Defeasible Conditionals in Answer Set Programming

Project Aims

In this project, we evaluated the usefulness of computing RC using a declarative language, ASP. The aims were:

1. Devise an implementation of Rational Closure based on Answer Set Programming.
2. Generate knowledge bases with ASP.
3. RC Entailment Interpreter.

Background

Propositional Logic

Propositional logic represent facts about the world. For example, we can represent "**Birds fly**", "**Penguins are birds**" and "**Penguins do not fly**" in the following knowledge base: $\mathcal{K} = \{p \rightarrow b, b \rightarrow f, p \rightarrow \neg f\}$

Classical Reasoning

Classical reasoning facilitates drawing conclusions. From the above, it would conclude penguins don't exist, as they cannot both fly and not fly.

$\mathcal{K} \models \neg p$

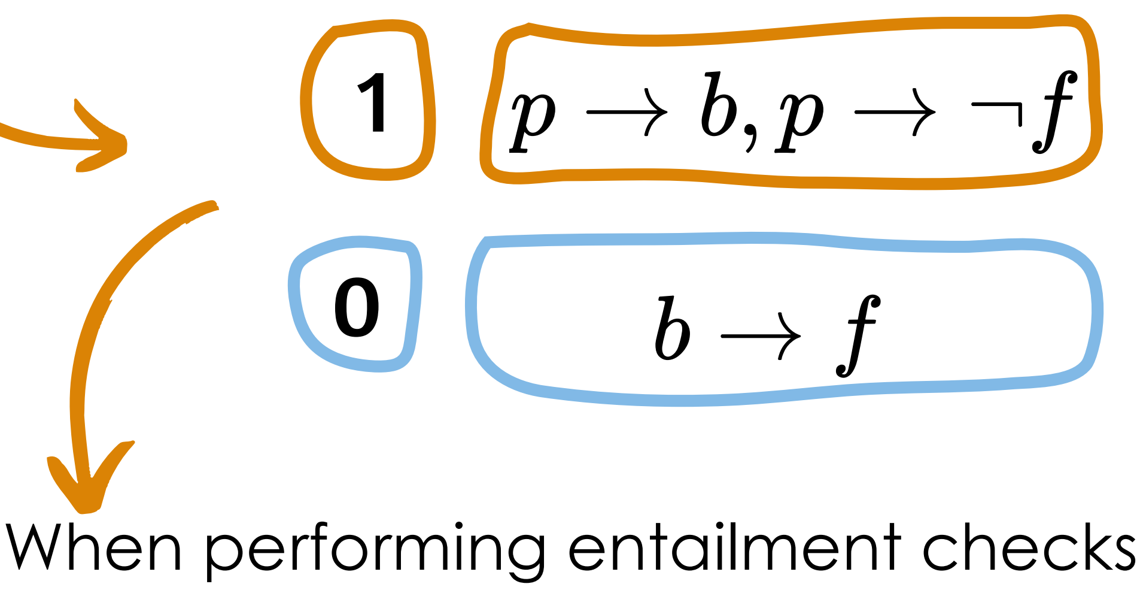
Defeasible Reasoning

Defeasible reasoning accounts for exceptions to typical cases. For instance, while birds usually fly, penguins are exceptions.

$\mathcal{K} \not\approx \neg p$

Rational Closure (RC)

Ranks statements based on their level of specificity and facilitates defeasible entailment



When performing entailment checks, lower ranks are removed leaving the relevant information needed to answer a query.

Answer Set Programming

Answer Set Programming (ASP) is a declarative programming paradigm that focuses on specifying what the desired solution is, rather than detailing how to compute it.

$$fly(X) : -bird(X).$$

$$bird(X) : -penguin(X).$$

$$-fly(X) : -penguin(X).$$

Declarative RC

Aim

Devise a working prototype for RC in ASP. Two approaches were used to develop two prototypes for RC.

1. Search-based Approach

This approach follows the ASP problem-solving methodology.

- Identify properties of solutions
- Enumerate the search space
- Find solution among all candidates

Bottlenecks

- Search space size increases exponentially
- Require optimisation statement to find solution

2. Recursive Approach

Expressed RC recursively and eliminated bottlenecks.

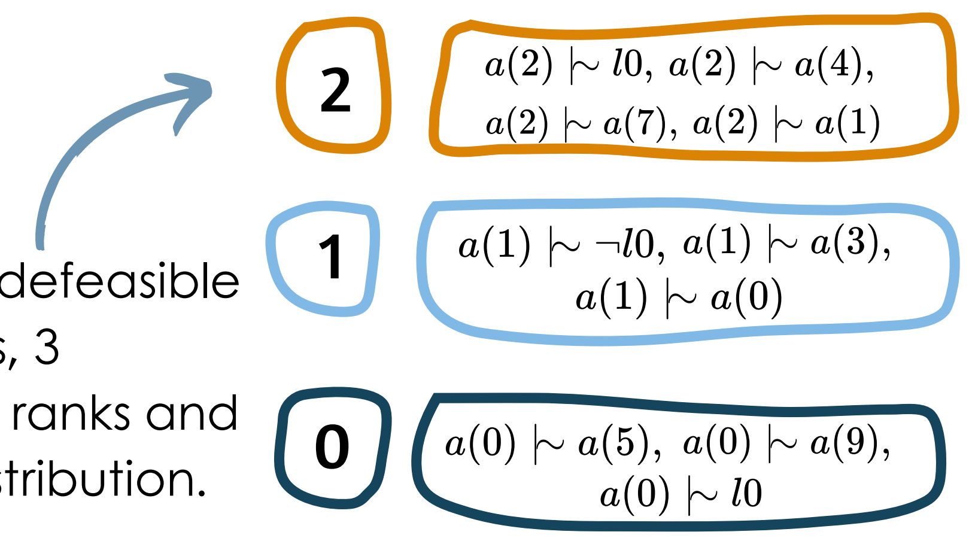
KB Generator

Aim

Devised an ASP implementation that generated defeasible knowledge bases based on parameters.

Features of generator:

- Number of defeasible ranks
- Number of defeasible implications
- Classical statements included in the knowledge base
- Distribution of statements amount the ranks
- Encoded defeasible statements which are classical


2
$a(2) \mid\!\sim l0,\ a(2) \mid\!\sim a(4),$
$a(2) \mid\!\sim a(7),\ a(2) \mid\!\sim a(1)$
1
$a(1) \mid\!\sim \neg l0,\ a(1) \mid\!\sim a(3),$
$a(1) \mid\!\sim a(0)$
0
$a(0) \mid\!\sim a(5),\ a(0) \mid\!\sim a(9),$
$a(0) \mid\!\sim l0$

KB with 10 defeasible statements, 3 defeasible ranks and uniform distribution.

RC Interpreter

Aim

Developed a software tool to allow the interfacing of RC in ASP, and visualised an explanation service for RC entailment process.

Features of RCI

Entailment Checker:

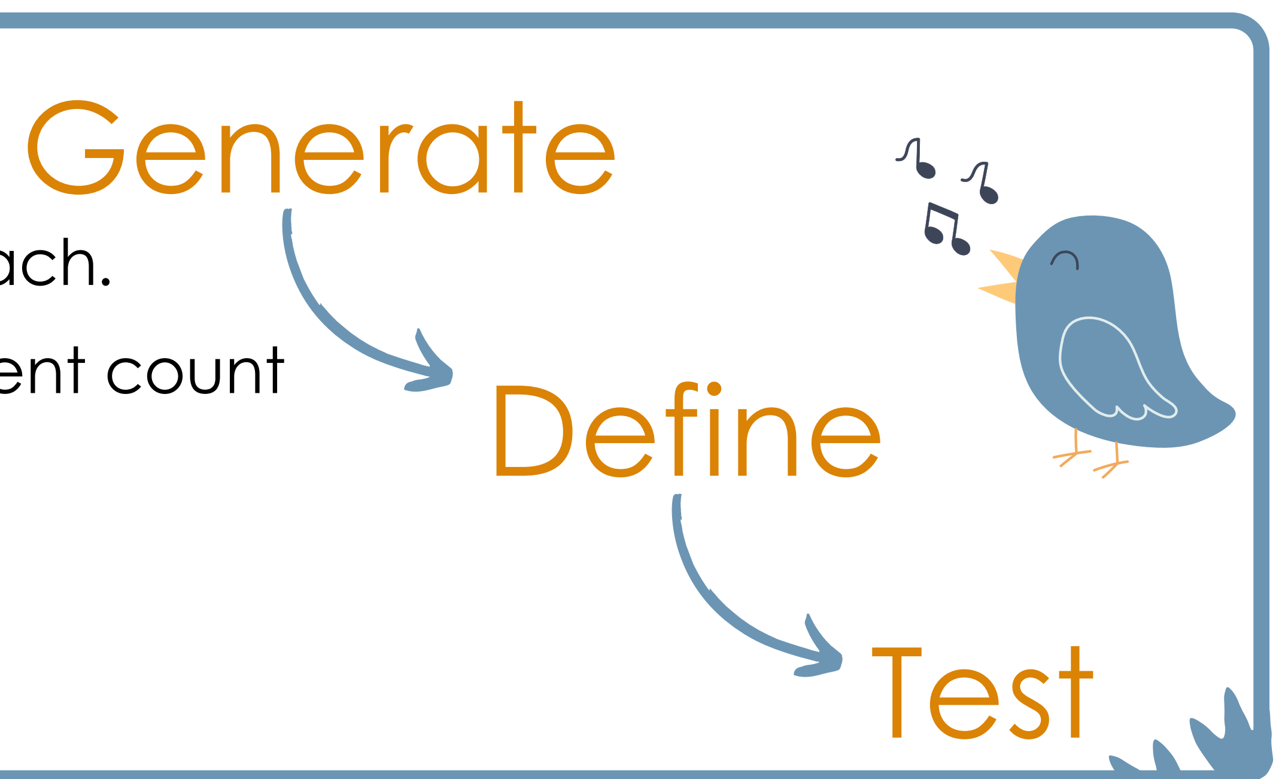
- Input, upload, or generate a knowledge base.
- Enter in a defeasible query.
- Determine entailment of a given query.

Visualisation Page:

- Step through the process of how the entailment result was reached.
- Provide explanations.
- Step through the BaseRank and Rational Closure algorithms.

Conclusions

- Search Based Approach resulted in bottlenecks which affected the performance of RC.
- Recursive approach eliminated all bottlenecks and outperformed the Search Based Approach.
- ASP was useful in generating large knowledge bases, however increasing ranks and statement count affected performance.
- ASP is a sophisticated language with complex tools which has a steep learning curve.
- ASP allows for programs to be concise and abstracted.



Authors

Sibusiso Buthelezi bthsib016@myuct.ac.za

Racquel Dennison dnnrac003@myuct.ac.za

Jack Mabotja mbtjac003@myuct.ac.za

Supervisors

Tommie Meyer tmeyer@cair.org.za

Jesse Heyninck jesse.heyninck@ou.nl