COMP4418

Knowledge Representation and Reasoning

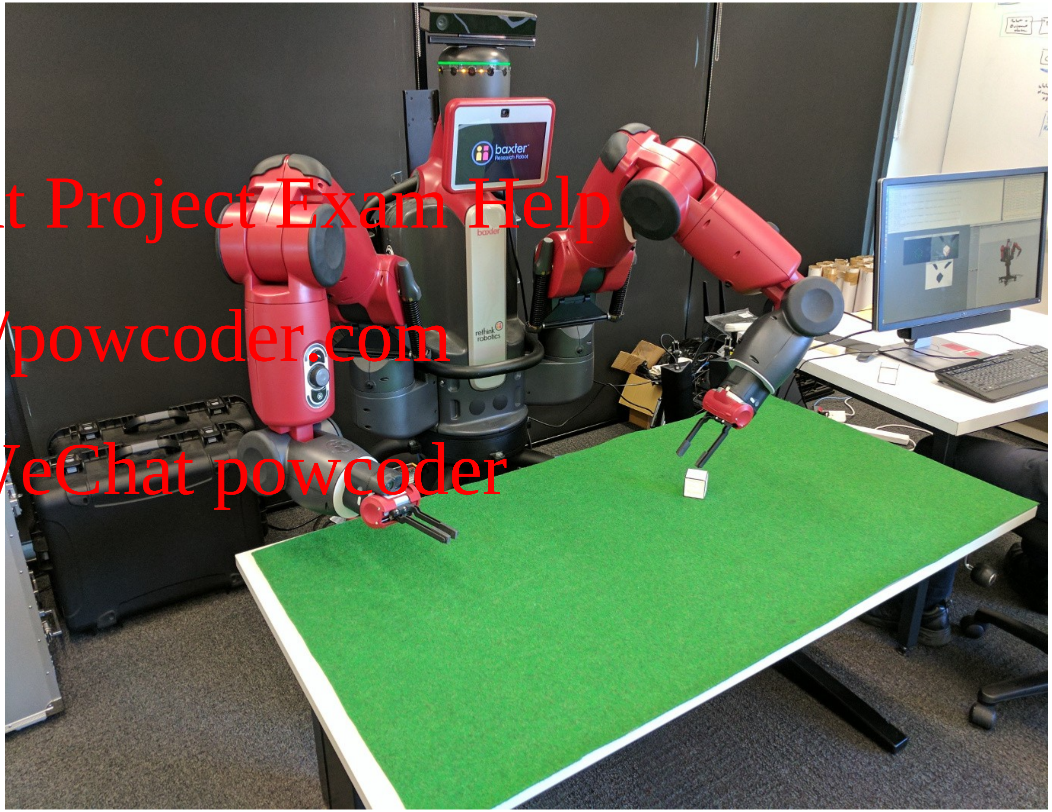
Week 3 – Practical Reasoning

David Rajaratnam

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

Practical Reasoning - My Interests

- Cognitive Robotics.
- Connect high level cognition with low-level sensing/actuators.
- Logical reasoning to make robot behave intelligently.
- Baxter Blocksworld video...

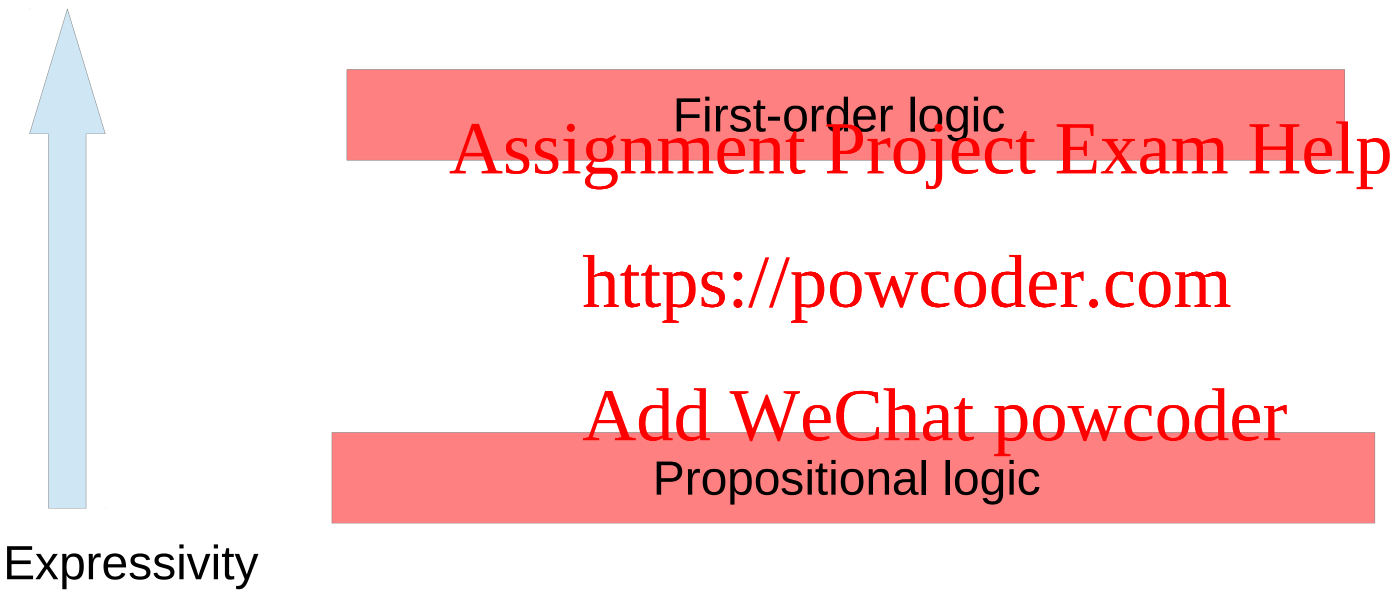


Recap of Weeks 1 & 2

- Week 1: Propositional logic
  - Simple propositions: "Socrates is bald"
  - Semantics: meaning decided using truth tables
  - Syntax: provability decided using inference rules - resolution for CNF
  - But… limited expressivity
- Week 2: First-order logic
  - Able to capture properties of objects and relationships between objects
  - Semantics: meaning decided using interpretations
  - Syntax: provability using inference rules - resolution + unification for CNF
  - highly expressive but… undecidable.

A Brief Overview of KRR Formalisms

Many Formalisms in KRR



Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

*actually semi-decidable, but distinction is not important for this course.

Propositional logic – Satisfiablity is NP-complete

Expressivity

Computational Complexity

Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiablity is NP-complete

Expressivity

Computational Complexity

Many important problems:

- Scheduling
- Timetabling
- Vehicle routing

Many Formalisms in KRR

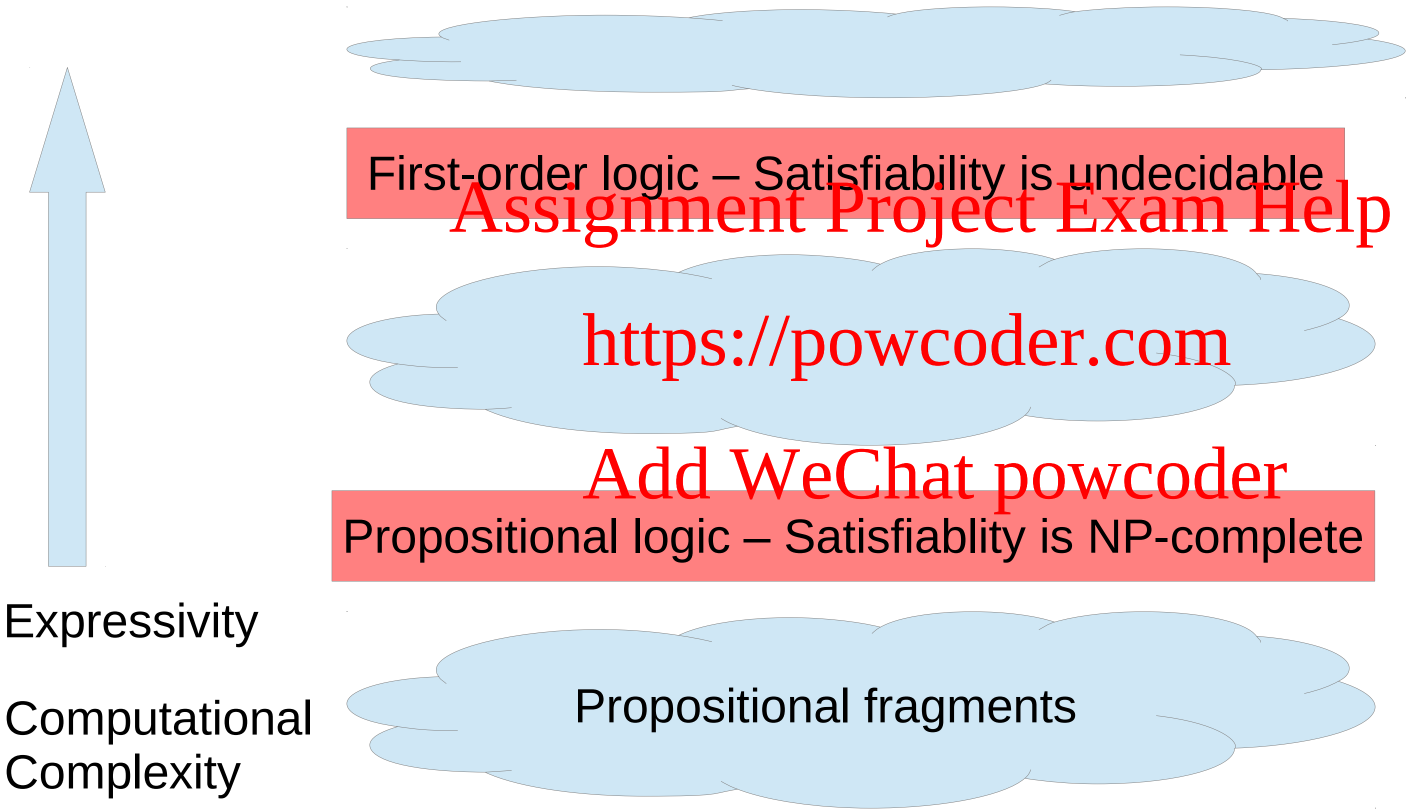
First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiablity is NP-complete

Expressivity

Computational Complexity

Many Formalisms in KRR



When speed is important:
- Databases

Many Formalisms in KRR

Higher-order logics – some interest

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiablity is NP-complete

Propositional fragments

Expressivity

Computational Complexity

When speed is important:

- Databases

Many Formalisms in KRR

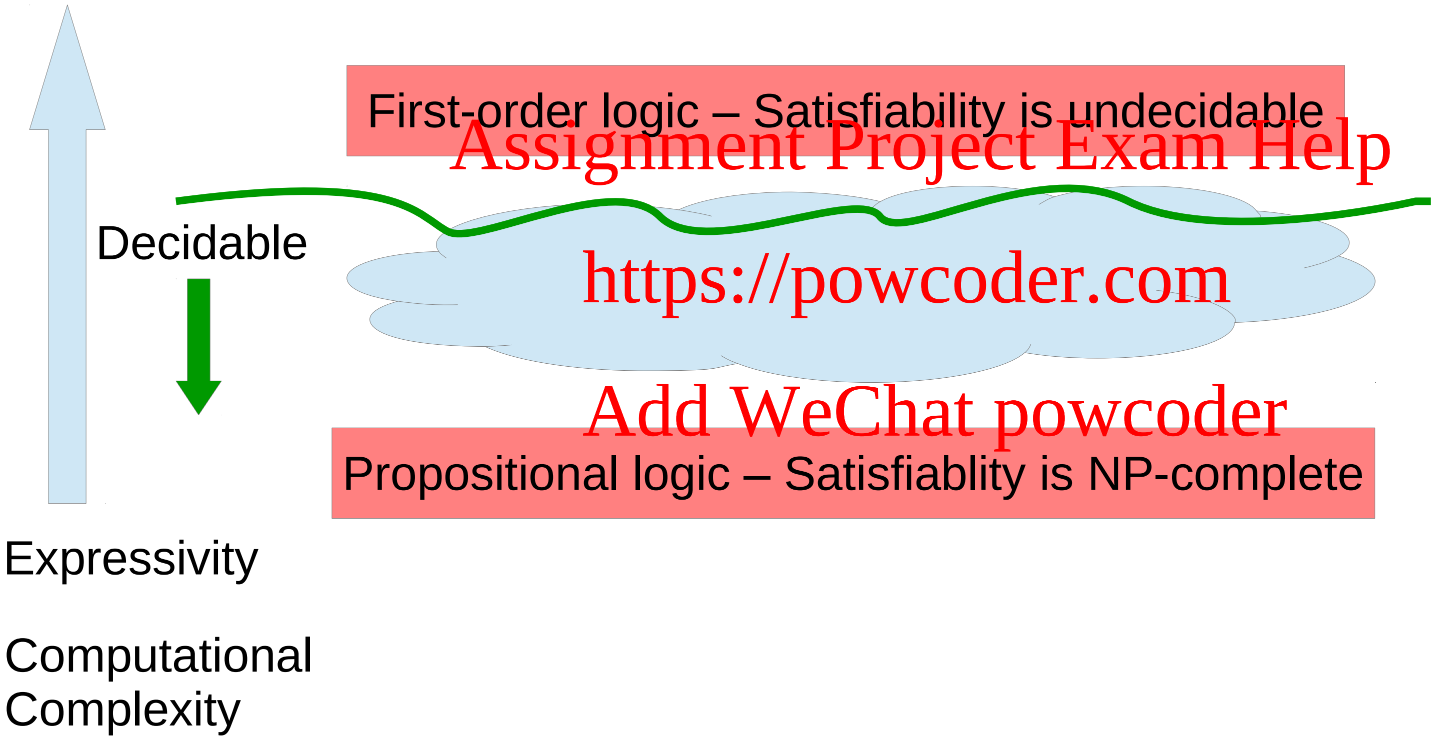
First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiablity is NP-complete

Expressivity

Computational Complexity

Many Formalisms in KRR



Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

Decidable

Description logics, modal logics,
Answer Set Programming

Propositional logic – Satisfiablity is NP-complete

Expressivity

Computational
Complexity

Many interesting problems:
- Planning
- Reasoning about knowledge

Horn Clauses

Clause Recap

From weeks 1 & 2:

- Every formula can be converted to Conjunctive Normal Form (CNF)
- Any CNF can be viewed as a set of clauses
- Entailment checking with resolution is complete (proof by refutation)
- So using sets of clauses provides:
  - Intuitive language for expressing knowledge

$$\neg a, a \vee b \quad \text{vs} \quad \neg(a \vee (\neg a \wedge \neg b))$$

  - Simple proof procedure that can be implemented

Reading Clauses as Implication

Clauses can be intuitively interpreted in two ways:

- As disjunction: $\text{rain} \lor \text{sleet}$
- As implication: $\neg\text{child} \lor \neg\text{male} \lor \text{boy}$
  - for syntactic convenience: $\text{child} \land \text{male} \rightarrow \text{boy}$
  - so can be read as: if "child" and "male" then "boy"

To understand why this makes sense go back to the truth tables:

A	B	$\neg A$	$\neg A \lor B$	$A \rightarrow B$
True	True	False	True	True
True	False	False	False	False
False	True	True	True	True
False	False	True	True	True

Horn Clauses

- ***Horn clause*** is a clause with at most one positive literal
- A ***positive*** (or ***definite) clause*** has exactly one positive literal

  $\neg\text{child} \vee \neg\text{male} \vee \text{boy}$

- A ***negative clause*** (or ***constraint***) has no positive literals

  $\neg\text{open} \vee \neg\text{closed}$

  - Note, since $\neg\text{open} \vee \neg\text{closed} \equiv \neg\text{open} \vee \neg\text{close} \vee \text{False}$

  - Hence $\text{open} \wedge \text{closed} \rightarrow \text{False}$ ($\text{open} \wedge \text{closed} \rightarrow \bot$ or $\text{open} \wedge \text{closed} \rightarrow$)
  - Also know as a **goal** when performing refutation proof
- A ***fact*** is a definite clause with no negative literals (i.e., a single positive literal):

  raining

Resolution with Horn Clauses 1

Two options:

Neg Pos → Neg

Pos Pos → Pos

Examples:

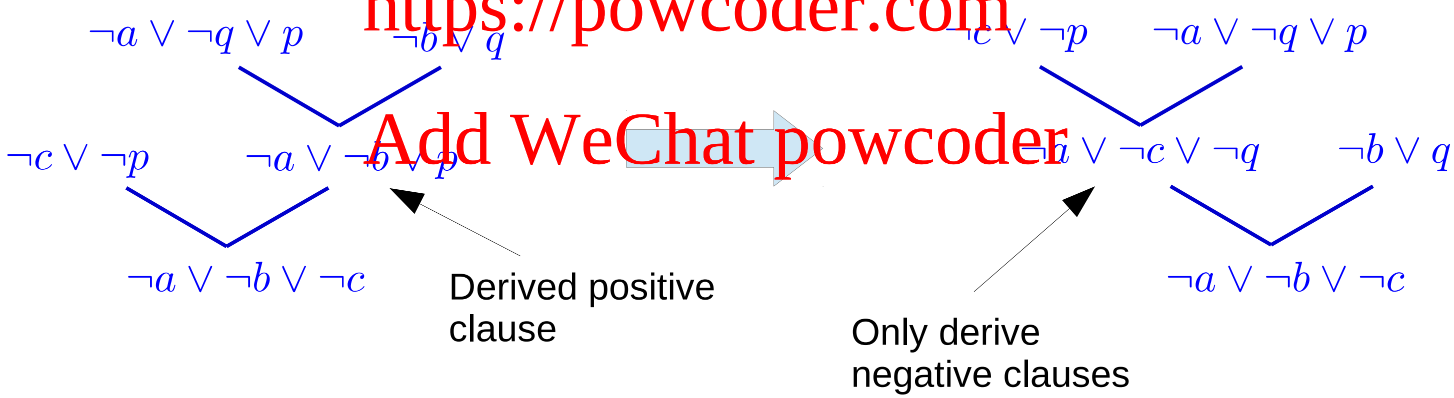
$\neg a \vee \neg b$ $\quad a \vee \neg c$ → $\neg b \vee \neg c$

$\neg a \vee b$ $\quad a \vee \neg c$ → $\neg c \vee b$

Resolution with Horn Clauses 2

It is possible to rearrange derivations (of negative clauses) so that all new derived clauses are negative clauses:

Given clauses: $\neg a \vee \neg q \vee p \qquad \neg b \vee q \qquad \neg c \vee \neg p$



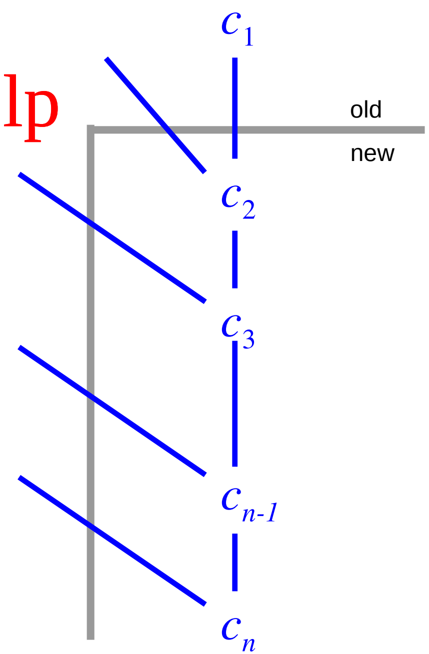
SLD Resolution

Can change derivations such that each derived clause is a resolvent of the previous derived (negative) one and some positive clause in the original set of clauses

- Since each derived clause is negative, one parent must be positive (and so from original set) and one negative.

- Continue working backwards until both parents of derived clause are from the original set of clauses

- Eliminate all other clauses not on direct path



SLD Example

To show that $\mathrm{KB} \models \mathrm{Girl}$ derive a contradiction from $\mathrm{KB} \cup \{\neg\mathrm{Girl}\}$

$$\boxed{\begin{array}{ll} \mathrm{KB} = \{ & \mathrm{FirstGrade}, \\ & \mathrm{FirstGrade} \rightarrow \mathrm{Child}, \\ & \mathrm{Child} \wedge \mathrm{Male} \rightarrow \mathrm{Boy}, \\ & \mathrm{Kindergarten} \rightarrow \mathrm{Child}, \\ & \mathrm{Child} \wedge \mathrm{Female} \rightarrow \mathrm{Girl}, \\ & \mathrm{Female} \quad \} \end{array}}$$

$\neg\mathrm{Child} \vee \neg\mathrm{Female} \vee \mathrm{Girl}$ $\neg\mathrm{Girl}$

Female $\neg\mathrm{Child} \vee \neg\mathrm{Female}$

$\neg\mathrm{FirstGrade} \vee \mathrm{Child}$ $\neg\mathrm{Child}$

$\mathrm{FirstGrade}$ $\neg\mathrm{FirstGrade}$

$\perp$

Note: Horn clauses capture a very intuitive way that we express knowledge.

SLD Resolution (formal)

An <u>SLD-derivation</u> of a clause c from a set of clauses S is a sequence of clauses $c_1, c_2, \ldots c_n$ such that $c_n = c$, and

1. $c_1 \in S$
2. c_{i+1} is a resolvent of c_i and a clause in S

Written as: $S \vdash^{\mathrm{SLD}} c$

SLD mean S(elected) literals
L(inear) form
D(efinite) clauses

In General SLD is incomplete

SLD resolution is not complete for general clauses.

An example: $S = \{\ p \vee q,\ p \vee \neg q,\ \neg p \vee q,\ \neg p \vee \neg q\ \}$

So S is unsatisfiable, that is: $\mathrm{S} \vdash \perp$, but $\mathrm{S} \not\vdash^{\mathrm{SLD}} \perp$

SLD cannot derive the contradition because it needs to eventually perform resolution on the intermediate clauses p and $\neg p$ (or q and $\neg q$)

Completeness of SLD

- But SLD resolution IS complete for Horn clauses.

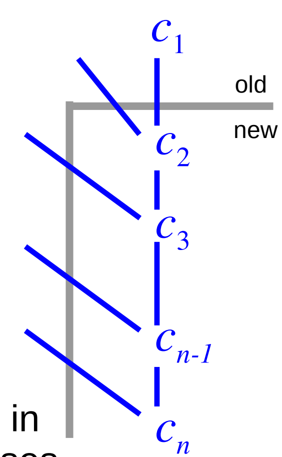
Theorem: If H is a set of Horn clauses then $\mathrm{H} \vdash \bot$ iff $\mathrm{H} \vdash^{\mathrm{SLD}} \bot$

- This is a good result as searching for appropriate clauses to resolve on is simpler for SLD resolution.

- Satisfiability for propositional Horn clauses is P-complete.

- Nothing is for free: **loss of expressivity.**
- Cannot express simple (positive) disjunctions.

$\text{open} \vee \text{closed}$



n is polynomial in number of clauses

Back to the KRR Overview

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiablity is NP-complete

Expressivity

Computational Complexity

Propositional fragments – **Horn clauses**

Back to the KRR Overview

First-order logic – Satisfiability is undecidable

Decidable

Propositional logic – Satisfiablity is NP-complete

Expressivity

Tractable

Propositional fragments – **Horn clauses**

Computational Complexity

First-Order (FO) Clauses

Week 2 recap:

- Conversion to FO CNF is same as propositional case except:
  - Standardise variable names
  - Skolemise (getting rid of existential quantifers)
  - Drop universal quantifiers
- FO resolution is same as propositional case except:
  - Find substitutions to unify the two clauses

First-Order (FO) Horn Clauses

- Same as propositional case except in a FO language
- SLD-resolution also same; with addition of unification
- Completeness of FO Horn also holds

Theorem: If H is a set of Horn clauses then $\mathrm{H} \vdash \bot$ iff $\mathrm{H} \vdash^{\mathrm{SLD}} \bot$

- But…

First-Order (FO) Horn Clauses

- FO Horn is undecidable. With Horn SLD resolution we can still generate an infinite sequence of resolvents.

KB:

$\mathrm{LessThan}(\mathrm{succ}(x), y) \rightarrow \mathrm{LessThan}(x, y)$

Query:

$\mathrm{LessThan}(0, 0)$

Should fail since $KB \not\models \mathrm{LessThan}(0, 0)$

$\neg\mathrm{LT}(0, 0)$

$\neg\mathrm{LT}(\mathrm{s}(x), y) \vee \mathrm{LT}(x, y)$

$x/0, y/0$

$\neg\mathrm{LT}(1, 0)$

$\neg\mathrm{LT}(\mathrm{s}(x), y) \vee \mathrm{LT}(x, y)$

$x/1, y/0$

$\neg\mathrm{LT}(2, 0)$

$\neg\mathrm{LT}(\mathrm{s}(x), y) \vee \mathrm{LT}(x, y)$

$x/2, y/0$

$\neg\mathrm{LT}(3, 0)$

Basis for Logic Programming

- Since FO Horn is undecidable it is also very expressive
- FO Horn and SLD resolution form the basis for Prolog
  - A general purpose programming language based on logic
  - Provides an intuitive language for expressing knowledge
  - Prolog is Turing-complete
  - Prolog is a form of declarative programming – you specify what the program should do not how it should do it

Prolog

….go to Prolog slides

Concluding Remarks

Conclusion

- Scoped out the KRR landscape and relationship between formalisms
- Looked at propositional and first-order Horn clauses and SLD resolution

  - Empasised distinction between *Semantics* vs *Syntax*

    - *Entailment* (meaning) $S \models \alpha$
    - *Inference* (symbol manipulation) $S \vdash \alpha$

- Looked at Prolog
  - Turing complete: general purpose programming language
  - Declarative programming allows for compact representations

Coming Weeks

- Prolog's expressivity comes with a cost
  - Efficiency issues and undecidability
  - Operational behaviour violates logical semantics; cut (!) operator, ordering of clauses.
- In coming weeks will look at more specialised logics that take a different approach to balance expressibility-computability-efficiency