nlp

A compiler for nested logic programs

Vladimir Sarsakov Torsten Schaub Hans Tompits
Stefan Woltran

14th August 2003

1 Introduction

These pages describe a system for compiling nested logic programs into disjunctive logic programs under answer set semantics [2]. Our system is conceived as a front-end to the logic programming systems `dlv` and (since recently) `gnt`.

The underlying compiler is implemented in the programming language Prolog; it has been developed under the Logic Programming Systems SICStus and SWI; its code employs standard Prolog programming constructs so that it remains portable to other Prolog systems.

We deal with nested logic programs under the answer set semantics [3] which is an extension of the stable models semantics [1] for handling logic programs with classical negation.

An excellent introduction to logic programming under these semantics is due to Vladimir Lifschitz and can be accessed through his home-page.

This emerging subfield common to logic programming and nonmonotonic reasoning is also referred to *Answer Set Programming.* Some links on the subject are given here.

2 Getting started

The best way of getting started is to consult an exemplary session under SICStus Prolog. In these session we take the example and proceeded in the following way:

1. We start by loading the compiler (here into SICStus)

   ```
   [nlp41].
   ```

   (Or alternatively `[nlp50].`)

2. We compile an original file t1.nlp by means of

   ```
   nlp2htl('Examples/t1').
   ```

   This results in the file t1.htl which is not readily usable by `dlv`.

   Our compiler is pretty verbose and displays also intermediate versions of the compiled program (use the flag `verbose_mode/no_verbose_mode` for switching).

3. Alternatively, one may combine the two latter steps by appeal to the command

   ```
   nlp2htl2dlv('Examples/t1').
   ```

   In all, this call produces 3 files:

   - t1.dic
     A dictonary, provided that the appropriate flags (cf.
     The "logically" resulting file of the transformation.
   - t1.dlv
     The `dlv`-specific file obtained from the "logically" resulting file of the transformation.

4. We finally call `dlv` by issuing the commands:

   ```
   dlv('Examples/t1').
   ```

3 Syntax

- `true/0` is a predefined fact (that is used for `dlv` in order to make facts go into the intentional database);
- `false/0` is a predefined symbol never to be found in any answer set (used for defining classical negation via integrity constraints);
- `not/1` are prefix predicates standing for negation as failure;
- `,/2` is conjunction;
- `;/2` is disjunction;
- `:-/2` is used for describing rules in the usual way.

4 Command predicates

General translations:

- `nlp2dlp/1`

  Takes a Filename `<filename>` and compiles file `<filename>.nlp` into file `<filename>.dlp` according to the standard translation of nested logic programs into disjunctive logic programs.

  This may result in an exponential blow-up in the worst-case.

- `nlp2htl/1`

  Takes a Filename `<filename>` and compiles file `<filename>.nlp` into file `<filename>.htl` according to the structural translation of nested logic programs into disjunctive logic programs.

  This translation is guarenteed to result in a polynomial blow-up in the worst-case.

System specific translations:

- `dlv/1`

  Takes a filename `<filename>` and pipes the file `<filename>.dlv` into `dlv`.

  There is a binary version `dlv/2` that allows for passing options to `dlv`.

  Try `dlv(<filename>,'')`.

Flags

- `set_labelling/1` (default: `number`; alternatively: `simple` or `formula`)
- `set_label_string/1` (default: 'l')
- `do_labels_beyond_negation/0` and `no_labels_beyond_negation/0` (default)
- `do_dictionary_file/0` and `no_dictionary_file/0` (default)
- `verbose_mode/0` (default) and `no_verbose_mode/0`

5 The files

5.1 Source code

- The compiler comes within the single file: `nlp41.pl`
- The new version supporting `gnt` is now available!

5.2 Documentation files

- Here is a dvi version of this file.
- Here is a PostScript version of this file.
- Here is a pdf version of this file.

6 Benchmarking

Benchmarks and experimental results are available here

7 What's new?

January 2002 A first stable version is provided.

August 2003 A new version, supporting `gnt`, is available. Also, it includes new comments, such as `nlp2dlp2gnt(Name)`, `nlp2str2gnt(Name)`, `nlp2htl2gnt(Name)`, `gnt(Name)`,

August 2003 Benchmarking examples and results are published.

THIS IS TO BE EXPANDED SOON (TS, Aug 14, 2003)!

8 Future gimmicks

- Better documentation.
- An example database.
- ...

9 Comments

... are highly welcome! Just send me email!

References

[1] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the International Conference on Logic Programming*, pages 1070–1080. The MIT Press, 1988.

[2] M. Gelfond and V. Lifschitz. Classical negation in logic programs and deductive databases. *New Generation Computing*, 9:365–385, 1991.

[3] V. Lifschitz, L. Tang, and H. Turner. Nested Expressions in Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.