Computing Default Logic Extensions: An Implementation

A.P. Courtney and **N.Y. Foo**
Basser Department of Computer Science
University of Sydney, NSW 2006, Australia
allenc | norman@cs.usyd.edu.au

G. Antoniou
Computing & Information Technology
Griffith University, QLD 4111, Australia
ga@cit.gu.edu.au

Default logic [3] is a useful formalism for reasoning with incomplete information, its intuitive characteristics making it particularly suited for applications. *Exten* is a system currently capable of computing first-order Reiter, Justified and Constrained default extensions. It is part of a project to create a full default logic workbench, with future work involving query evaluation, further support for default variants and integration with belief revision. As such, it has been implemented in an object-oriented manner, and is designed to facilitate experimentation. The interface is based around a small language, giving the user flexibility in editing default theories and changing various parameters (such as compute next n extensions or carry out 'success' checks every m steps).

Default reasoning is known to be computationally hard. One efficiency increasing technique used in *Exten* is stratification [1] which, if applicable, allows the computation of extensions in a modular way. *Exten* uses a forward-chaining approach and applies additional pruning techniques, some of which are outlined below.

```
PROCEDURE Compute-Ext( $\Pi$, Drest, Dout)
NotClosed := false;
M := { $\delta \in$ Drest | pre($\delta$) $\in$ In($\Pi$) } = { $\delta_1, \ldots, \delta_n$ };
FOR i := 1 TO $n$ DO
    Drest := Drest - { $\delta_i$ };
    IF $\forall\psi \in$ just($\delta_i$) are consistent with In($\Pi$) THEN
        Compute-Ext($\Pi \circ \delta_i$, Drest, Dout);
        NotClosed := true;
    Dout := Dout $\cup$ { $\delta_i$ };
IF NotClosed = false THEN
    IF $\forall\delta \in$ Dout are blocked by In($\Pi$) THEN
        ext := ext $\cup$ { In($\Pi$) };
```

(failure checking not shown, $In(\Pi)$ refers to the knowledge state where Π is the current default chain)

When a default δ_i has its prerequisite met at a node in the process tree (the map of default application chains used), all extensions containing δ_i can be found underneath this node. Thus δ_i can be safely removed from the set of available defaults *Drest*, resulting in smaller subtrees for remaining extensions and effective use of common tree branches. The justifications of defaults in *Dout* are checked when testing for closure of a process - if all are blocked we can conclude that a *new* extension has been found. This means that *Compute-Ext* will not produce multiple copies of the same extension.

Theorem 1. *Let $\pi = \pi_1 \circ \pi_2$ be a successful path of the process tree, closed under the defaults currently available (Drest). Further suppose that no default was added to Dout during the construction of π_2. Then no new extensions can be found by expanding the process tree under π_1.*

In many cases this will substantially reduce the search tree size. The method has a *local* effect to the search tree, ie. beneath π_1. In contrast, a more general method with a *global* effect is as follows.

Theorem 2. *Let π be a maximally successful path of the process tree $T = (W,D)$. Let M be the set of defaults in D (including those in Dout) that are blocked or failed along this path. Then every new extension computed after π must contain at least one default from M. M is called a goal.*

Instead of trying every available default at a given node, theorem 2 shows us that if a goal is applicable we only need to expand those subtrees starting with defaults in M. Different goals may be used at nodes along a path, with *Exten* using a heuristic prefering shorter goals.

A final comment is that all pruning methods described are general purpose in the sense that they apply to arbitrary default theories. We have already found they integrate well with stratification, and it seems plausible that other techniques could be added to offer further reductions in specific cases (*Exten already incorporates some optimizations for theories with normal defaults*). Proofs for the given theorems and algorithm can be found in [2].

References

[1] Cholewinski, P. Stratified Default Theories. *Proc. Computer Science Logic 1994*, Springer LNCS 993
[2] Courtney, A.P. Towards a Default Logic Workbench: Computing Extensions. Hons Thesis, Basser Dep. of Computer Science, University of Sydney, 1995.
[3] Reiter, R. A Logic for Default Reasoning. *Artificial Intelligence* 13(1980): 81-132