

# COMP4418

## Knowledge Representation and Reasoning

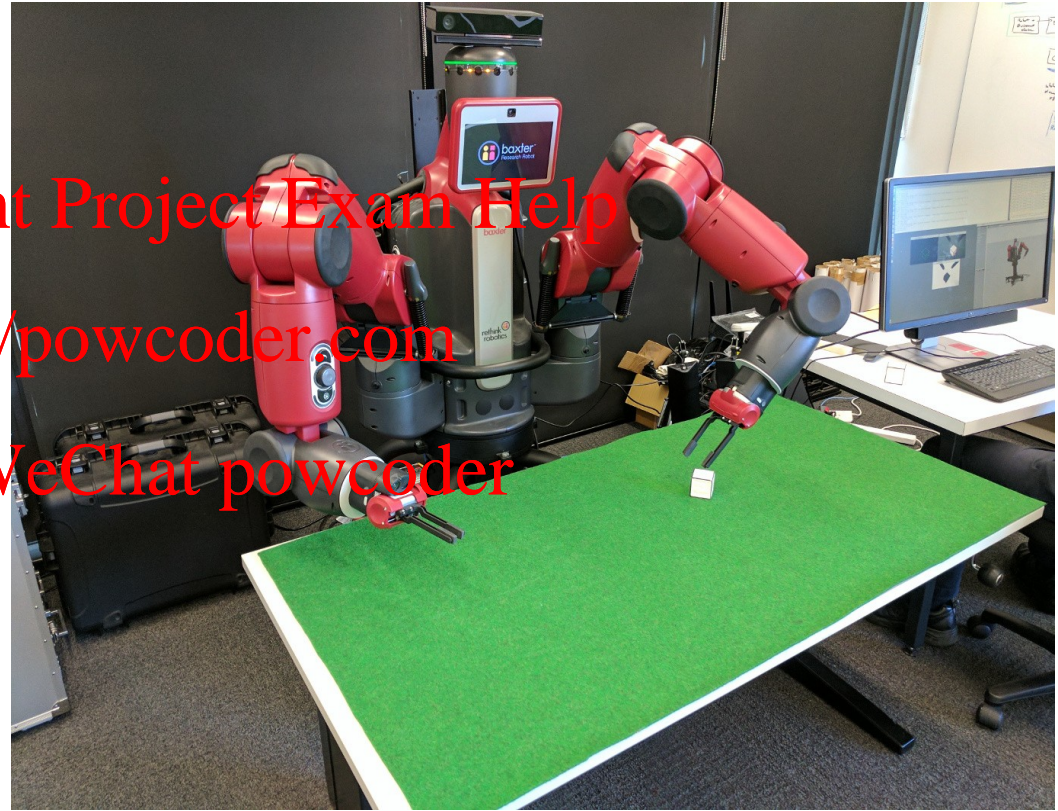


Assignment Project Exam Help  
Week 3 – Practical Reasoning  
<https://powcoder.com>  
David Rajaratnam

Add WeChat powcoder

# 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...



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# 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.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Assignment Project Exam Help

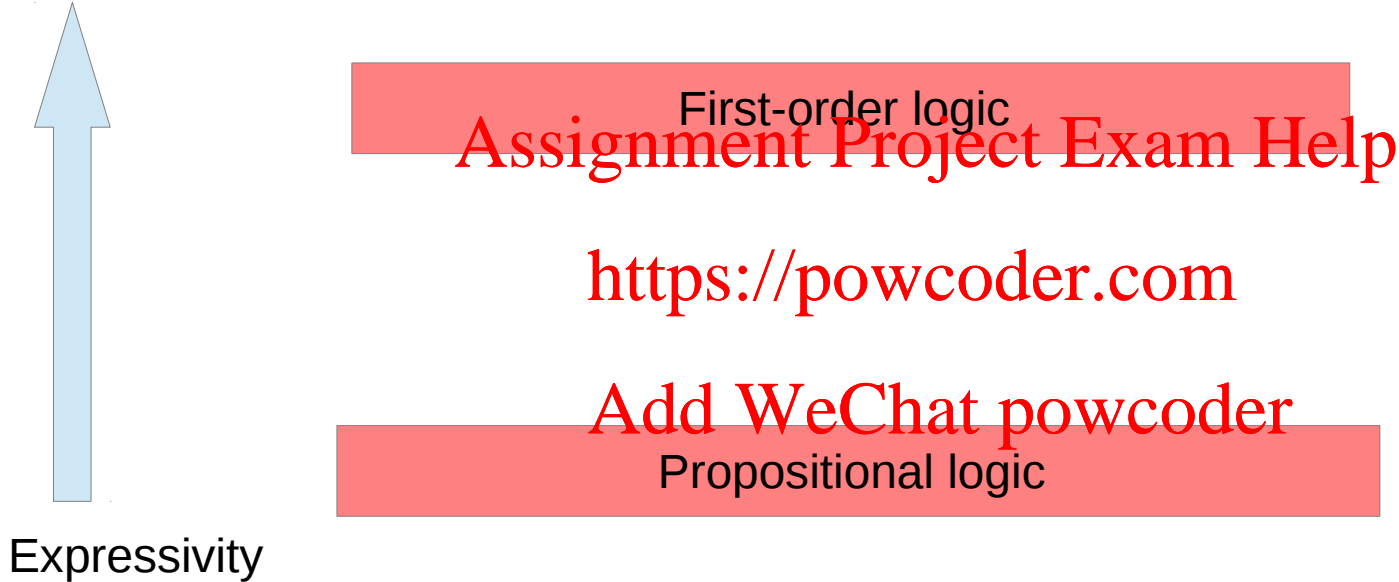
# A Brief Overview of KRR

<https://powcoder.com>

## Formalisms

Add WeChat powcoder

# 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.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete

Expressivity

Computational  
Complexity



UNSW  
THE UNIVERSITY OF NEW SOUTH WALES

# Many Formalisms in KRR



First-order logic – Satisfiability is undecidable

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete

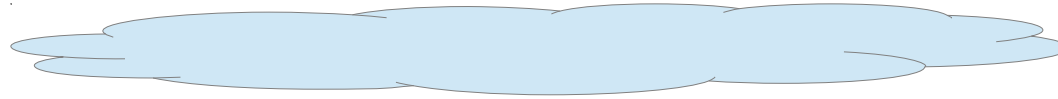
Expressivity

Computational  
Complexity

Many important problems:

- Scheduling
- Timetabling
- Vehicle routing

# Many Formalisms in KRR



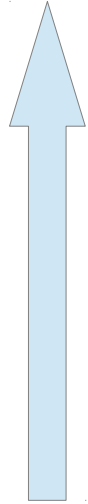
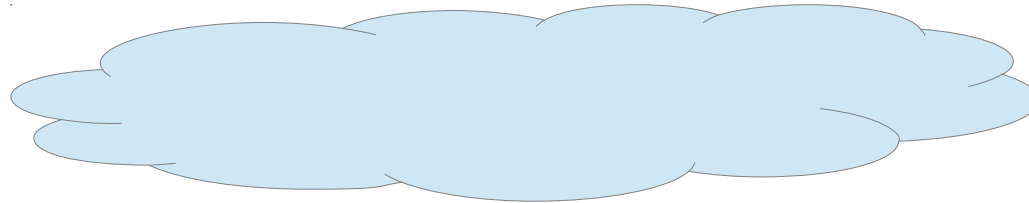
First-order logic – Satisfiability is undecidable

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete



Expressivity

Computational  
Complexity



# Many Formalisms in KRR

↑  
Expressivity  
Computational  
Complexity

First-order logic – Satisfiability is undecidable

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete

Propositional fragments

When speed is important:  
• Databases

# Many Formalisms in KRR

↑  
Expressivity  
Computational  
Complexity

Higher-order logics – some interest

First-order logic – Satisfiability is undecidable

Assignment Project Exam Help

<https://powcoder.com>

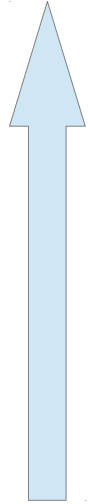
Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete

Propositional fragments

When speed is important:  
• Databases

# Many Formalisms in KRR



First-order logic – Satisfiability is undecidable

Assignment Project Exam Help

<https://powcoder.com>

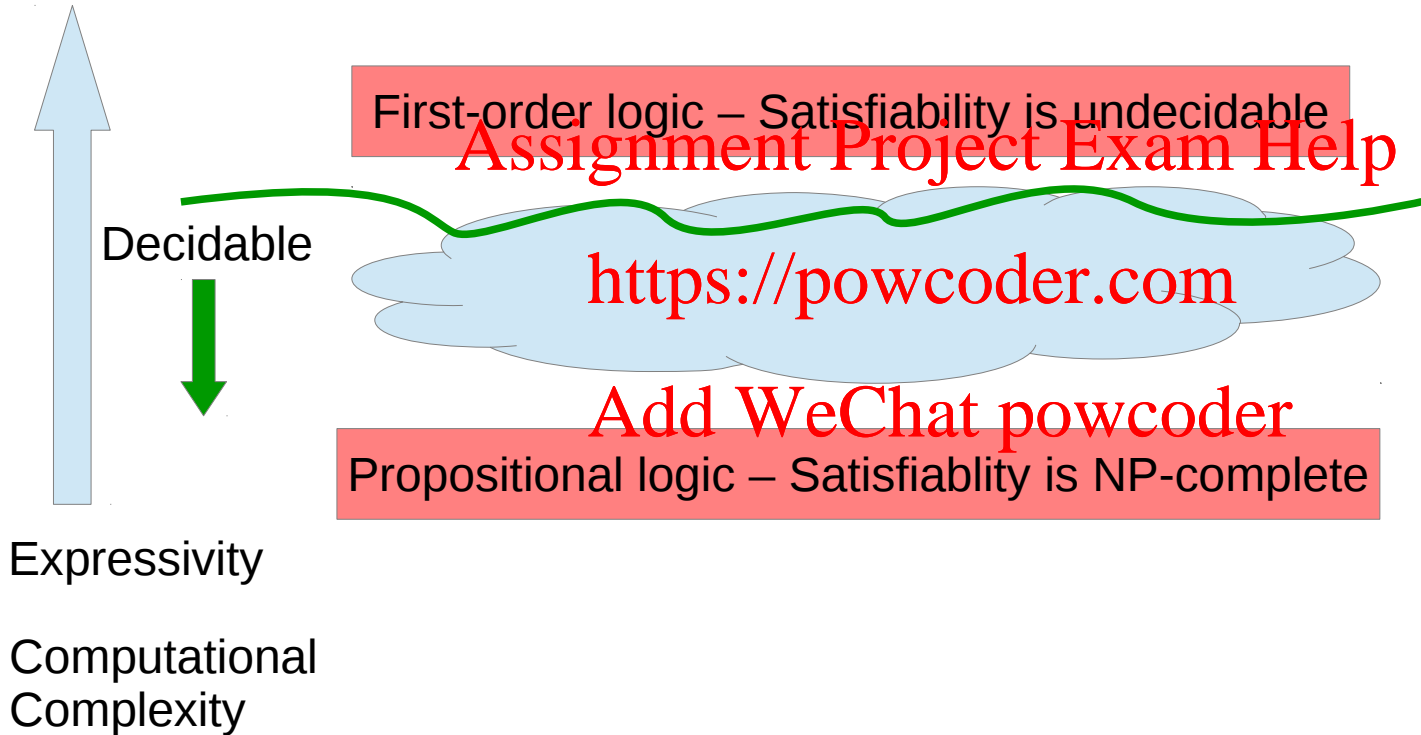
Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete

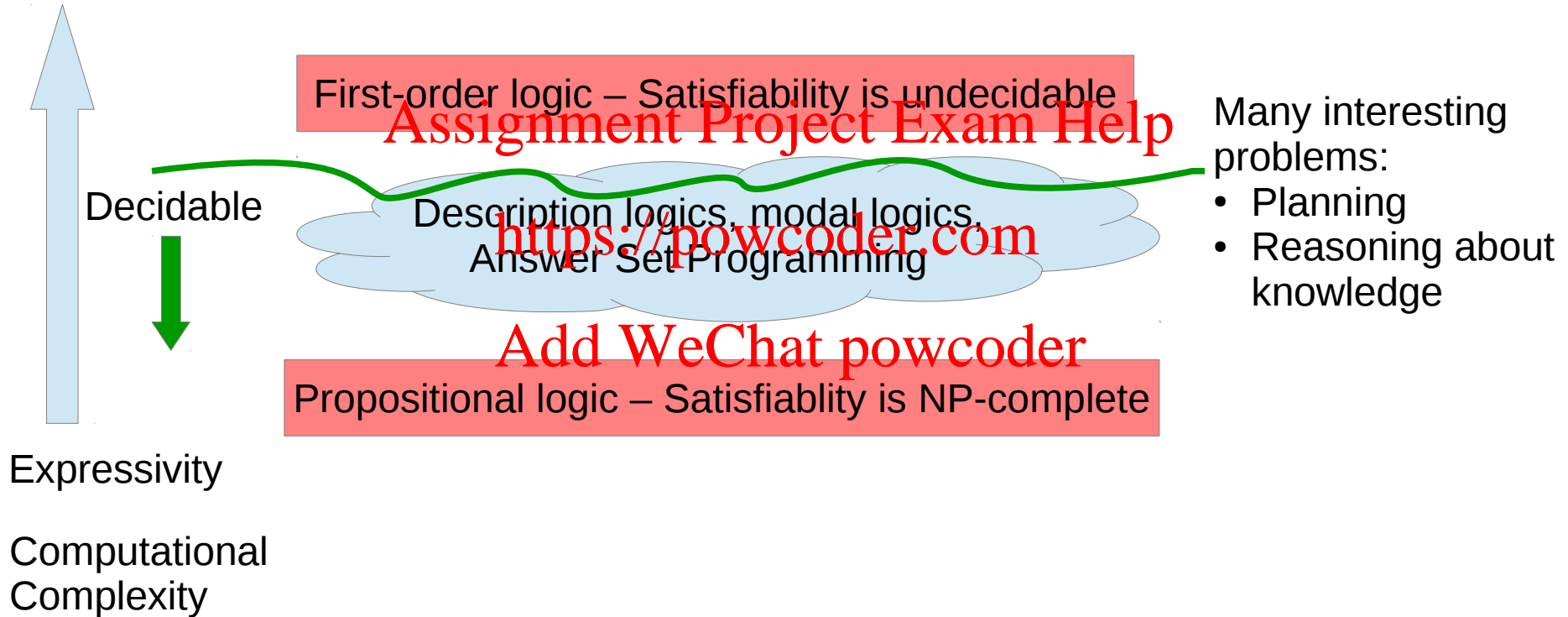
Expressivity

Computational  
Complexity

# Many Formalisms in KRR



# Many Formalisms in KRR



Assignment Project Exam Help  
**Horn Clauses**  
<https://powcoder.com>

Add WeChat powcoder

# 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
- $\neg a, a \vee b$     vs     $\neg(a \vee (\neg a \wedge \neg b))$
- Simple proof procedure that can be implemented

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Reading Clauses as Implication

Clauses can be intuitively interpreted in two ways:

- As disjunction:  $\text{rain} \vee \text{sleet}$
- As implication:  $\text{child} \vee \text{male} \vee \text{boy}$ 
  - for syntactic convenience:  $\text{child} \wedge \text{male} \rightarrow \text{boy}$
  - 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	$\neg A$	$\neg A \vee B$	$A \rightarrow B$
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

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

Assignment Project Exam Help

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

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

<https://powcoder.com>

– Note, since  $\neg \text{open} \vee \neg \text{closed} \equiv \neg (\text{open} \wedge \text{closed}) \vee \text{False}$

Add WeChat powcoder

– Hence  $\text{open} \wedge \text{closed} \rightarrow \text{False}$  ( $\text{open} \wedge \text{closed} \rightarrow \perp$  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):

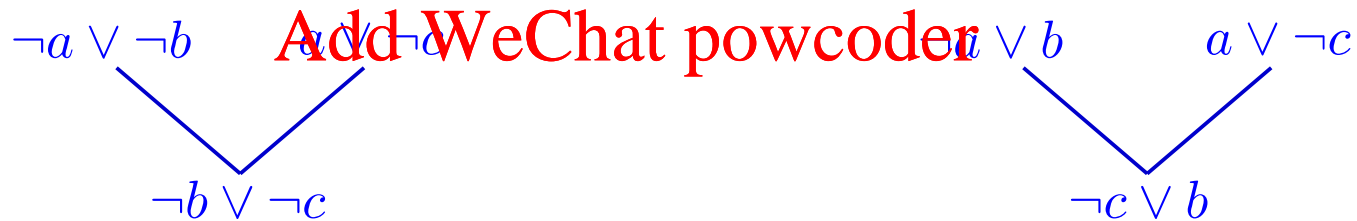
$\text{raining}$

# Resolution with Horn Clauses 1

Two options:



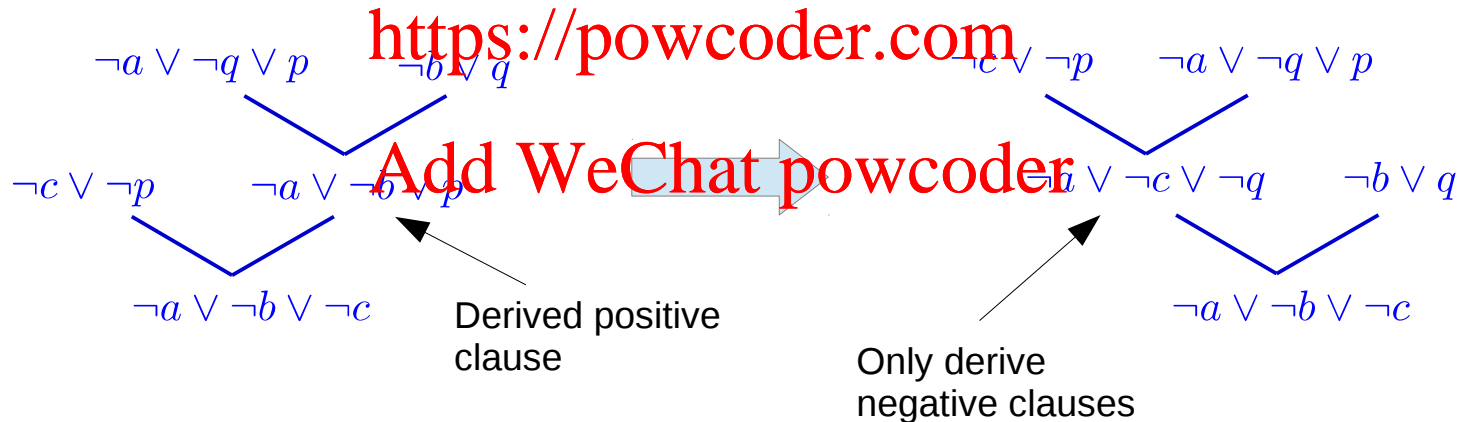
Examples:



# Resolution with Horn Clauses 2

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

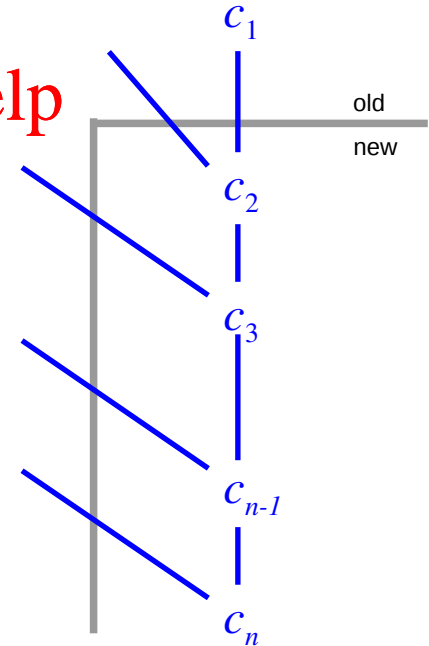
Given clauses: **Assignment Project Exam Help**



# 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



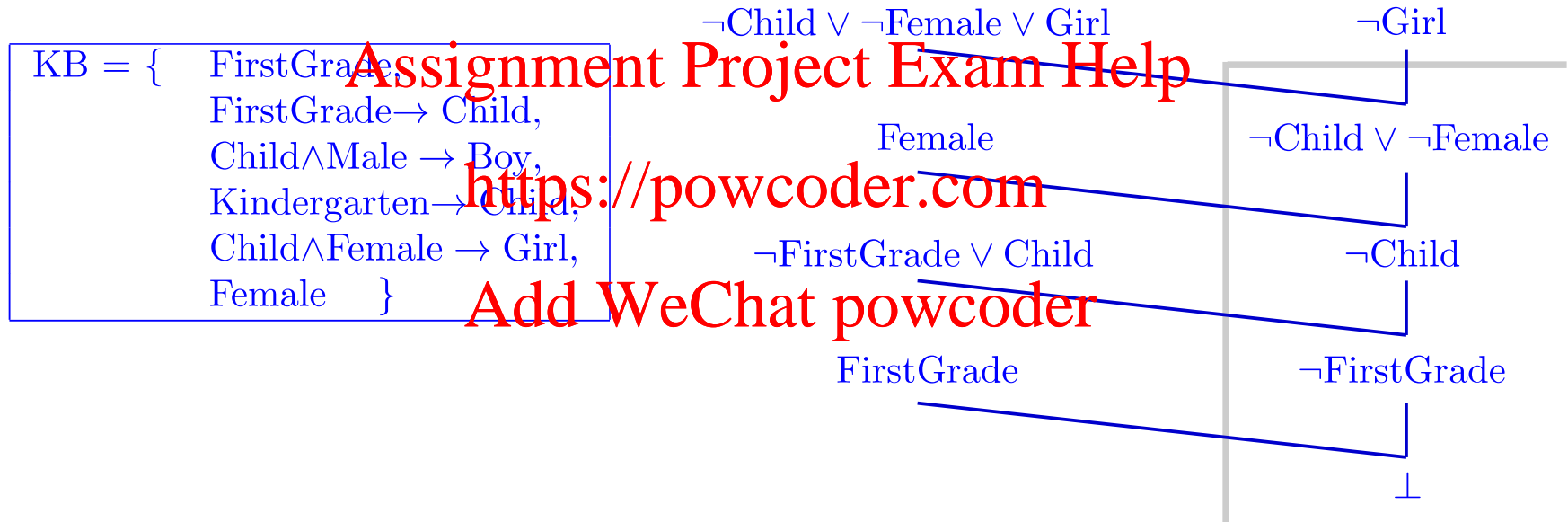
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# SLD Example

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



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

# SLD Resolution (formal)

An SLD-derivation of a clause  $c$  from a set of clauses  $S$  is a sequence of clauses  $c_1, c_2, \dots, c_n$  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  $S$

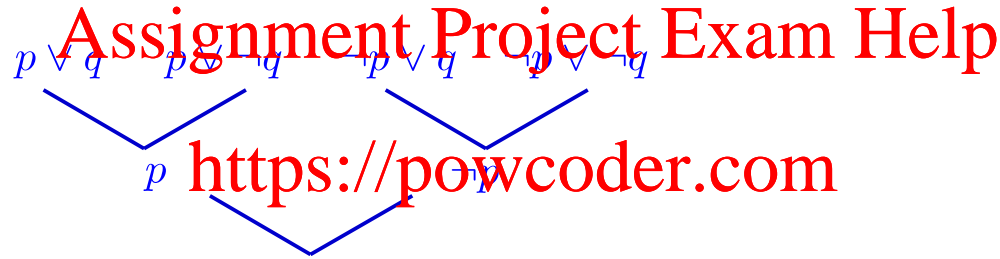
Written as:  $S \vdash^{\text{SLD}} c$

SLD means S(elect)ed L(inear) form D(efinite) clauses

# In General SLD is incomplete

SLD resolution is not complete for general clauses.

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



Add WeChat powcoder

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

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

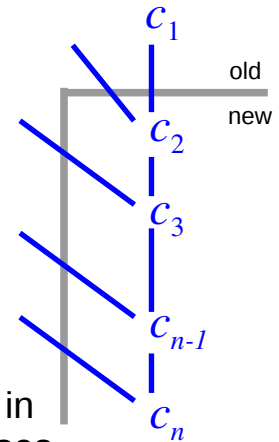
# Completeness of SLD

- But SLD resolution IS complete for Horn clauses.

**Theorem:** If  $H$  is a set of Horn clauses then  $H \models \perp$  iff  $H \vdash^{\text{SLD}} \perp$

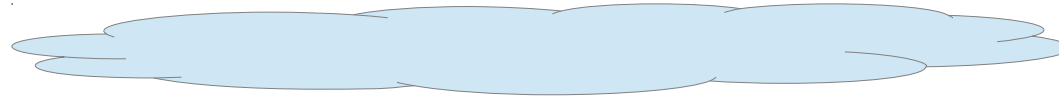
- This is a good result as searching for appropriate clauses to resolve on is simpler for SLD resolution.
- Satisfiability for propositional Horn clauses is P-complete.
- Nothing is for free: **loss of expressivity**.
- Cannot express simple (positive) disjunctions.

open  $\vee$  closed





# Back to the KRR Overview



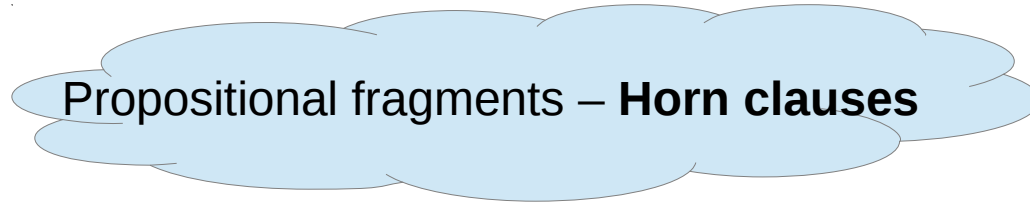
First-order logic – Satisfiability is undecidable

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Propositional logic – Satisfiability is NP-complete

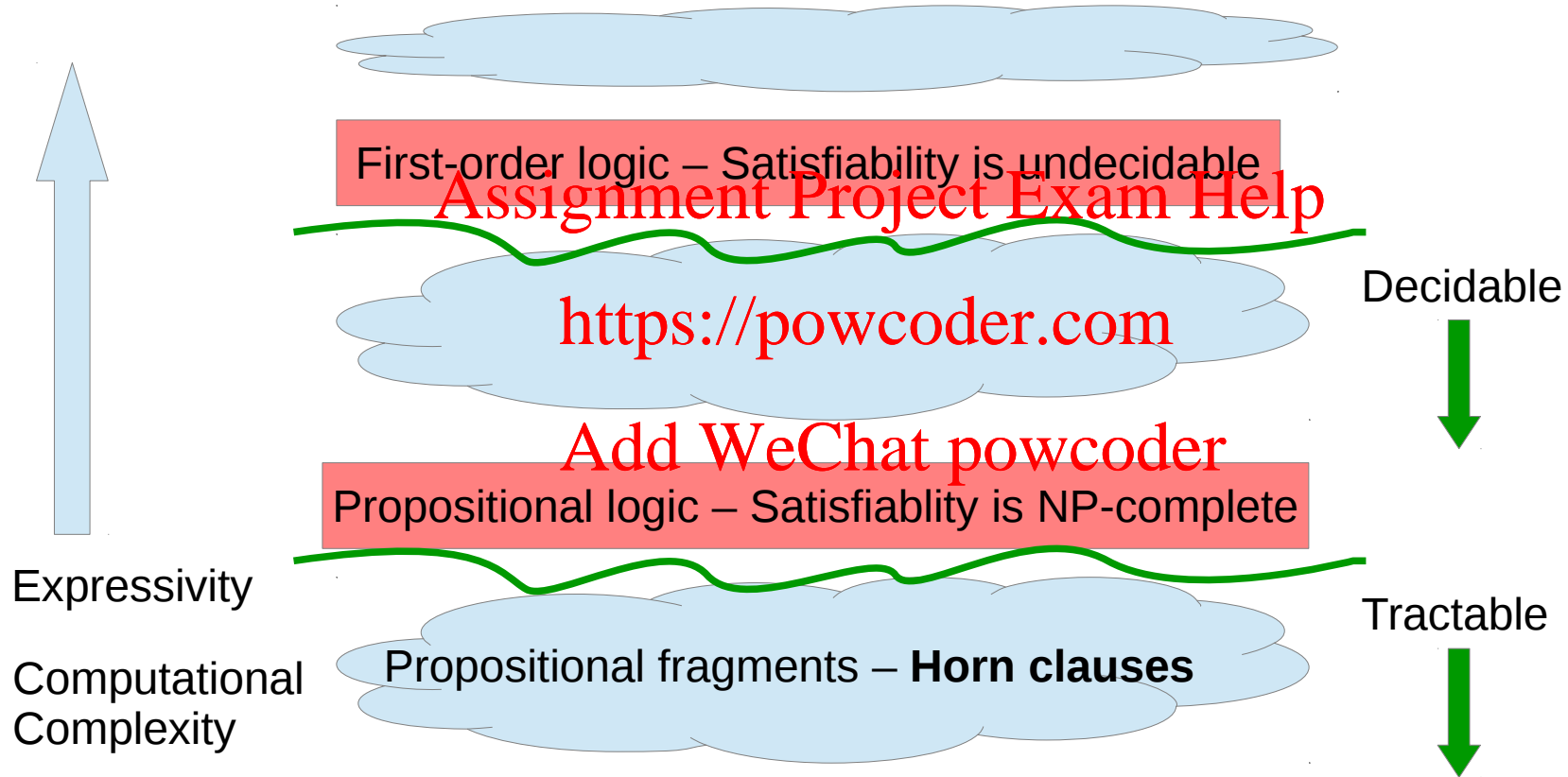


↑

Expressivity

Computational Complexity

# Back to the KRR Overview



# 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 quantifiers)
  - Drop universal quantifiers
- FO resolution is same as propositional case except:
  - Find substitutions to unify the two clauses

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# 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  $H \vdash \perp$  iff  $H \vdash^{\text{SLD}} \perp$

- But...

# First-Order (FO) Horn Clauses

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

KB:

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

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

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

Query:

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

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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

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

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

$x/0, y/0$

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

$x/1, y/0$

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

$x/2, y/0$

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

# 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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Assignment Project Exam Help

Prolog

<https://powcoder.com>

....go to Prolog slides

Add WeChat powcoder

Assignment Project Exam Help

# Concluding Remarks

<https://powcoder.com>

Add WeChat powcoder



# 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

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Coming Weeks

- Prolog's expressivity comes with a cost
  - Efficiency issues and non-terminability
  - 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