

# SPADEase: The Good, the Bad and the Ugly

---

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Presentation</b>	<b>2</b>
2.1	Slide 1 . . . . .	2
2.2	Slide 2 . . . . .	2
2.3	Slide 3 . . . . .	3
2.4	Slide 4 . . . . .	3
2.5	Slide 5 . . . . .	4
2.6	Slide 6 . . . . .	4
2.7	Slide 7 . . . . .	5
2.8	Slide 8 . . . . .	6
2.9	Slide 9 . . . . .	7
2.10	Slide 10 . . . . .	7
2.11	Slide 11 . . . . .	8
2.12	Slide 12 . . . . .	9
2.13	Slide 13 . . . . .	10
2.14	Slide 14 . . . . .	10
2.15	Slide 15 . . . . .	11
2.16	Slide 16 . . . . .	11
2.17	Slide 17 . . . . .	12
2.18	Slide 18 . . . . .	12
2.19	Slide 19 . . . . .	13
2.20	Slide 20 . . . . .	14
2.21	Slide 21 . . . . .	14
2.22	Slide 22 . . . . .	15
2.23	Slide 23 . . . . .	15
2.24	Slide 24 . . . . .	16
2.25	Slide 25 . . . . .	16

---



---

\*Hums are short notes intended for distribution between those involved with EPSRC grant GR/T12289/01. Hums describe  $\epsilon$ -baked ideas, where  $1 \geq \epsilon \geq 0$ . (Hum refers to both the Praxis Humming bird and Winnie-the-Pooh's indirect approach to writing: "Poetry and Hums aren't things which you get, they're things which get you.").

# 1 Introduction

Approaching the closing five weeks of the SPADeEase project, Andrew Ireland is visiting Praxis High Integrity Systems to see what has been achieved so far and guide activity in these closing weeks. To get Andrew, and Praxis, up to speed to on the project, a lunchtime seminar is being held. Here the slides and some notes from this seminar are collected.

## 2 Presentation

### 2.1 Slide 1

SPADEase: The Good, the Bad and the Ugly

Bill J Ellis

Dependable Systems Group  
School of Mathematical &  
Computer Sciences  
Heriot-Watt University  
Edinburgh

(On secondment at)  
Praxis High Integrity Systems  
Bath

HERIOT WATT UNIVERSITY

Praxis High Integrity Systems

### 2.2 Slide 2

Introduction

Overview

- Introduction
  - Context
  - Objectives
- Background
  - Program proof in SPARK
  - Proof Planning
- SPADEase features
  - Good, Bad, and Ugly
- Ongoing Work
- Conclusions

- Begin by explaining the context for this project.
- Outline the key objectives of this project.
  - And how we plan to meet these.
- Give high level background to:

- Program proof in SPARK.
- Proof Planning.
- Describe key features of the new system SPADEase:
  - Categorising these as good, bad or ugly.
- Outline work for the remainder of the project.
- Finish with some conclusions.

### 2.3 Slide 3

**Introduction**

NuSPADE

**Duration:** 2001 ⇔ 2004  
**Funding:** EPSRC critical systems programme (GR/R24081)

**Research associate:** Bill Ellis  
**Principle investigator:** Andrew Ireland  
**Collaboration:** Praxis

**Aim:** Investigate the role of proof planning within the SPARK approach to high integrity software  
**Nature:** Traditional research project

### 2.4 Slide 4

**Introduction**

SPADEase

**Duration:** Jan 2005 ⇔ July 2005  
**Funding:** EPSRC research assistant industrial secondment scheme (RAIS) (GR/T11289/01)  
**Context:** Builds upon the NuSPADE project

**Research associate:** Bill Ellis (*Seconded to Praxis*)  
**Principle investigator:** Andrew Ireland  
**Collaboration:** Praxis

**Aim:** Towards increased verification automation for high integrity software engineering  
**Nature:** Knowledge transfer, industrial secondment

## 2.5 Slide 5

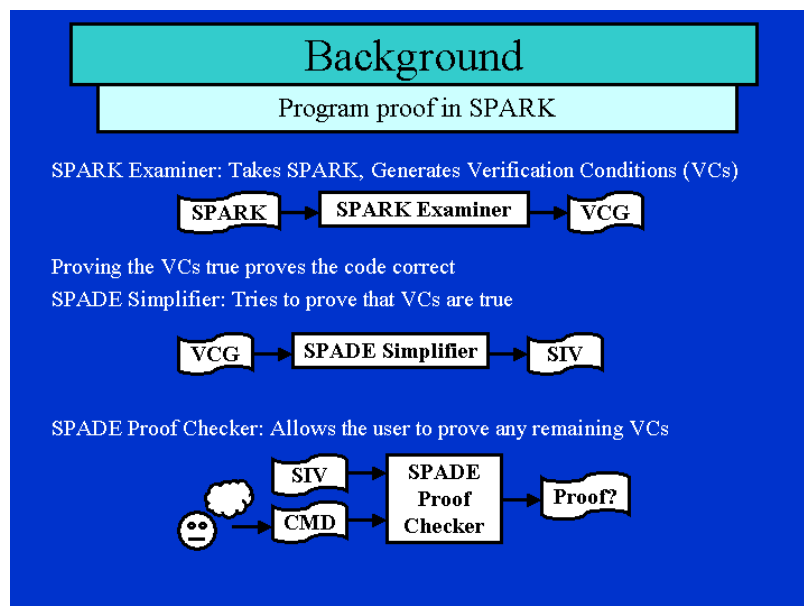
# Introduction

## SPADEase Objectives

- Key RAIS Goals:
  - Transfer knowledge from original research project:
    - Demonstrate viable industrial system
    - Implement this system as an extension to the SPARK approach
    - Evaluate and publish
  - Training in an industrial environment:
    - Seek out industrial training within Praxis
    - Work in the SPARK corner for six months

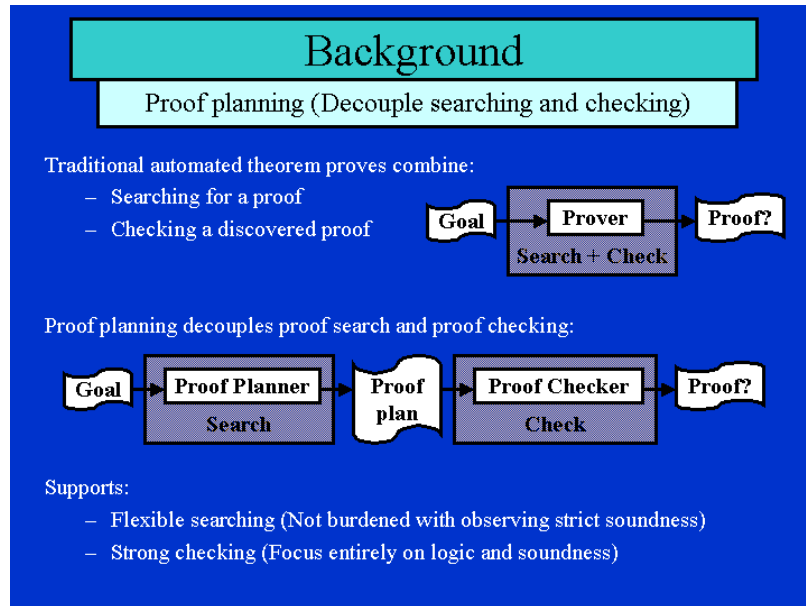
- There are two key goals that every RAIS project should try to reach.
- The first is to transfer knowledge from the original research project.
  - Demonstrate a viable industrial system.
    - \* To show this knowledge is useful.
  - Implement this system as an extension to the existing SPARK tools.
    - \* To show this knowledge is directly applicable.
  - Evaluate and publish.
    - \* To provide guidance in applying this knowledge.
- The second RAIS goal is to train the research associate in an industrial environment.
  - Seek out industrial training within Praxis.
    - \* Attend any courses, or Praxis meetings.
  - Primarily, just work in the SPARK corner for six months.

## 2.6 Slide 6



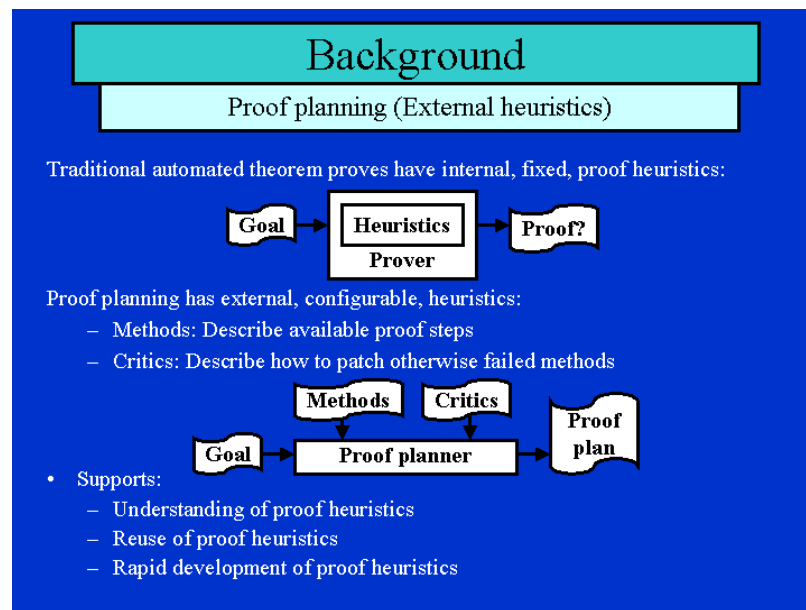
- SPARK Examiner takes as input a SPARK program.
  - Generates Verification Conditions (VCs) in a VCG file.
  - Proving every VC is true proves the code is correct.
- SPADE Simplifier takes these VCs as input.
  - Tries to automatically prove them as true.
  - And returns any it can not prove in an SIV file.
- SPADE Proof Checker supports the user to prove any remaining VCs.
  - The user interacts with the Proof Checker.
  - To construct a proof that convinces the Proof Checker that the VC is true.

## 2.7 Slide 7



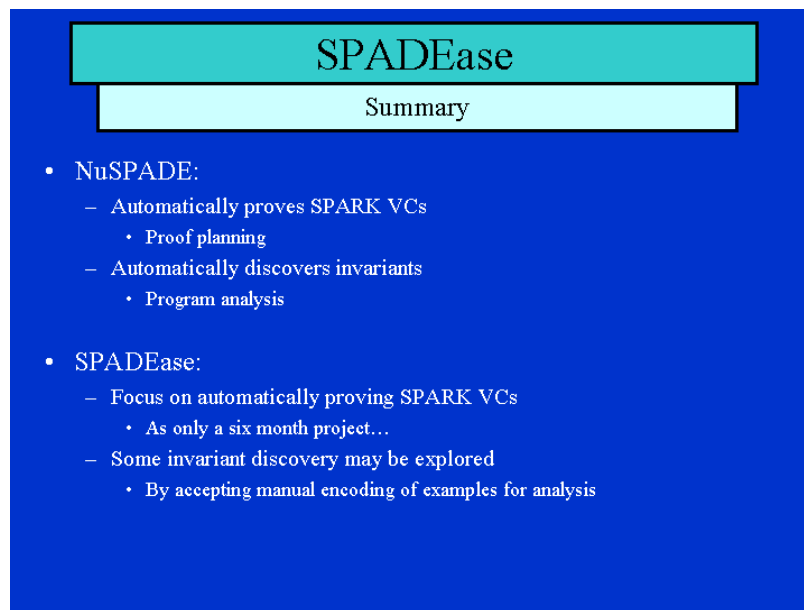
- Traditional automated theorem provers interleave the task of:
  - Searching for a proof.
  - Checking a discovered proof.
- Proof planning decouples these separate tasks:
  - Having an explicit searching component.
  - And an explicit proof checking component.
- This supports more flexible searching:
  - As the search is not burdened with demonstrating strict soundness.
- This supports stronger proof checking:
  - As the checking focuses entirely on logic and soundness.

## 2.8 Slide 8



- Traditional automated theorem proves have internal, fixed, proof heuristics.
- While proof planning has external, configurable, heuristics.
  - Methods describe available proof steps.
  - Critics are expressed in the same manner as methods.
  - Only they describe how to patch the failure of methods.
  - Both are described using a high level proof description language.
- Doing this supports understanding of proof strategies.
  - As every heuristic is expressed in the same style.
  - Using a high level description language.
- Further, this eases the reuse of proof heuristics.
  - As presenting the heuristics as external components reduces them to their essence.
  - making them less implementation dependent.
  - And much more straight forward to transfer across different systems.
- And supports the rapid development of proof heuristics.

## 2.9 Slide 9

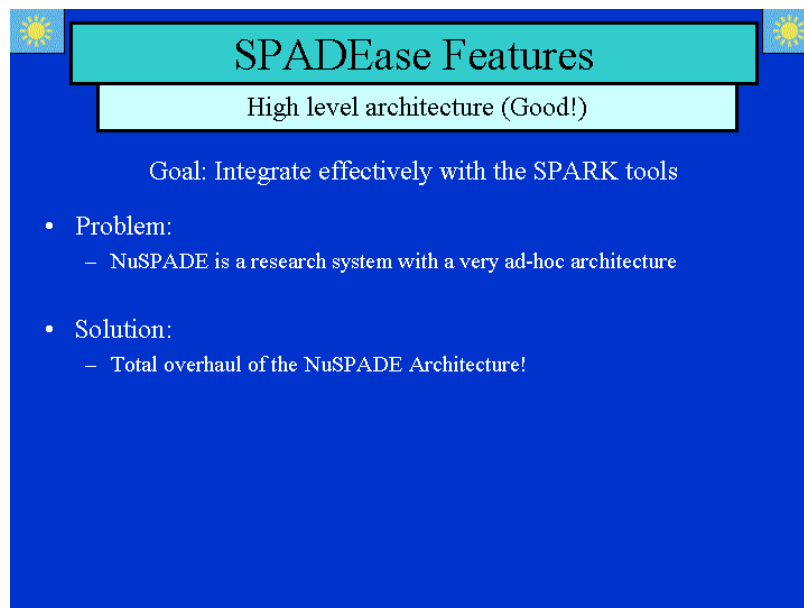


SPADEase

Summary

- NuSPADE:
  - Automatically proves SPARK VCs
    - Proof planning
  - Automatically discovers invariants
    - Program analysis
- SPADEase:
  - Focus on automatically proving SPARK VCs
    - As only a six month project...
  - Some invariant discovery may be explored
    - By accepting manual encoding of examples for analysis

## 2.10 Slide 10



SPADEase Features

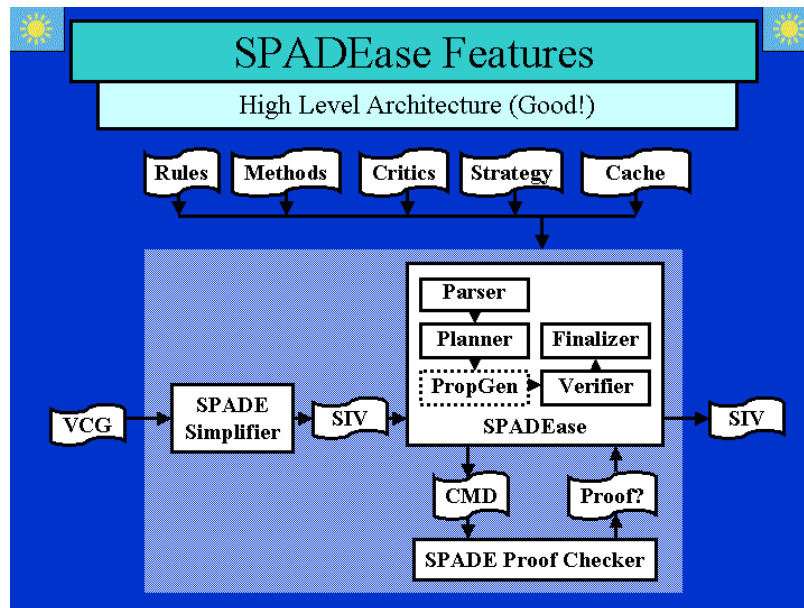
High level architecture (Good!)

Goal: Integrate effectively with the SPARK tools

- Problem:
  - NuSPADE is a research system with a very ad-hoc architecture
- Solution:
  - Total overhaul of the NuSPADE Architecture!

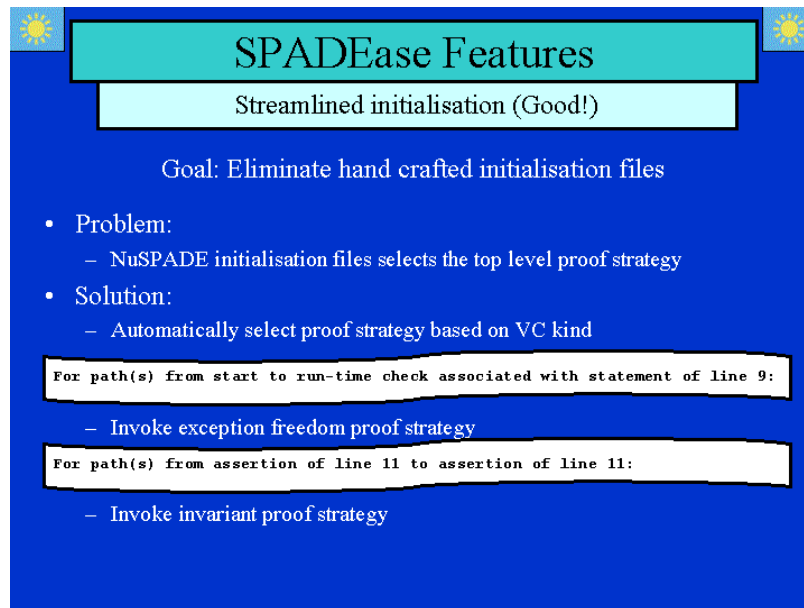
- Want to integrate effectively with the SPARK tools.
- However.
  - NuSPADE is a research system.
  - And has a very ad-hoc architecture.
- Solution.
  - Create a brand new architecture for SPADEase.

## 2.11 Slide 11



- This is the architecture for SPADEEase.
- Note how SPADEEase performs the same operation as the Simplifier.
  - A VCG file is provided as input.
  - And an SIV is provided as output.
- To begin, the Simplifier is called to prove as many VCs as possible.
- The core of SPADEEase is then applied to any remaining VCs.
- SPADEEase comprises of four discrete interacting components.
- Begins with the Parser.
  - Parse and process all data required for planning.
  - Includes all the rules. Built in rules, and RLS, and RUL rule.
  - All planning components: Methods, Critics and a Strategy for controlling these.
  - And a data cache.
- The planner takes the result from the parser.
  - And plans every remaining VC.
- In a complete version of SPADEEase program analysis would be performed by PropGen.
  - This would try to discover program invariants in reaction to failed plans.
  - At best, a limited version of PropGen will be seen in SPADEEase.
- The verifier takes the result from the planner.
  - Every discovered plan is converted into a CMD log.
  - And checked within the SPADE Proof checker.
- The finaliser inspects the results of the Proof Checker.
  - And generates the final SIV file.

## 2.12 Slide 12



The slide features a blue background with a light blue header box containing the title "SPADEase Features". Below the title is a white box with the text "Streamlined initialisation (Good!)". The main content is a list of bullet points and two code snippets. The first bullet point is "Problem:" with a sub-bullet "– NuSPADE initialisation files selects the top level proof strategy". The second bullet point is "Solution:" with a sub-bullet "– Automatically select proof strategy based on VC kind". The first code snippet is "For path(s) from start to run-time check associated with statement of line 9:" followed by a sub-bullet "– Invoke exception freedom proof strategy". The second code snippet is "For path(s) from assertion of line 11 to assertion of line 11:" followed by a sub-bullet "– Invoke invariant proof strategy".

### SPADEase Features

Streamlined initialisation (Good!)

Goal: Eliminate hand crafted initialisation files

- Problem:
  - NuSPADE initialisation files selects the top level proof strategy
- Solution:
  - Automatically select proof strategy based on VC kind

For path(s) from start to run-time check associated with statement of line 9:

- Invoke exception freedom proof strategy

For path(s) from assertion of line 11 to assertion of line 11:

- Invoke invariant proof strategy

- Want to eliminate hand crafted initialisation files.
  - As these prevent the automatic application of SPADEase.
- A Problem.
  - NuSPADE requires an initialisation file to select the top level proof strategy.
- Solution.
  - Automatically select the proof strategy based on VC kind.
- So, for example, if the VC is declared as being:
  - For path(s) from start to run-time check associated with statement of line 9:
  - Then it is a run-time VC.
  - And the top level exception freedom strategy is executed.
- And if the VC is declared as being:
  - For path(s) from assertion of line 11 to assertion of line 11:
  - Then the invariant proof strategy is executed.

## 2.13 Slide 13

### SPADEase Features

Streamlined initialisation (Good!)

- Problem:
  - NuSPADE initialisation files filter the available rules
- Solution:
  - Do not perform any rule filtering
  
- Problem:
  - Every visible rule requires time consuming pre-processing
  - No rule filtering creates a lengthy initialisation phase...
- Solution:
  - Rule processing is cached
    - Takes a few hours to generate the rule cache
    - Takes a few seconds to load the rule cache

- Another problem.
  - The initialisation file is also used to filter the available rules.
- Solution.
  - Is straight forward.
  - Do not perform any rule filtering.
- However, this creates a new problem.
  - Every visible rule requires time consuming pre-processing.
- Fortunately, this processing always returns the same results for the same rule.
- Solution.
  - Cache all of this rule processing.
  - Although it takes a few hours to generate the rule cache.
  - It only takes a few seconds to load the rule cache.

## 2.14 Slide 14

### SPADEase Features

Configurable strategies (Good!)

Goal: Provide more control over proof strategies

- In proof search methods are applied to goals
  - Successful methods:
    - Prove a goal is true
    - Create sub-goals (which are hopefully easier to prove true)

```
graph TD
    subgraph Panel1
        G1_1[Goal 1] --> G2_1[Goal 2]
    end
    subgraph Panel2
        G1_2[Goal 1] -- M1 --> G2_2[Goal 2]
        G2_2 -- M2 --> P1((Proof!))
    end
    subgraph Panel3
        G1_3[G] --> G2_3a[G]
        G1_3 --> G2_3b[G]
        G1_3 --> G2_3c[G]
        G2_3a --> G3_3a[G]
        G2_3b --> G3_3b[G]
        G2_3c --> P3((P))
    end
```

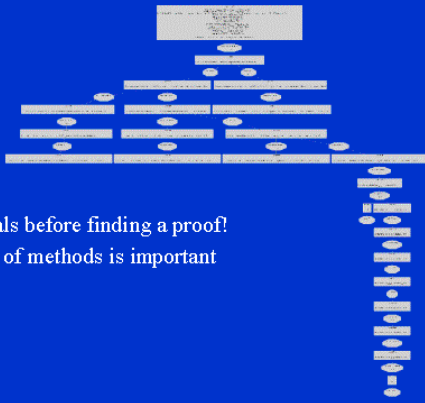
Strategies control which methods are applied to which goals and when

2.15 Slide 15

## SPADEase Features

Configurable strategies (Good!)

A real example...



Proof search can create many goals before finding a proof!

- So controlling the application of methods is important

2.16 Slide 16

## SPADEase Features

Configurable strategies (Good!)

Strategies in NuSPADE:

Strategy	Method	...	
Goal 1	S1	Ma Mb Mc Md	Begin with a strategy
Goal 1	Ma	S1 Mb Mc Md	Method fails
Goal 1	Mb	S1 Mc Md	Method succeeds
Goal 1	S1	Mc Md	
Goal 2	S1	Ma Mb Mc Md	

NuSPADE:

- Adopt the same strategy for all goals
- Always considers every method application to every goal before terminating

2.17 Slide 17

## SPADEase Features

Configurable strategies (Good!)

- Problem:
  - Some sub-goals are (heuristically) better than their parents
    - So do not always want to search alternative methods at the parent
  - Some sub-goals are (heuristically) best tackled with particular methods
    - So do not always want to adopt the same strategy as for the parent
- Solution:
  - Replace list of methods with a list of method callers, that:
    - Select which method to call
    - If the method is successful:
      - What strategy to adopt at any sub-goals
      - If remaining methods at the parent should be removed (cut)

2.18 Slide 18

## SPADEase Features

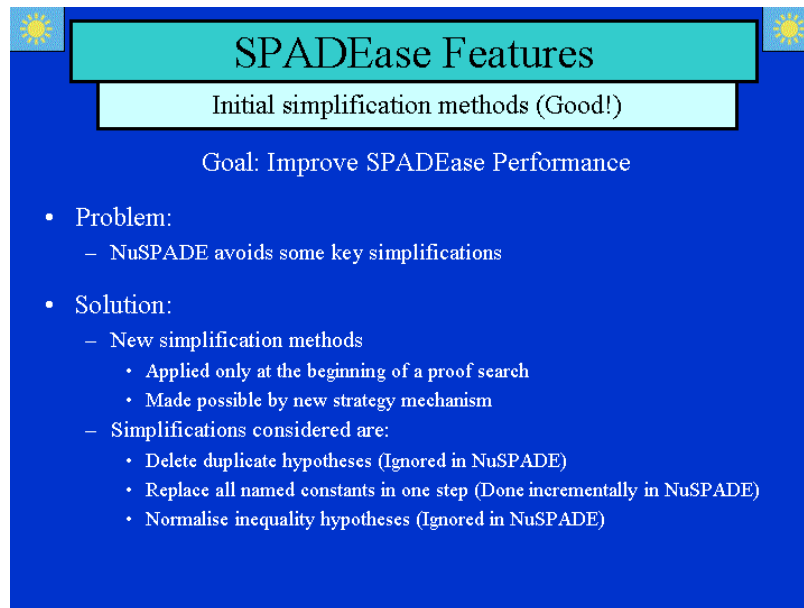
Configurable strategies (Good!)

Strategies in SPADEase:

Goal 1	S1	Ca	Cb	Cc	Cd	Begin with a strategy
Goal 1	Ma	S1	Cb	Cc	Cd	Ca Ma S2 no Method fails
Goal 1	Mb	S1	Cc	Cd		Cb Mb S2 Yes Method succeeds
Goal 1	S1					Goal 2 S2 Cx Cy Cz

SPADEase:

- Can select strategy for sub-goals (improved control)
- Can chose to cut remaining method applications (more efficient)

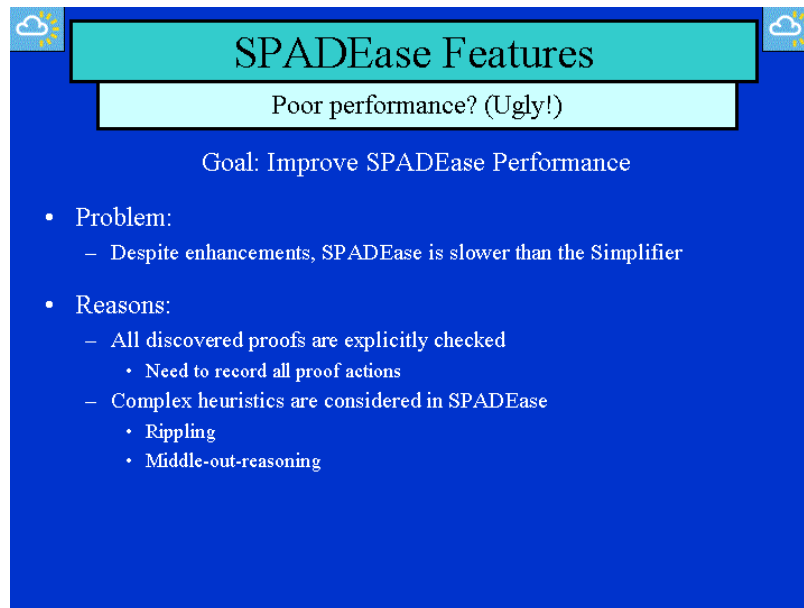


The slide features a blue background with a teal title box at the top containing the text "SPADEease Features". Below the title is a white box with the text "Initial simplification methods (Good!)". Underneath that is the text "Goal: Improve SPADEease Performance". The main content is a bulleted list:

- Problem:
  - NuSPADE avoids some key simplifications
- Solution:
  - New simplification methods
    - Applied only at the beginning of a proof search
    - Made possible by new strategy mechanism
  - Simplifications considered are:
    - Delete duplicate hypotheses (Ignored in NuSPADE)
    - Replace all named constants in one step (Done incrementally in NuSPADE)
    - Normalise inequality hypotheses (Ignored in NuSPADE)

- Want to improve SPADEease Performance.
- Is particularly important to be industrially applicable.
- NuSPADE avoids some key simplifications.
  - Because this was less important in a research system.
  - And difficult to employ effectively using the NuSPADE strategy mechanism.
- Solution is new start up simplification methods.
- These apply only once, just after a goal is loaded.
  - Delete double hypotheses.
  - Which brings an obvious efficiency boost.
  - All named constants are replaced with their values in a single method call.
  - NuSPADE performed this incrementally via several method calls.
  - Which is much slower.
  - Place all inequality hypotheses into a standard form.
  - Eliminating the need for search within methods that process inequalities.

## 2.20 Slide 20



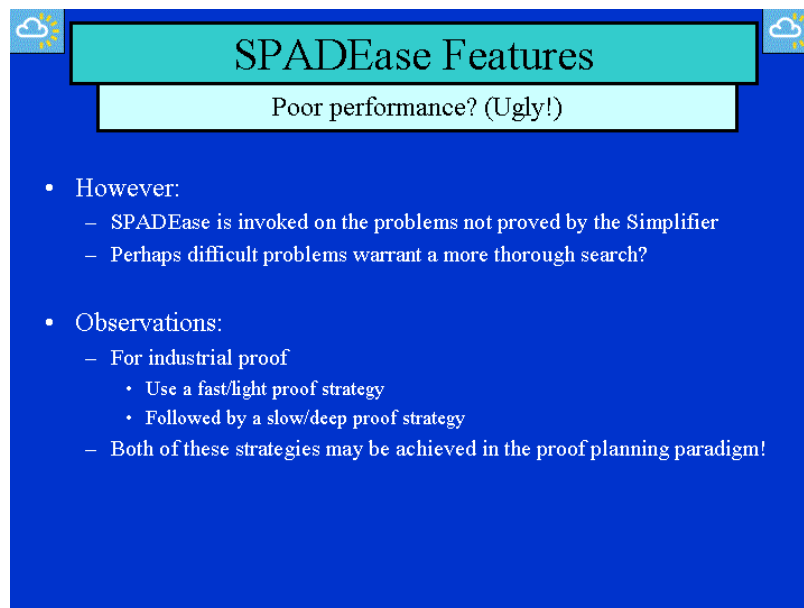
**SPADEase Features**

Poor performance? (Ugly!)

Goal: Improve SPADEase Performance

- Problem:
  - Despite enhancements, SPADEase is slower than the Simplifier
- Reasons:
  - All discovered proofs are explicitly checked
    - Need to record all proof actions
  - Complex heuristics are considered in SPADEase
    - Rippling
    - Middle-out-reasoning

## 2.21 Slide 21



**SPADEase Features**

Poor performance? (Ugly!)

- However:
  - SPADEase is invoked on the problems not proved by the Simplifier
  - Perhaps difficult problems warrant a more thorough search?
- Observations:
  - For industrial proof
    - Use a fast/light proof strategy
    - Followed by a slow/deep proof strategy
  - Both of these strategies may be achieved in the proof planning paradigm!

## 2.22 Slide 22

**SPADEase Features**

Simplification explodes (Ugly!)

Goal: Support term simplification

- Problem:
  - Rules are not manually filtered in SPADEase
  - Many more rules are visible in SPADEase than in NuSPADE
  - Leads to a search explosion in NuSPADE's term simplification method...
- Solution:
  - Introduce a new term simplification method
  - Exploit the heuristic of bringing together related terms
  - This method exists on paper... but not in code

## 2.23 Slide 23

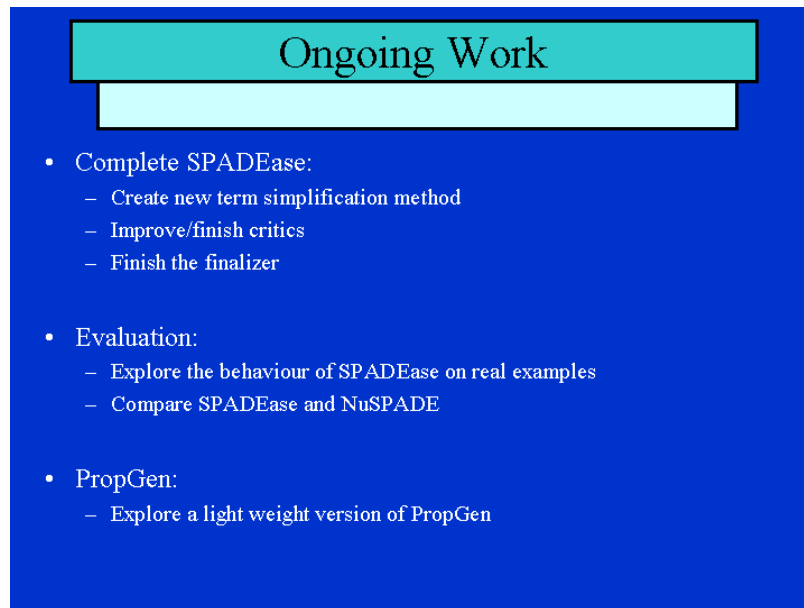
**SPADEase Features**

Modifying the Proof Checker (Bad!)

Goal: Check plans in the SPADE Proof Checker

- Problem:
  - The Checker's behaviour is not (realistically) predictable
- Solution:
  - NuSPADE: Many small changes to the Checker
    - Complex CMD Files
    - Does not address all areas of unpredictably
  - SPADEase: One large change to the Checker
    - Clear CMD files
    - Fully predictable Checker
    - But is it sound?!
  - Ideally: Proof Checker should be predictable out-of-the-box

## 2.24 Slide 24

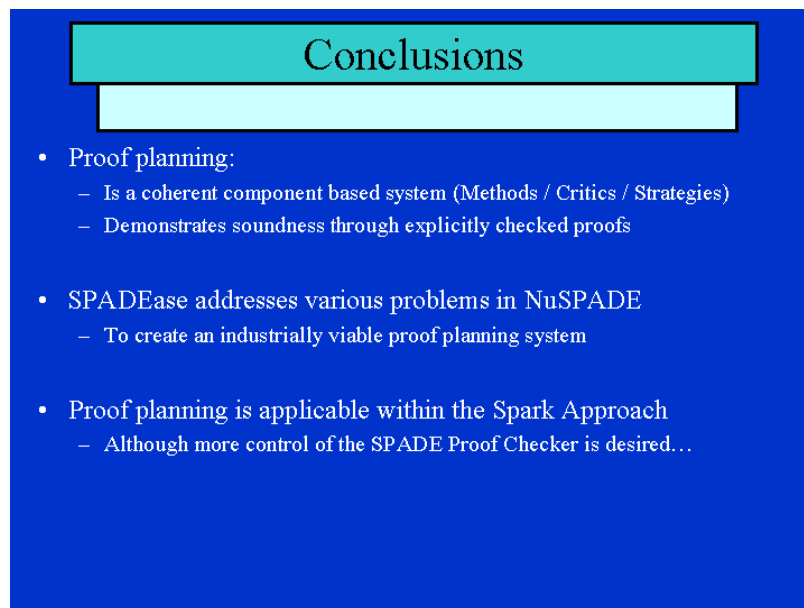


The slide features a blue background with a teal header box containing the text "Ongoing Work". Below the header is a light blue rectangular box. The main content consists of a bulleted list of tasks and evaluations.

### Ongoing Work

- Complete SPADEase:
  - Create new term simplification method
  - Improve/finish critics
  - Finish the finalizer
- Evaluation:
  - Explore the behaviour of SPADEase on real examples
  - Compare SPADEase and NuSPADE
- PropGen:
  - Explore a light weight version of PropGen

## 2.25 Slide 25



The slide features a blue background with a teal header box containing the text "Conclusions". Below the header is a light blue rectangular box. The main content consists of a bulleted list of conclusions regarding proof planning and SPADEase.

### Conclusions

- Proof planning:
  - Is a coherent component based system (Methods / Critics / Strategies)
  - Demonstrates soundness through explicitly checked proofs
- SPADEase addresses various problems in NuSPADE
  - To create an industrially viable proof planning system
- Proof planning is applicable within the Spark Approach
  - Although more control of the SPADE Proof Checker is desired...