COMP4418

Knowledge Representation and Reasoning

Practical Reasoning - My Interests

- Cognitive Robotics.
- Connect high level cognition with low-level sensing/actuators
- Logical reasoning 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 de for CNF
  - But... limited express
- 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 of KRR

Many Formalisms in KRR

First-order logic

Propositional lo

Expressivity

Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

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

Propositional logic – Satisfiabli lete

Expressivity

Computational Complexity

Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiabli lete

Expressivity

Computational Complexity

Many important problems:

- Scheduling
- Timetabling
- Vehicle routing

Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiabli lete

Expressivity

Computational Complexity

Many Formalisms in KRR

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiabli lete

Propositional fragments

Expressivity

Computational Complexity

When speed is important:

- Databases

Many Formalisms in KRR

Higher-order logics – some interest

First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiabli lete

Propositional fragments

Expressivity

Computational Complexity

When speed is important:

- Databases

Many Formalisms in KRR

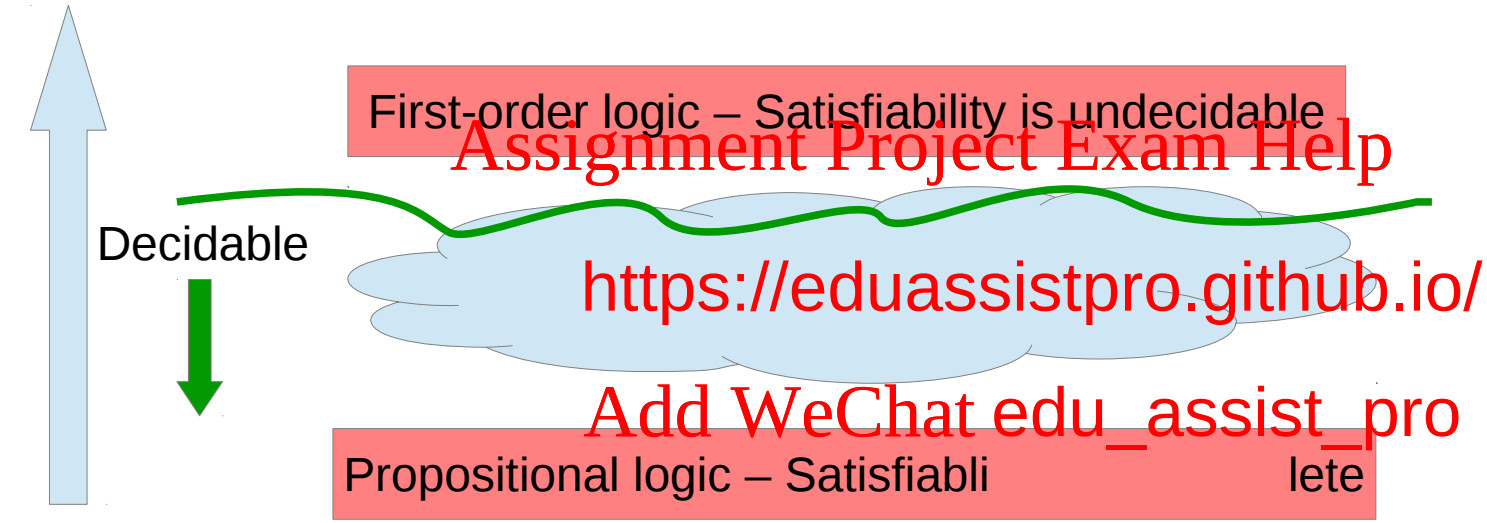
First-order logic – Satisfiability is undecidable

Propositional logic – Satisfiabli lete

Expressivity

Computational Complexity

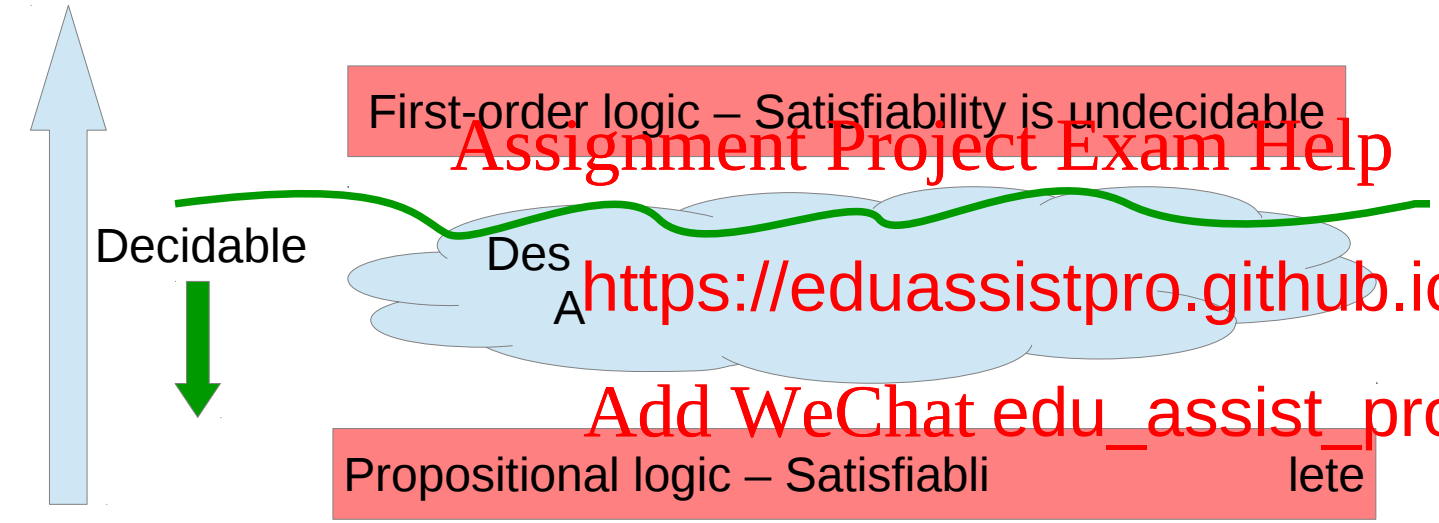
Many Formalisms in KRR



Expressivity

Computational Complexity

Many Formalisms in KRR



Many interesting problems:

- Planning
- Reasoning about knowledge

Expressivity

Computational Complexity

Clause Recap

From weeks 1 & 2:

- Every formula can be converted to Conjunctive Normal Form (CNF)
- Any CNF can be v
- Entailment check roof 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: $\text{rain} \vee \text{sleet}$
- As implication: $\neg\text{child} \vee \neg\text{male} \vee \text{boy}$
  - for syntactic co $\text{child} \wedge \text{male} \rightarrow \text{boy}$
  - so can be read hen "boy"

To understand why this makes sense go back s:

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

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

- A ***negative clause*** (o

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

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

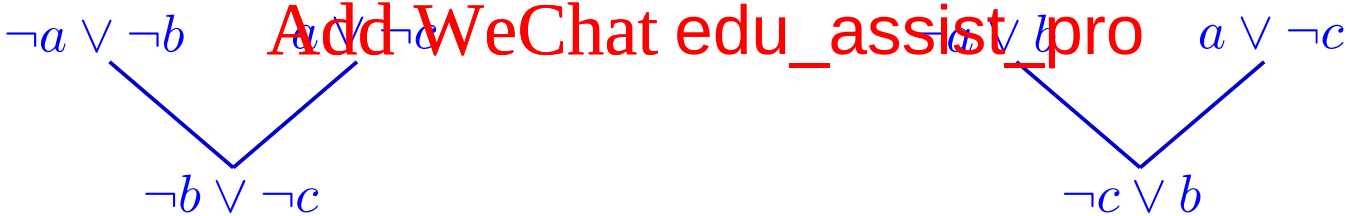
  - Hence $\text{open} \wedge \text{closed} \rightarrow \text{False}$ ($\text{open} \wedge \text{closed} \rightarrow \bot$ or $\text{open} \wedge \text{closed} \rightarrow$)
  - Also know as a **goal** when performing refutation proof
- A ***fact*** is a definite clause with no negative literals (i.e.,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

  raining

Resolution with Horn Clauses 1

Two options:

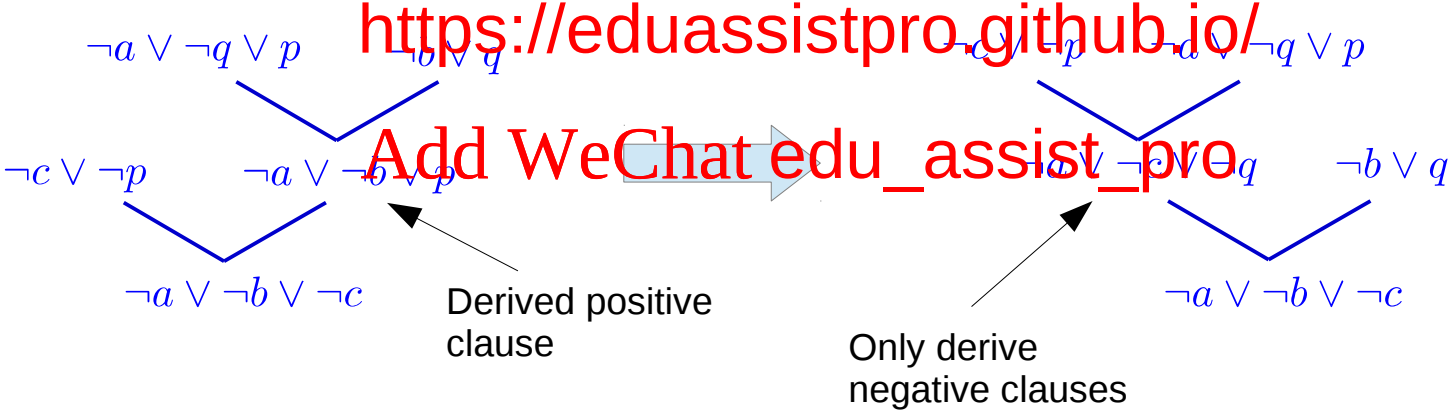
Examples:



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: $\neg a \vee \neg q \vee p$ $\neg b \vee q$ $\neg c \vee \neg p$



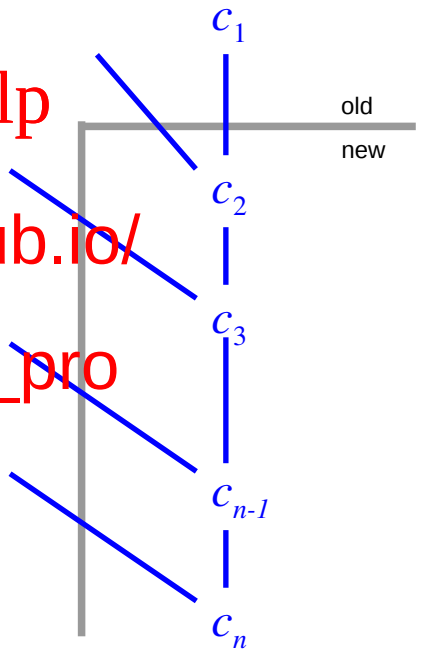
SLD Resolution

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

- Continue working ba ause 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 $\mathrm{KB} \cup \{\neg\mathrm{Girl}\}$

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

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

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

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

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

$\perp$

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 resolven

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

SLD mean

L(inear) form

D(efinite) clauses

In General SLD is incomplete

SLD resolution is not complete for general clauses.

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

$$\dfrac{\dfrac{p \vee q \qquad p \vee \neg q}{p} \qquad \dfrac{\neg p \vee q \qquad \neg p \vee \neg q}{\neg p}}{\bot}$$

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

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.

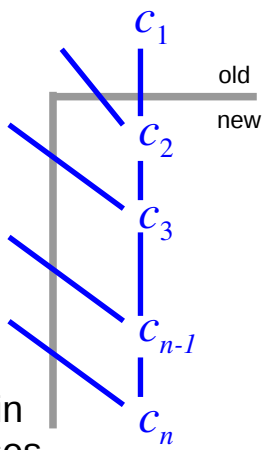
Theorem: If H is a set of Horn clauses then $\mathrm{H} \vdash \bot$ iff $\mathrm{H} \vdash^{\mathrm{SLD}} \bot$

- This is a good re clauses to resolve on is sim

- Satisfiability for propositional Horn plete.

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

open $\vee$ closed



n is polynomial in number of clauses

Back to the KRR Overview

First-order logic – Satisfiability is undecidable

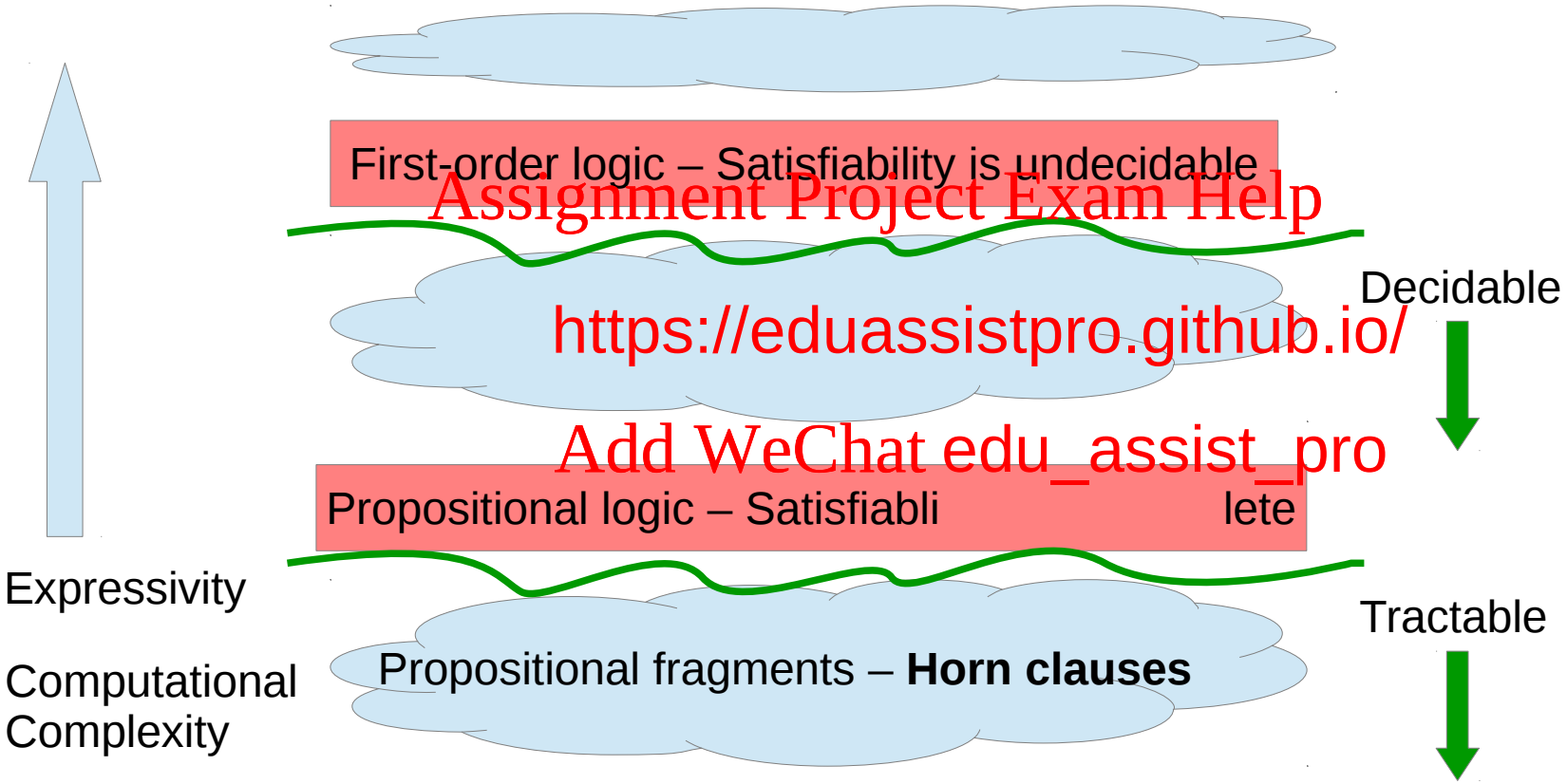
Propositional logic – Satisfiabli lete

Expressivity

Computational Complexity

Propositional fragments – **Horn clauses**

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 (g rs)
  - Drop universal quantifiers
- FO resolution is same as proposition
  - 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 $H \models \bot$ iff $H \vdash_{SLD} \bot$

- But…

First-Order (FO) Horn Clauses

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

KB:

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

Query:

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

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

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

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

$x/0, y/0$

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

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

$x/1, y/0$

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

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

$x/2, y/0$

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

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 pu sed on logic
  - Provides an 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

Con arks

Conclusion

- Scoped out the KRR landscape and relationship between formalisms
- Looked at propositional and first-order Horn clauses and SLD resolution
  - Empasised d *Syntax*
    - *Entailme* $S \models \alpha$
    - *Inference* (symbol manipula $S \vdash \alpha$
- 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 b tics; cut (!) operator, ordering of cl
- In coming weeks will look at more sa that take a different approach to balance expressibility-c ciency