DENECKER, M. 2011. A prototype of a knowledge-based programming environment. *CoRR abs/1108.5667*.

FREEMAN-BENSON, B. N., MALONEY, J., AND BORNING, A. 1990. An incremental constraint solver. *Commun. ACM 33,* 1, 54–63.

GEBSER, M., GROTE, T., KAMINSKI, R., AND SCHAUB, T. 2011. Reactive answer set programming. In *Logic Programming and Nonmonotonic Reasoning*, J. Delgrande and W. Faber, Eds. Lecture Notes in Computer Science, vol. 6645. Springer Berlin Heidelberg, 54–66.

GEBSER, M., KAMINSKI, R., KAUFMANN, B., AND SCHAUB, T. 2012. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

IDP 2013. The IDP system. `http://dtai.cs.kuleuven.be/krr/software`.

JANSEN, J., JORISSEN, A., AND JANSSENS, G. 2013. Compiling input$*$ FO($\cdot$) inductive definitions into tabled Prolog rules for IDP^3. *Theory and Practice of Logic Programming (TPLP) 13,* 4-5, 691–704.

VAEZIPOOR, P., MITCHELL, D., AND MARIËN, M. 2011. Lifted unit propagation for effective grounding. *CoRR abs/1109.1317*.

VLAEMINCK, H. 2012. Applications of feasible inference for expressive logics. Ph.D. thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium. Approximating Definitions are introduced in chapter 3.

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., DE CAT, B., AND DENECKER, M. 2009. Towards computing revised models for FO theories. In *INAP*, S. Abreu and D. Seipel, Eds. 199–212.