CSE 579: Knowledge Representation & Reasoning

Module 3: Answer Set Programming

Ali Altunkaya
Spring 2023

Module 3 Highlights

We will study 3 different formal languages during the course:

1-) Propositional Logic (PL)

2-) First Order Logic (FOL)

3-) Answer Set Programming (ASP)

Ultimate goal: Encode natural language in a formal language so we can do … (reasoning)

ASP is more expressive than FOL: we can encode more complex sentences from natural language into the formal language.

Outline

Module 3 Highlights: 1. Introduction to ASP

Limitations of FOL: Transitive closure, defaults, exceptions, probabilities, vague knowledge, etc.

Introduction to Answer Set Programming (ASP):

– We'll start actual programming (coding) in this module.

– Answer Set Programming is a declarative programming.

– Answer Set = Stable model of the problem.

What is the difference between declarative programming & traditional programming?

- Traditional programming focus on solutions/algorithms (e.g. Dijkstra's, Kruskal's, MergeSort etc.) .
- Declarative programming focus on how to model the problem, not details about algorithms or how to solve it. Solve the problem via satisfiability. Find the model (solution) by minimizing the search space using DPLL algorithm.

Module 3: Summary of concepts

Formula

Program (or Rules)
A rule is a formula of the form $F \rightarrow G$
if there is no implication within F and G.
example: $p \wedge q \leftarrow \neg r \vee q$

Positive Program
A rule is a positive rule,
if there is no negation.
example: $p \wedge q \leftarrow r \vee q$

Definite Program
A rule is a definite rule,
if there is no negation and head is atom.
example: $p \leftarrow r \vee q$

Interpretations
The set of all possible solutions.

Model
A model is an interpretation that satisfies
a rule (or set of rules).
example: $p \wedge q \leftarrow T$
models: $\{p\}$, $\{q\}$, $\{p, q\}$

Minimal Model
A model is a minimal model, if no other model is its subset.
Our goal is to capture notion of Rationality Principle
(Believe nothing you are forced to believe.)
example: $p \wedge q \leftarrow T$
minimal models: $\{p\}$, $\{q\}$

Stable Model
Stable model of a definite program is the the minimal model of it.
Definite program has a unique stable model, since head is a single atom.

Stable model of a positive program is a minimal model of it.
Positive program can have many stable models, since head does not need to be a single atom.

Module 3 Highlights: Rules

Difference between Rule & Formula:

- We write F→G to denote F←G. (a small syntax difference with PL & FOL) Just follow the arrow for implication.
- Rules are restricted form of formulas,
- A rule is a formula of the form F←G, where F and G are implication free (⊥ T ¬ ∧ ∨ are allowed in F and G)
- If there is only formula F, it means F ← T.
- Head ← Body (left head, right body)

A **program** is a set of **rules**.

Representing Interpretations as Sets:

- We identify an interpretation with the set of atoms that are true in it. (different than PL and FOL)
- Ø (empty set) = means where all atoms are false.

Module 3 Highlights: 2. Minimal Models

Minimal Models: a model *I* of a formula F is minimal, if no other model of F is a subset of *I*.

Definite Rule: a rule $\overset{\text{head} \leftarrow \text{body}}{(F \leftarrow G)}$ is definite if

- its head is an atom
- its body (if exists) does not contain negation (¬).

Definite Program is a set of definite rules.

An algorithm to find a minimal model of a definite program:

- S = Ø // S is the set for minimal model
- Repeat for each rule (h ← B), until there is no change in S
  - if S satisfies Body, then S := S U {h}

Module 3 Highlights: 2. Stable Models

Proposition: A definite program has a unique minimal model.

- If it is not definite, there can be many minimal models.

Stable model of a definite program is its minimal model.

- Since there is a unique minimal model, there will be a unique stable model.

Rationality Principle: Each rule represent some beliefs. So a program is a set of beliefs. Minimal models help us to capture the notion of rationality, for a given program.

- Believe nothing you are forced to believe.
- If one believes in the body of a rule, the one must also believes in its head.

Module 3 Highlights: 2. Stable Models

Positive Programs: A rule/program is positive if it doesn't contain negation ¬.

Positive program is more general than Definite program, because:

- In definite program, the head must be an atom: $p \rightarrow q \vee r$
- In positive program, the head doesn't need to be atom: $p \vee q \rightarrow r$, therefore positive programs can have more than one minimal models.
- Every definite program is also a positive program, but not vice versa.

But we still keep these two definitions:

- A stable model of a definite program is the minimal model of it. (unique)
- A stable model of a positive program is a minimal model of it. (>=1)

Module 3: 3. Definite/Positive Programs in Clingo

Terms in Clingo: There are kinds of terms in Clingo. We use terms in the rules (or formulas).

- Integers, symbolic constants and variables
  - 1, 2, 3, a, b, c, X, Y, Z   (variables are capitalized,)
- $t_1 \circ t_2$ where t_1, t_2 are terms and $\circ$ is an arithmetic operation
  - 3*4, 3+4, 4/2, 3**2
- |t| where t is a term (absolute value)
  - |-3|

Terms that do not contain variable are called **ground terms**. The value of a ground term is defined recursively:

- If t is an integer or constant, then its value is the integer t itself.
- If $t = t_1 \circ t_2$ then its value is the integer, the result.
- If $t = |t_1|$ then its value is integer, the absolute value of t_1.

Module 3: 3. Definite/Positive Programs in Clingo

An **atom** in clingo program is a symbolic constant followed by a list of terms.

A **comparison** is a pair of terms separated by comparison operators: =, !=, <, >, <=, >=

Positive Clingo Program: are just set of rules. A rule is either:

- $H_1, H_2, \ldots H_m$ (m >= 1)
- $H_1, H_2, \ldots H_m$:– $B_1, B_2, \ldots B_n$ (m, n>= 0)

where $H_1, H_2, \ldots H_m, B_1, B_2, \ldots B_n$ are atoms and comparisons. (Header,Body)

Example rule: declarative sentence to explain “what is large?”

large(C) :– size(C,S1), size(uk, S2), S1>S2

Module 3: 3. Definite/Positive Programs in Clingo

Rule Instances: are obtained by replacing all variables with specific values.

- The values come from the set S of symbolic constants and the set Z of integers.

Propositional Image of a clingo program consists of its rule instances rewritten as propositional formulas.

Convert clingo programs into propositional formulas, and call it prop image.

To rewrite a ground rule as a formula, we eliminate variables:

- replace the symbol :– and all commas in the head/body by propositional connectives, see table
- replace each comparison $t_1<t_2$ in the head/body by T if it is True and by $\perp$ if it is false.
- replace empty head with $\perp$ (false), and replace empty body with T(true).

Module 3: 3. Definite/Positive Programs in Clingo

Expressing Definitions in Positive Programs:

- **Multiple Rules:** The definition of a predicate can consists of several rules:
  - parent (X, Y) :– father (X, Y)
  - parent (X, Y) :– mother (X, Y)

- **Recursive rules:** A predicate can be defined recursively, where the defined predicate occurs both in head&body of a rule. (not possible in FOL)
  - ancestor (X, Y) :– parent(X, Y)
  - ancestor (X, Z) :– ancestor(X, Y), ancestor(Y, Z)

Module 3: 4. Clingo Programs - Allowing Intervals

We already know the 3 kind of terms.

- Integers, symbolic constants and variables: 1, 2, 3, a, b, c, X, Y, Z
- $t_1 \circ t_2$ where t_1, t_2 are terms and ○ is an arithmetic operation 3*4, 3+4
- |t| where t is a term (absolute value) |-3|

Now we add another kind of term: intervals

- $t_1..t_2$ where t_1, t_2 are terms
  - 1..10

The value of an interval term is defined recursively too:

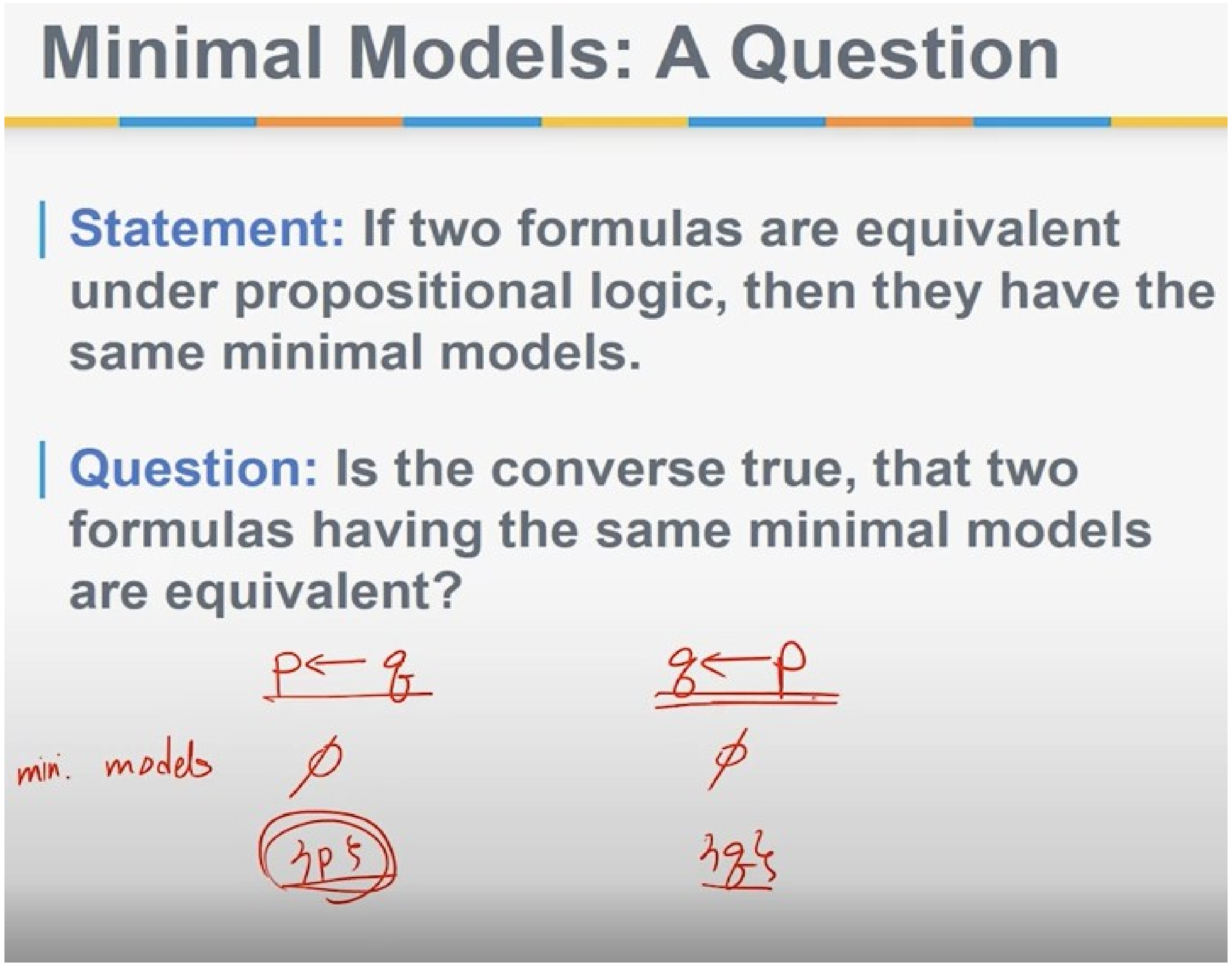
- 1..10 → {1, 2, 3 … 10}
- 1..10 + 1 → {2, 3, 4 … 11}
- (2..4) * (2..4) → {4, 6, 8, 9, 12, 16} since it is a set, there will no repeated values

Module 3: 4. Clingo Programs - Allowing Intervals

Propositional Image of a clingo program consists of its rule instances rewritten as propositional formulas.

- We need to replace each interval in the Head/Body with its prop formula
- But intervals behave different in the Head & Body of a rule. !!!

see slides …



Minimal Models: A Question

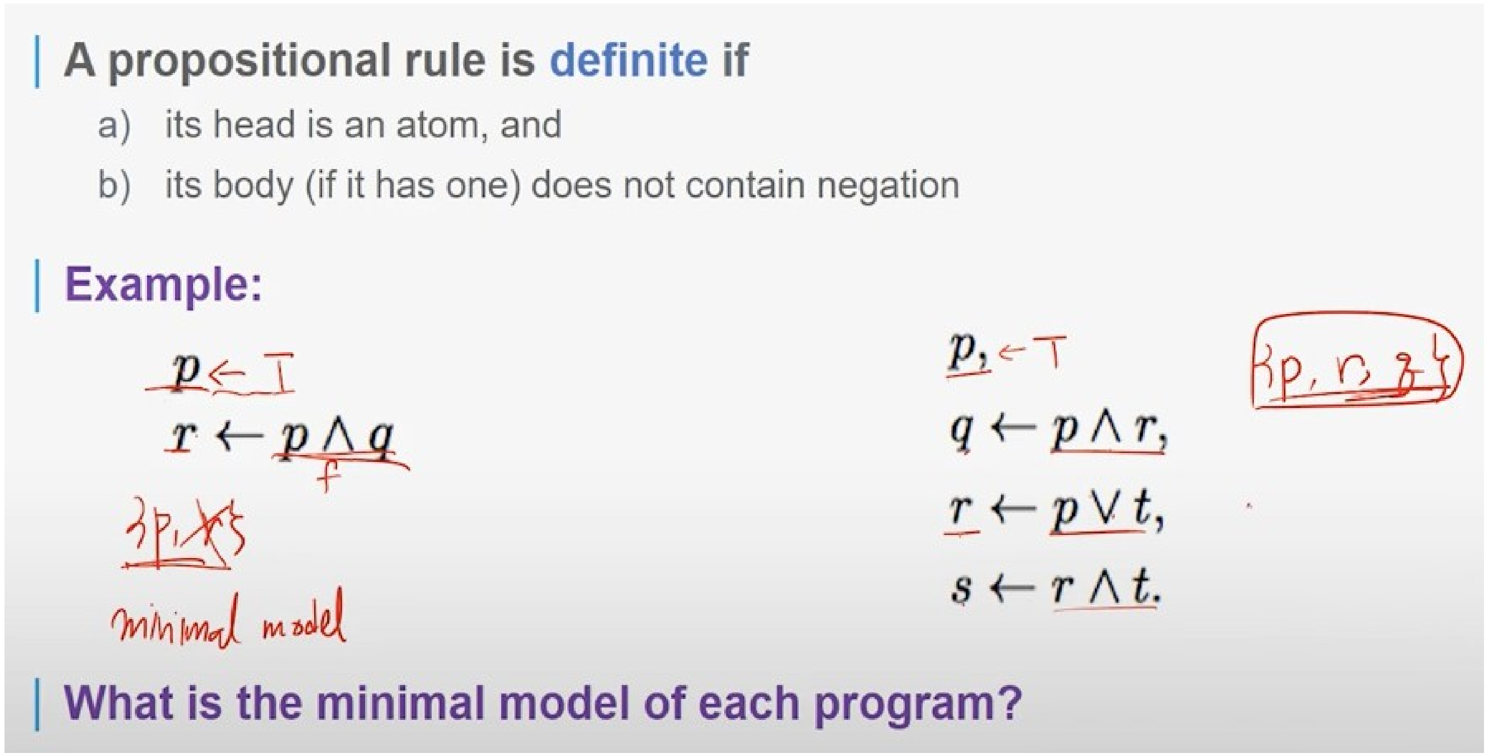
Statement: If two formulas are equivalent under propositional logic, then they have the same minimal models.

Question: Is the converse true, that two formulas having the same minimal models are equivalent?

$\underline{p \leftarrow q}$ $\underline{\underline{q \leftarrow p}}$

min. models $\emptyset$ $\emptyset$

$\boxed{\{p\}}$ $\underline{\{q\}}$



A propositional rule is definite if

a) its head is an atom, and

b) its body (if it has one) does not contain negation

Example:

$p \leftarrow \top$

$r \leftarrow \underbrace{p \wedge q}_{f}$

$\{p, \cancel{r}\}$

minimal model

$p, \leftarrow \top$

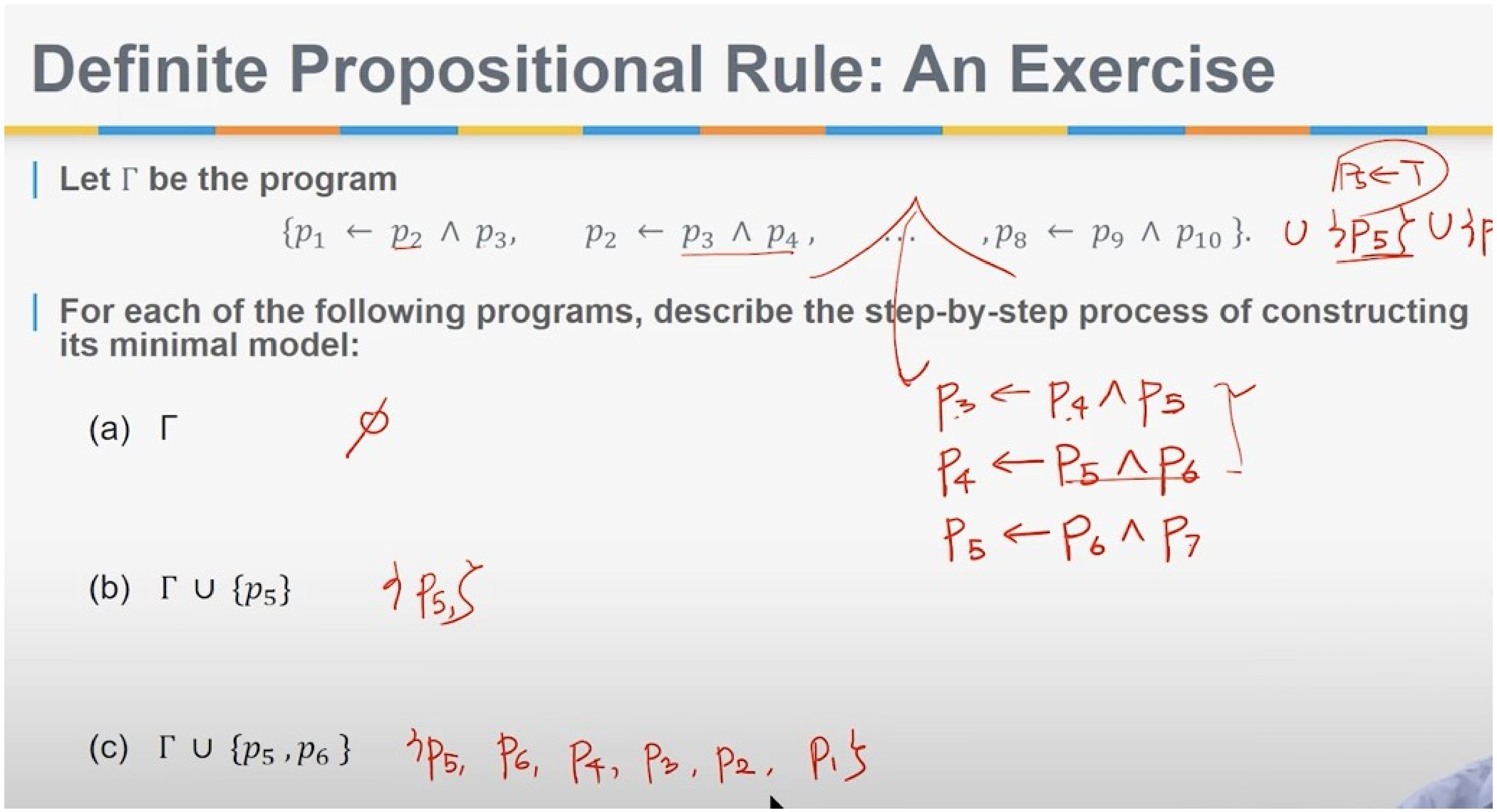
$q \leftarrow p \wedge r,$

$r \leftarrow p \vee t,$

$s \leftarrow r \wedge t.$

$\boxed{\{p, r, q\}}$

What is the minimal model of each program?



Definite Propositional Rule: An Exercise

- Let Γ be the program

$$\{p_1 \leftarrow p_2 \wedge p_3, \quad p_2 \leftarrow p_3 \wedge p_4, \quad \ldots \quad , p_8 \leftarrow p_9 \wedge p_{10}\}. \quad \cup \{\underline{p_5}\} \cup \{p$$

$$\boxed{p_3 \leftarrow T}$$

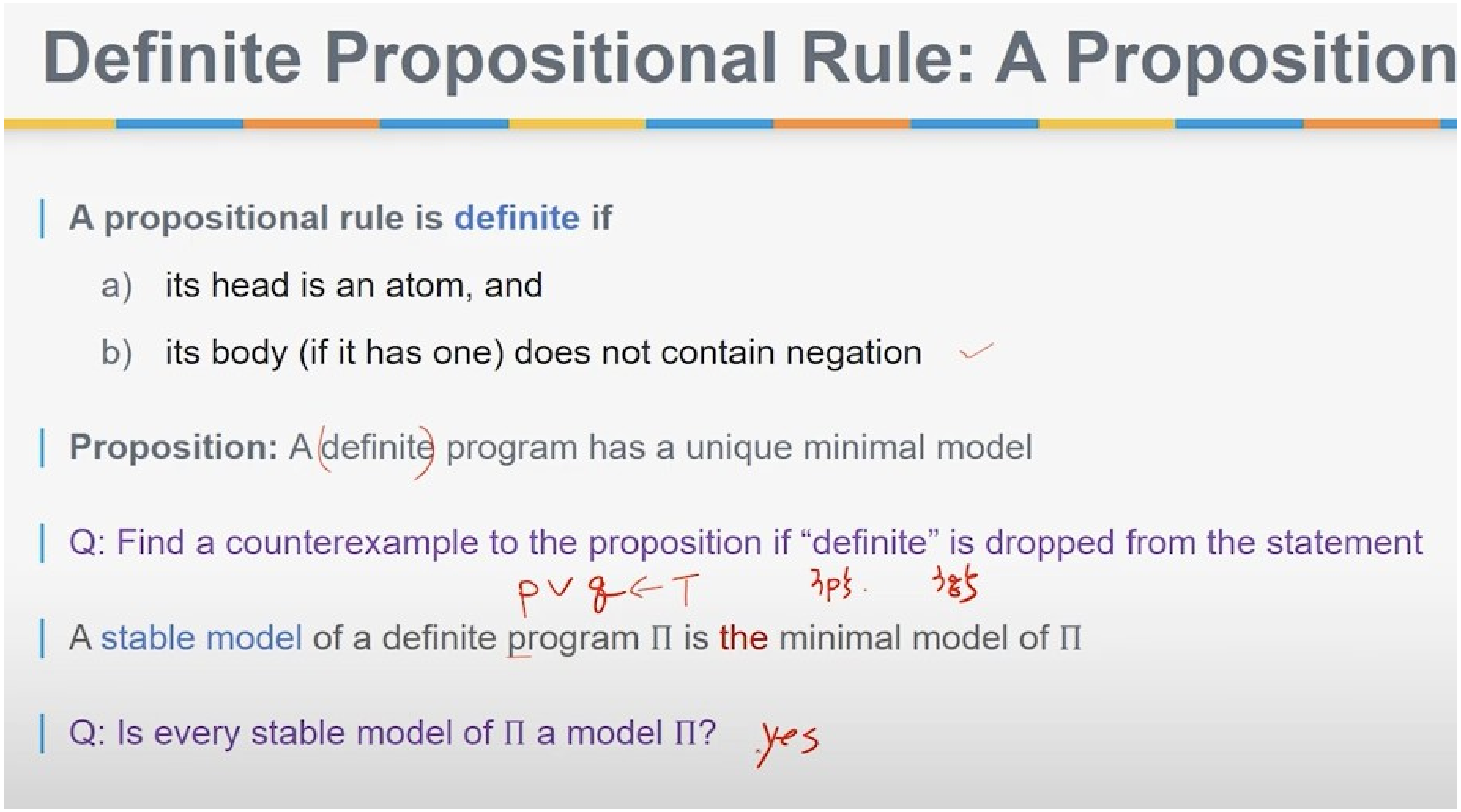
- For each of the following programs, describe the step-by-step process of constructing its minimal model:

$$p_3 \leftarrow p_4 \wedge p_5$$
$$p_4 \leftarrow \cancel{p_5 \wedge p_6}$$
$$p_5 \leftarrow p_6 \wedge p_7$$

(a) Γ $\quad \emptyset$

(b) $\Gamma \cup \{p_5\}$ $\quad \{p_5,\}$

(c) $\Gamma \cup \{p_5, p_6\}$ $\quad \{p_5, p_6, p_4, p_3, p_2, p_1\}$



Definite Propositional Rule: A Proposition

- A propositional rule is **definite** if
  a) its head is an atom, and
  b) its body (if it has one) does not contain negation ✓
- **Proposition:** A definite program has a unique minimal model
- Q: Find a counterexample to the proposition if "definite" is dropped from the statement

  $p \vee q \leftarrow \top$ $\{p\}$. $\{q\}$
- A stable model of a definite program Π is the minimal model of Π
- Q: Is every stable model of Π a model Π? yes

Module 3: Propositional Image of Formulas & Intervals

- Comma in the head replaces $\vee$
- Comma in the body replaces $\wedge$

- If head is empty, replaces $\perp$
- If body is empty, replaces T

- Atom p(1..n) the head: conjunction “$\wedge$” of all formulas
- Atom p(1..n) the body: disjunction “$\vee$” of all formulas

- Comparison in the head: T if all “$\wedge$” comparisons are true, otherwise $\perp$
- Comparison in the body: T if some “$\vee$” comparisons are true, otherwise $\perp$

Module 3: Negation as Failure

- It is about semantics. If you cannot prove “p” is true, then assume “¬p” is true. So we can break the cycles in the semantics (meaning), as seen below:

  $p \leftarrow \neg q$

  $q \leftarrow \neg p$

- Until now, we didn't allow negation in definite/positive programs.
- Until now, each definite program has a single/unique stable model.
- When we use negation, a program can have:
  - More than one stable model,
  - One stable model
  - Or no stable models.

Module 3: Negation as Failure

To define “Negation as Failure”, we need auxiliary concepts:

- A Critical part: of a rule is a sub-formula of its head/body that begins with negation, but it is not part of another sub-formula that begins with negation.
- The Reduct Π^{x}: of a program Π relative to the interpretation X is the positive program (no negation), by replacing each critical part ¬H with
  - T if X satisfies ¬H
  - $\perp$ otherwise

There are three concepts:

- Π: is the original program that we want to find its stable model.
- X: the program may have many interpretations=2^{n}. X is just one interpretation.
- Π^{x}: Reduct of program Π for interpretation X.

If X is a minimal model of Π^{x}, then X is a stable model of original program Π.

Thanks

&

Questions