CS-Note* 14 Bill J. Ellis
May 27, 2002

POPLOG-PROLOG **versus** SICSTUS? (No Contest)

Contents

1 Introduction

The very early beginnings of NuSPADE are being constructed. During this process it became quite apparent that POPLOG-PROLOG is not the best tool for the job. Rather than struggle on with POPLOG-PROLOG it was decided[1] a more viable alternative was required. As has been hinted at in some previous notes (CS-note 7, CS-note 11 and CS-note 13) SICSTUS stands out as a viable alternative.

2 Advantages of SICSTUS

A short summary of the most significant advantages of SICSTUS over POPLOG-PROLOG in developing NuSPADE are listed below:

- **Well supported**
- **Up to date documentation**
- **Strong debugging support**
- **Direct access to sophisticated constraint logic programming**
- **Fully implemented module support**
- **Is available on Microsoft Windows and most Unix-ices**
- **Has big integers**
  This is also available in POPLOG-PROLOG, and relied upon in SPADE.
- **Is resource efficient for Prolog**
  POPLOG-PROLOG is implemented in POPLOG. It is not a geared-up Prolog implementation. Efficiency is important in a search intensive program like NuSPADE.
- **Interfaces with TCL/TK and Java for portable GUI**

*Notes in this series are for ϵ-baked ideas, for $1 \geq \epsilon \geq 0$. Only exceptionally should they be cited or distributed outwith the NuSPADE project (EPSRC GR/R24081).

[1] After finding myself grep-ing through the POPLOG-PROLOG source code to establish why some behaviour described in the manuals was not being observed in practise.

3 Porting SPADE to SICSTUS

The main reason for considering POPLOG-PROLOG is that SPADE is written in POPLOG-PROLOG. Implementing NuSPADE in SICSTUS will require porting SPADE to SICSTUS. A number of points about creating this SICSTUS version of SPADE (SICSTUS-SPADE) are considered in the following sections.

3.1 Breaks all the Core Principle

The core principles, outlined in CS-note 6, are repeated below:

Integrity: NuSPADE will only modify the SPADE proof state via the SPADE command level.

Compatibility: NuSPADE will preserve SPADE at its core, *i.e.* no modifications will be made to SPADE.

Transparency: NuSPADE will provide users with the look-and-feel of a SPADE proof session.

As SICSTUS is a completely different Prolog to that in POPLOG-PROLOG, the entire engine of SPADE will be changed. Although not in the manner expected, this is effectively breaking **Integrity**. Porting SPADE to Sicstus will require changes to SPADE, breaking **Compatibility**. Finally, some minor interface changes might also be expected. For example, the `callpro` command in SPADE gives full access to POPLOG. Clearly this will not be available in SICSTUS-SPADE, and leads to breaking **Transparency**.

Thus porting SPADE is in violation of all the core principles. The strategy used will be to minimise as far as is sensible the changes to SPADE in creating SICSTUS-SPADE. Once the port is complete the core principles will apply to SICSTUS-SPADE just as they would for SPADE.

3.2 SICSTUS Engine as Robust as POPLOG-PROLOG

As noted in CS-note 13 the SPADE proof checker relies on itself being correct. As SPADE is executed in POPLOG-PROLOG this is also a reliance on POPLOG-PROLOG being correct. Porting SPADE to SICSTUS changes this to a reliance on SICSTUS being correct.

Neither SICSTUS nor POPLOG-PROLOG are formally proven systems. Each may potentiality contain bugs that could result in a perfectly written SPADE deriving as true what is false. It is argued that as SICSTUS is a commercial product used in industry it is likely to be at least as robust as POPLOG-PROLOG.

If this issue is of concern, it would be quite possible to generate proofs using NuSPADE with SICSTUS-SPADE at its core. These could then be repeated inside SPADE, by executing the appropriate SPADE-script. This would give assurance that SICSTUS-SPADE is still behaving as SPADE. Further, the diversity of using two separate Prologs might improve confidence further than if SPADE (or SICSTUS-SPADE) had been used in isolation.

3.3 No NuSPADE for VAX Users

POPLOG-PROLOG is available on VAX, and hence SPADE is available on VAX. SICSTUS does not come as standard on VAX. Thus neither SICSTUS-SPADE nor NuSPADE will be available on VAX.

There is no VAX readily available at Heriot-Watt, preventing testing for this platform even if using POPLOG-PROLOG. Perhaps more significantly, the VAX line was retired by Compaq in 1999[2]. It is speculated that the absence of a VAX version of NuSPADE is unlikely to be a issue. Especially since it will still be possible to use SPADE on VAX regardless of the availability of NuSPADE.

3.4 Does not Appear a Major Obstacle

Initial efforts suggest that porting SPADE to SICSTUS should not pose many problems. The main areas of incompatibility are focused around non standard Prolog features such as exception handling and interfacing with the operating system.

3.5 The Licence Server

SICSTUS supports bidirectional communication through C. It should therefore be quite possible to port the SPADE interface to the FLEXLM licence server (mainly existing in `getlicence.pl`) to SICSTUS-SPADE. However, this is expected to be a fiddly task, requiring knowledge of FLEXLM. Thus it shall be given a lower priority in porting SPADE.

[2]See: http://www.compaq.com/alphaserver/vax/vax_letter_final.html

3.6 Mature SPADE Changes Slowly

It is considered that SPADE changes less dramatically and at a slower pace than other SPARK tools. This should make it more straight forward to ensure that SICSTUS-SPADE stays in sync with SPADE.

3.7 Record Port Changes

The changes made to SPADE during the port will be recorded in full. It should be straight forward to identify the changes made to SPADE in SICSTUS-SPADE. The intention is to demonstrate this by being able to automatically port SPADE to and from SICSTUS and POPLOG-PROLOG by switching around appropriate sections of code.

3.8 Non Port Changes To SPADE Then SICSTUS-SPADE

In the process of building NuSPADE it is likely that a number of small changes will be made to SPADE. For example the restrictive mode outlined in CS-note 8. These changes should be made to SPADE and then ported to SICSTUS-SPADE, rather than directly to SICSTUS-SPADE. Doing this will ensure that SPADE is always the main source distribution, with SICSTUS-SPADE as nothing more than a record of minimum changes to make SPADE run on SICSTUS.

3.9 Publishing NuSPADE

So long as Praxis do not have access to SICSTUS they will be unable to compile NuSPADE in-house. Thus binaries[3] for NuSPADE will be provided for execution on the platform required. Even where considering POPLOG-PROLOG this was the intended method for distributing NuSPADE.

It is noted that this may limit the potential for Praxis users to pinpoint the source of any errors they might encounter. It may be beneficial to package the NuSPADE source as documentation even though Praxis currently lack a copy of SICSTUS to compile it.

4 In Context Of Other SPARK Tools

4.1 The EXAMINER

4.1.1 Connections

The EXAMINER generates the VCG, RLS and FDL files read in by SPADE.

4.1.2 Potential Issues

In the VCG files generated by the EXAMINER their is a space between functor and arguments where the functor is `for_all` or `for_some`. For example:

```
C1: for_some (p_: integer, ((p_ >= indexrange__first) and (
    p_ <= indexrange__last + 1)) and (( for_all (q_:
    integer, ((q_ >= indexrange__first) and (q_ <= p_ - 1)) -> (element(
    flag, [q_]) = red))) and ( for_all (r_: integer, ((
    r_ >= p_) and (r_ <= indexrange__last)) -> (element(
    flag, [r_]) = white))))) .
```

Although POPLOG-PROLOG can parse these, the same is not true for SICSTUS. A possible solution is to introduce `for_all` and `for_some` as Prolog operators allowing the terms to be read in inside SICSTUS. For example:

```
:- op(800        ,fy   ,for_all                 ).
:- op(800        ,fy   ,for_some                ).
```

It will then be necessary to perform some post-processing on these terms to convert these to the functor(argument) form. Further, the SIMPLIFIER does not generate the space separated format. It will be necessary to implement the post-processor such that it also caters for the non-space behaviour.

An alternative solution is to change the EXAMINER such that it does not leave a space at the end of these functors.

[3] A runtime system and saved NuSPADE image.

4.2 The SIMPLIFIER

4.2.1 Connections

The SIMPLIFIER generates modified VCG files as SIV files to be read in by SPADE.

4.2.2 Potential Issues

Unlike the EXAMINER, the SIMPLIFIER does not leave any inconvenient spaces after any functors. There is no issues here.

4.3 BUILDLIB

4.3.1 Connections

BUILDLIB generates rule library files that are read in by SPADE.

4.3.2 Potential Issues

BUILDLIB outputs terms in quoted Prolog form to be read in and asserted directly in SPADE. Exactly the same behaviour can be repeated in SICSTUS-SPADE.

4.4 SPADE

4.4.1 Connections

SPADE can save its current progress and resume later. Further, SPADE can read in a previously saved SPADE-script to execute a collection of commands.

4.4.2 Potential Issues

SICSTUS-SPADE will support the same commands of SPADE and should have compatible interaction. Thus the SPADE-scripts should be applicable in both SPADE and SICSTUS-SPADE.

Saving is implemented in SPADE by storing the essential state of the system as a collection of predicates. These are then loaded during a new SPADE session to resume. The essential state of SICSTUS-SPADE and SPADE is likely to be the same. Thus resuming in this manner should be possible[4].

5 Conclusion

NuSPADE is a research system building on the strengths of SPADE. It is necessary for NuSPADE to be implemented in the same language as SPADE. SPADE currently runs in POPLOG-PROLOG. However, POPLOG-PROLOG is not considered to be the best tool for building NuSPADE. Although SPADE runs in POPLOG-PROLOG, it is not essential. A port to another Prolog, in particular the strong SICSTUS, is considered quite viable. This would then allow for NuSPADE to be developed in this stronger Prolog. While there are some negative aspects of not using the home language of SPADE, these are mostly minor.

[4]Note that the save-resume process in general, irrespective of which Prolog is used, is considered an area for improvement. Although not yet published in a CS-note there is rough plans for a stronger proof recording mechanism. It is observed that any proof state can be recovered by taking the original problem and applying the correct series of proof steps.