Foundations of Logic & Inference

Stuart C. Shapiro

Department of Computer Science
and Center for Cognitive Science
State University of New York at Buffalo
Buffalo, NY 14260-2000

shapiro@cs.buffalo.edu

copyright 1995 by Stuart C. Shapiro

Contents

1 Introduction

- **Logic** is the study of correct reasoning.
- It is not a particular KRR language.
- There are many systems of logic (logics).
- AI KRR research can be seen as a hunt for the "right" logic.

Commonalities among Logics

- System for reasoning.
- Prevent reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)
- Language for expressing reasoning steps.

Parts of the Study/Specification of a Logic

Syntax: The rules for constructing well-formed expressions of the logic.

Semantics: The rules for determining the meaning of an expression of the logic.

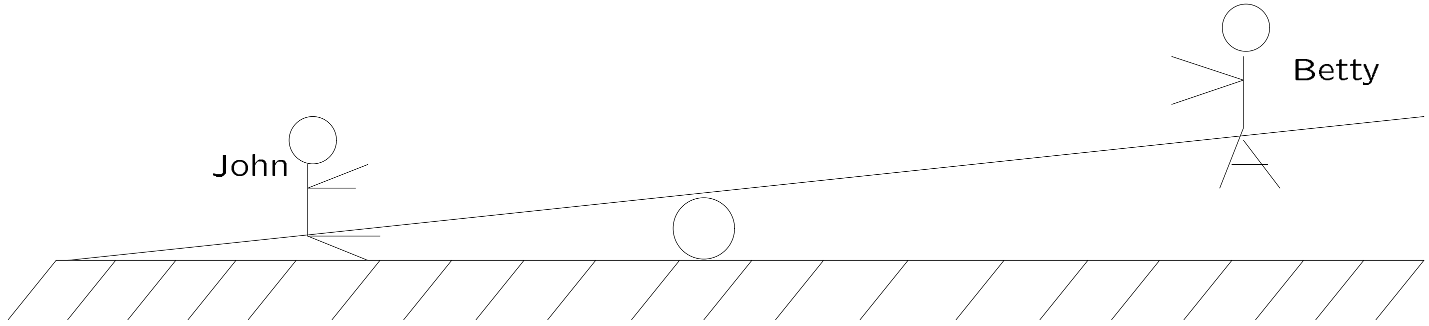
Syntactic Inference: The rules for determining the theorems of the logic.

Semantic Inference: The rules for identifying the logically valid expressions of the logic.

2 Propositional Logic

Logics that do not analyze information below the level of the sentence (proposition).

2.1 Seesaw World



Sentences

John is up

John is down

Betty is up

Betty is down

2.2 The "Standard" Propositional Logic

2.2.1 Syntax of the "Standard" Propositional Logic

Atomic Propositions

- Any letter of the alphabet, e.g.: P
- Any letter of the alphabet with a numeric subscript, e.g.: Q_3
- Any alphanumeric string, e.g.: *John is up*

is an atomic proposition.

Well-Formed Propositions (WFPs)

1. Every atomic proposition is a wfp.

2. If P is a wfp, then so is $\neg P$.

3. If P and Q are wfps, then so is

   (a) $P \wedge Q$ (b) $P \vee Q$

   (c) $P \Rightarrow Q$ (d) $P \Leftrightarrow Q$

4. Nothing else is a wfp.

Examples of WFPs

$$\neg(A \land B) \Leftrightarrow (\neg A \lor \neg B)$$

$$\textit{John is down} \Rightarrow \textit{Betty is up}$$

$$\textit{John is up} \Leftrightarrow \neg\textit{Betty is up}$$

2.2.2.1 Semantics of Atomic Propositions

Extensional Semantics:

- Relative to an interpretation/model/situation.
- Either True or False.

Intensional Semantics:

- Dependent on a Domain.
- Independent of any specific interpretation/model/situation.
- Statement in a previously understood language (e.g. English) that allows truth value to be determined in any specific situation.
- Often omitted, but shouldn't be.

Intensional Seesaw World Semantics

John is down: The individual named John is in the lower position on his end of the seesaw.

John is up: The individual named John is in the higher position on his end of the seesaw.

Betty is down: The individual named Betty is in the lower position on her end of the seesaw.

Betty is up: The individual named Betty is in the higher position on her end of the seesaw.

Alternative Intensional Seesaw World Semantics

John is down: The individual named Betty is in the higher position on her end of the seesaw.

John is up: The individual named John is in the lower position on his end of the seesaw.

Betty is down: The individual named John is in the higher position on his end of the seesaw.

Betty is up: The individual named Betty is in the lower position on her end of the seesaw.

Alternative Seesaw World
Syntax/Intensional Semantics

A: The individual named John is in the lower position on his end of the seesaw.

B: The individual named John is in the higher position on his end of the seesaw.

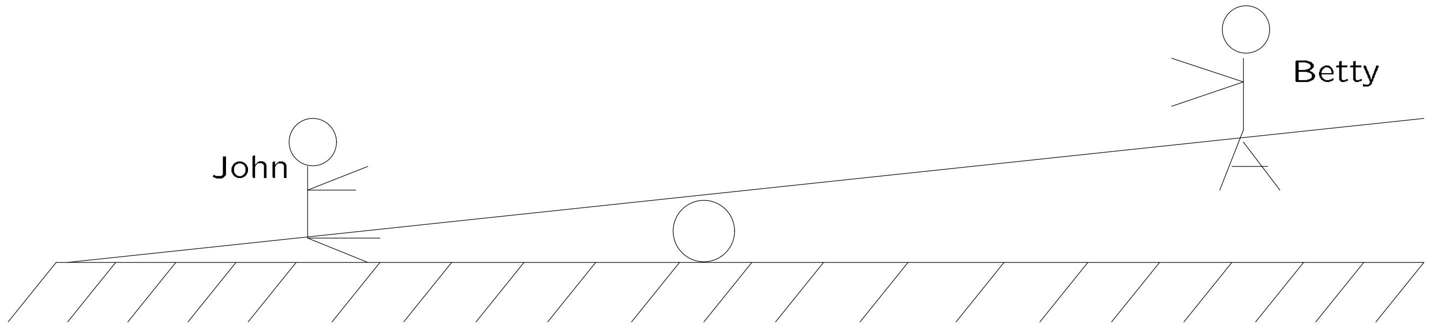
C: The individual named Betty is in the lower position on her end of the seesaw.

D: The individual named Betty is in the higher position on her end of the seesaw.

Intensional Semantics Moral

- Don't omit.
- Don't presume.
- No "pretend it's English semantics".

Seesaw World Extensional Semantics



John is up: False | *Betty is up*: True

John is down: True | *Betty is down*: False

Note: n propositions $\Rightarrow 2^n$ possible situations.

2.2.2.2.1 Intensional Semantics of WFPs

$[\neg P] =$ It is not the case that $[P]$.

$[P \wedge Q] = [P]$ and $[Q]$.

$[P \vee Q] =$ Either $[P]$ or $[Q]$ or both.

$[P \Rightarrow Q] =$ If $[P]$ then $[Q]$.

$[P \Leftrightarrow Q] = [P]$ if and only if $[Q]$.

Example Seesaw World Intensional WFP Semantics

[$\textit{John is up} \Leftrightarrow \neg\textit{Betty is up}$] = The individual named John is in the higher position on his end of the seesaw if and only if it is not the case that the individual named Betty is in the higher position on her end of the seesaw.

2.2.2.2.2 Extensional Semantics of WFPs

$[\![\neg P]\!]$ is True if $[\![P]\!]$ is False. Otherwise, it is False.

$[\![P \wedge Q]\!]$ is True if $[\![P]\!]$ is True and $[\![Q]\!]$ is True. Otherwise, it is False.

$[\![P \vee Q]\!]$ is False if $[\![P]\!]$ is False and $[\![Q]\!]$ is False. Otherwise, it is True.

$[\![P \Rightarrow Q]\!]$ is False if $[\![P]\!]$ is True and $[\![Q]\!]$ is False. Otherwise, it is True.

$[\![P \Leftrightarrow Q]\!]$ is True if $[\![P]\!]$ and $[\![Q]\!]$ are both True, or both False. Otherwise, it is False.

Extensional Semantics Truth Tables

P	$\neg P$
T	F
F	T

P	Q	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

Abbreviating "True" as "T" and "False" as "F".

Example Seesaw World Truth Table

John is up	*John is down*	*John is up* $\Leftrightarrow \neg$*John is down*
F	T	T

2.2.3 Syntactic Inference

- Specifies when a given wfp can be derived correctly from a set of (other) wfps.

$$A_1, \ldots, A_n \vdash P$$

- Determines what wfps are **theorems** of the logic.

$$\vdash P$$

- Depends on the notion of **proof**.

2.2.3.1 Hilbert-Style Syntactic Inference

- Set of **Axioms**.

- Small set of **Rules of Inference**.

Hilbert-Style Proof

- A **proof** of a theorem P is
  - An ordered list of wfps ending with P
  - Each wfp on the list is
    - Either an axiom
    - or follows from previous wfps in the list by one of the rules of inference.

Hilbert-Style Derivation

- A **derivation** of P from $A_1, \ldots, A_n$ is
  - A list of wfps ending with P
  - Each wfp on the list is
    - Either an axiom
    - or some A_i
    - or follows from previous wfps in the list by one of the rules of inference.

Example Hilbert-Style Axioms*

All instances of:

(A1). $(\mathcal{A} \Rightarrow (\mathcal{B} \Rightarrow \mathcal{A}))$

(A2). $((\mathcal{A} \Rightarrow (\mathcal{B} \Rightarrow \mathcal{C})) \Rightarrow ((\mathcal{A} \Rightarrow \mathcal{B}) \Rightarrow (\mathcal{A} \Rightarrow \mathcal{C})))$

(A3). $((\neg\mathcal{B} \Rightarrow \neg\mathcal{A}) \Rightarrow ((\neg\mathcal{B} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{B}))$

*From E. Mendelson, *Introduction to Mathematical Logic*, (Princeton: D. Van Nostrand) 1964, pp. 31–32.

Hilbert-Style Rule of Inference

Modus Ponens

$$\frac{\mathcal{A}, \mathcal{A} \Rightarrow \mathcal{B}}{\mathcal{B}}$$

Example Hilbert-Style Proof

(1)	$(A \Rightarrow ((A \Rightarrow A) \Rightarrow A))$	
	$\Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$	Instance of A2
(2)	$A \Rightarrow ((A \Rightarrow A) \Rightarrow A)$	Instance of A1
(3)	$(A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A)$	From 1, 2 by MP
(4)	$A \Rightarrow (A \Rightarrow A)$	Instance of A1
(5)	$A \Rightarrow A$	From 3, 4 by MP

Example Hilbert-Style Derivation

Of	$\neg$*Betty is down*
from	*John is down* $\Rightarrow$ *Betty is up*,
	Betty is up $\Rightarrow \neg$*Betty is down*,
and	*John is down*

(1)	*John is down*	Assumption
(2)	*John is down* $\Rightarrow$ *Betty is up*	Assumption
(3)	*Betty is up*	From 1, 2 by MP
(4)	*Betty is up* $\Rightarrow \neg$ *Betty is down*	Assumption
(5)	$\neg$ *Betty is down*	From 3, 4 by MP

Some AI Connections

AI	Logic
rules or domain rules	non-atomic assumptions or non-logical axioms
inference engine procedures	rules of inference
knowledge base	proof

2.2.3.2 Natural Deduction
(Style of Syntactic Inference)

- **No** Axioms.

- **Large** set of Rules of Inference.

  - A few **structural rules** of inference.

  - An **introduction rule** and an **elimination rule** for each connective.

Fitch-Style Proof*

- A **proof** of a theorem P is
  - An ordered list of wfps and subproofs ending with P
  - Each wfp or subproof on the list must be justified by a rule of inference.

*Based on F. B. Fitch, *Symbolic Logic: An Introduction*, (New York: Ronald Press), 1952.

Structural Rules of Inference

$$\begin{array}{r|l} i. & A \quad Hyp \\ \hline & \end{array}$$

$$\begin{array}{r|l} i. & A \\ & \vdots \\ j. & A \quad Rep, i \end{array}$$

$$\begin{array}{r|l} i. & A \\ & \vdots \\ & \begin{array}{r|l} j. & A \quad Reit, i \end{array} \end{array}$$

Fitch-Style Derivation

- A **derivation** of a theorem P from an assumption A is a hypothetical subproof whose hypothesis is A and which contains
  - An ordered list of wfps and inner subproofs ending with P
  - Each wfp or inner subproof on the list must be justified by a rule of inference.

Rules for $\neg$

$$
\begin{array}{r|l}
 & \begin{array}{r|ll}
i. & A & Hyp \\ \hline
 & \vdots & \\
j. & B & \\
j+1. & \neg B &
\end{array} \\
j+2. & \neg A \qquad\qquad\qquad\qquad \neg I, i\text{–}(j+1)
\end{array}
$$

$$
\begin{array}{r|ll}
i. & \neg\neg A & \\
j. & A & \neg E, i
\end{array}
$$

Rules for $\wedge$

$$
\begin{array}{r|ll}
i. & A & \\
 & \vdots & \\
j. & B & \\
k. & A \wedge B & \wedge I, i, j
\end{array}
$$

$$
\begin{array}{r|ll}
i. & A \wedge B & \\
j. & A & \wedge E, i
\end{array}
\qquad\qquad
\begin{array}{r|ll}
i. & A \wedge B & \\
j. & B & \wedge E, i
\end{array}
$$

Rules for $\vee$

$$\begin{array}{r|l}
i. & A \\
j. & A \vee B \quad \vee I, i
\end{array}
\qquad\qquad
\begin{array}{r|l}
i. & B \\
j. & A \vee B \quad \vee I, i
\end{array}$$

$$\begin{array}{r|lr}
i. & A \vee B & \\
 & \begin{array}{r|l} i+1. & A \quad Hyp \\ \hline & \vdots \\ j. & C \end{array} & \\
 & & \\
 & \begin{array}{r|l} j+1. & B \quad Hyp \\ \hline & \vdots \\ k. & C \end{array} & \\
k+1. & C & \vee E, i\text{–}k
\end{array}$$

Rules for $\Rightarrow$

$$
\begin{array}{r|l}
 & \begin{array}{r|ll} i. & A & Hyp \\ \hline & \vdots & \\ j. & B & \end{array} \\
k. & A \Rightarrow B \qquad\qquad \Rightarrow I, i\text{–}j
\end{array}
\qquad\qquad
\begin{array}{r|ll}
i. & A & \\
 & \vdots & \\
j. & A \Rightarrow B & \\
k. & B & \Rightarrow E, i, j
\end{array}
$$

Rules for $\Leftrightarrow$

$$
\begin{array}{r|l}
 & \begin{array}{r|ll} i. & A & Hyp \\ \hline & \vdots & \\ j. & B & \end{array} \\
 & \\
 & \begin{array}{r|ll} j+1. & B & Hyp \\ \hline & \vdots & \\ k. & A & \end{array} \\
k+1. & A \Leftrightarrow B \qquad\qquad\qquad \Leftrightarrow I, i\text{–}k
\end{array}
$$

$$
\begin{array}{r|l}
i. & A \\
 & \vdots \\
j. & A \Leftrightarrow B \\
k. & B \qquad\qquad \Leftrightarrow E, i, j
\end{array}
\qquad\qquad
\begin{array}{r|l}
i. & B \\
 & \vdots \\
j. & A \Leftrightarrow B \\
k. & A \qquad\qquad \Leftrightarrow E, i, j
\end{array}
$$

Example Fitch-Style Proof

$$
\begin{array}{lll}
1. & \quad A & Hyp \\
\hline
2. & \quad A & Rep, 1 \\
3. & A \Rightarrow A & \Rightarrow I, 1\text{–}2
\end{array}
$$

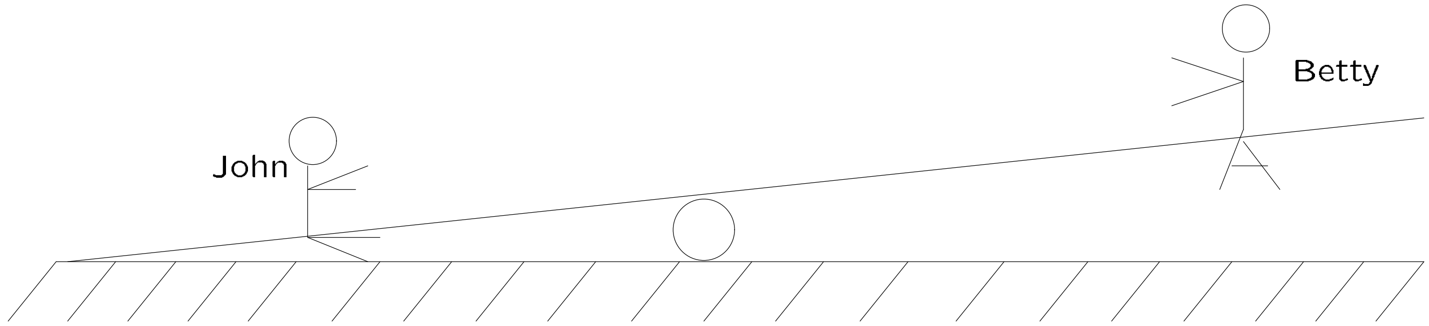
Example Fitch-Style Derivation

1.	*John is down*	
	$\wedge$(*John is down* $\Rightarrow$ *Betty is up*)	
	$\wedge$(*Betty is up* $\Rightarrow \neg$*Betty is down*)	Hyp
2.	*John is down*	$\wedge E, 1$
3.	*John is down* $\Rightarrow$ *Betty is up*	$\wedge E, 1$
4.	*Betty is up*	$\Rightarrow E, 2, 3$
5.	*Betty is up* $\Rightarrow \neg$*Betty is down*	$\wedge E, 1$
6.	$\neg$*Betty is down*	$\Rightarrow E, 4, 5$

Fitch-Style Proof of Axiom A1

$$
\begin{array}{lllll}
 & 1. & A & & Hyp \\
\hline
 & & 2.\ B & Hyp & \\
 & & 3.\ A & Reit, 1 & \\
 & 4. & B \Rightarrow A & & \Rightarrow I, 2\text{–}3 \\
5. & A \Rightarrow (B \Rightarrow A) & & & \Rightarrow I, 1, \text{–}4
\end{array}
$$

Domain Rules for Seesaw World



1. *John is up* $\Leftrightarrow \neg$*John is down*
2. *John is down* $\Leftrightarrow$ *Betty is up*
3. *Betty is up* $\Leftrightarrow \neg$*Betty is down*

Seesaw Situation Derivation

1.	*John is up* $\Leftrightarrow \neg$*John is down* $\wedge$*John is down* $\Leftrightarrow$ *Betty is up* $\wedge$*Betty is up* $\Leftrightarrow \neg$*Betty is down* $\wedge$*John is up*	Hyp
2.	*John is up*	$\wedge E, 1$
3.	*John is up* $\Leftrightarrow \neg$*John is down*	$\wedge E, 1$
4.	$\neg$*John is down*	$\Leftrightarrow E, 3$
5.	$\neg$*Betty is down*	Hyp
6.	*Betty is up* $\Leftrightarrow \neg$*Betty is down*	$\wedge E, 1$
7.	*Betty is up*	$\Leftrightarrow E, 5, 6$
8.	*John is down* $\Leftrightarrow$ *Betty is up*	$\wedge E, 1$
9.	*John is down*	$\Leftrightarrow E, 7, 8$
10.	$\neg$*John is down*	$Reit, 4$
11.	$\neg\neg$*Betty is down*	$\neg I$, 5–10
12.	*Betty is down*	$\neg E, 11$

More Connections

- $A \vdash P$ if and only if $\vdash A \Rightarrow P$.

- So proving theorems and deriving conclusions from assumptions are equivalent.

- But no atomic proposition is a theorem.

- So theorem proving makes more use of Introduction Rules than most AI reasoning systems.

2.2.4 Semantic Inference
1 Terminology

- P is **logically valid** ($\models P$) if and only if it is True under all interpretations (in all situations).

- $A_1, \ldots, A_n$ **logically implies** P ($A_1, \ldots, A_n \models P$) if and only if every interpretation that makes $A_1 \land \ldots \land A_n$ *True* also makes P *True*.

- P is **contradictory** if it is True in no situation.

- P is **satisfiable** if it is True in some situations and False in some others.

2.2.4 Semantic Inference

Semantic Inference methods

- allow one to **identify** the logically valid/contradictory/satisfiable wfps of the logic.

- allow one to **decide** whether a wfp is logically implied by a set of others.

NB: These issues are **determined** by the semantics, but **decided** by the semantic inference methods.

Truth Table Method Applied to Axiom A1

A	B	$B \Rightarrow A$	$A \Rightarrow (B \Rightarrow A)$
T	T	T	T
T	F	T	T
F	T	F	T
F	F	T	T

Truth Table Method
Applied to a Theorem

A	B	$\ldots$	$\neg(A \wedge B)$	$\neg A \vee \neg B$	$\neg(A \wedge B) \Leftrightarrow (\neg A \vee \neg B)$
T	T	$\ldots$	F	F	T
T	F	$\ldots$	T	T	T
F	T	$\ldots$	T	T	T
F	F	$\ldots$	T	T	T

Truth Table Method

Applied to a Logical Implication

	* JD	BU	BD	* $(JD \Rightarrow BU)$	* $(BU \Rightarrow \neg BD)$	$\neg BD$
	T	T	T	T	F	F
*	T	T	F	T	T	T
	T	F	T	F	T	F
	T	F	F	F	T	T
	F	T	T	T	F	F
	F	T	F	T	T	T
	F	F	T	T	T	F
	F	F	F	T	T	T

So, *(John is down)*, *(John is down $\Rightarrow$ Betty is up)*, *(Betty is up $\Rightarrow \neg$ Betty is down)* $\models \neg$ *(Betty is down)*

Significance of Domain Rules

To see if
Domain Rules $\models$ *Query*
Restrict Truth Table to rows that satisfy Domain Rules.

Restricted Seesaw World

JU	JD	BU	BD	$JU \Leftrightarrow \neg JD$	$JD \Leftrightarrow BU$	$BU \Leftrightarrow \neg BD$	Q
T	F	F	T	T	T	T	?
F	T	T	F	T	T	T	?

More Elimination of Distinctions

- $A_1, \ldots, A_n \models P$ if and only if $\models A_1 \wedge A_n \Rightarrow P$.
- So deciding validity and logical implication are equivalent.

3 Important Properties of Logical Systems

Soundness: $\vdash P$ implies $\models P$

Consistency: not both $\vdash P$ and $\vdash \neg P$

Completeness: $\models P$ implies $\vdash P$

Connections

- If at most 1 of $\models P$ and $\models \neg P$ then soundness implies consistency.

- Soundness is the essence of "correct reasoning."

- Completeness less important for running systems since a proof may take too long to wait for.

- The Propositional Logics we have been looking at are complete.

- Gödel's Incompleteness Theorem: A logic strong enough to formalize arithmetic is *either* inconsistent *or* incomplete.

More Connections

$$A_1, \ldots, A_n \vdash P \qquad \Leftrightarrow \qquad \vdash A_1 \wedge \ldots \wedge A_n \Rightarrow P$$

$$\text{soundness} \Downarrow\Uparrow \text{completeness} \qquad \text{soundness} \Downarrow\Uparrow \text{completeness}$$

$$A_1, \ldots, A_n \models P \qquad \Leftrightarrow \qquad \models A_1 \wedge \ldots \wedge A_n \Rightarrow P$$

Wang's Algorithm*
A Semantic Inference Method

To Show $A \models P$:

- Try to construct a truth table line in which A is True and P is false.

- If you succeed, $A \not\models P$.

- If you fail, $A \models P$.

*Wang, H. 1960. Toward Mechanical Mathematics. IBM Journal of Research and Development 4, 2-22. Reprinted in K. M. Sayre and F. J. Crosson (Eds.) The Modeling of Mind: Computers and Intelligence. Simon and Schuster, New York, 1963.

Implemented Wang Function

`(wang '(`$A_1, \ldots, A_n$`) '(`P`))`

Returns `T` if $A_1, \ldots, A_n \models P$;

`NIL` otherwise.

Using Wang's Algorithm on Axiom A1

```
(wang                '()    '((A => (B => A))))
 0: (WANG            NIL    ((A => (B => A))))
   1: (WANG          (A)    ((B => A)))
     2: (WANG        (B A) (A))
     2: returned T
   1: returned T
 0: returned T
T
```

Using Wang's Algorithm on a Non-Theorem

```
(wang                    '() '((A => (A & B))))
 0: (WANG                NIL ((A => (A & B))))
   1: (WANG              (A) ((A & B)))
     2: (WANG            (A) (A))
     2: returned T
     2: (WANG            (A) (B))
     2: returned NIL
   1: returned NIL
 0: returned NIL
NIL
```

Using Wang's Algorithm to see if

$$JD, JD \Rightarrow BU, BU \Rightarrow \neg BD \models \neg BD$$

```
(wang               '(JD (JD => BU) (BU => (~ BD))) '((~ BD)))
 0: (WANG           (JD (JD => BU) (BU => (~ BD)))  ((~ BD)))
   1: (WANG         (BU JD (BU => (~ BD)))          ((~ BD)))
     2: (WANG       ((~ BD) BU JD)                  ((~ BD)))
     2: returned T
     2: (WANG       (BU JD)                        (BU (~ BD)))
     2: returned T
   1: returned T
   1: (WANG         (JD (BU => (~ BD)))            (JD (~ BD)))
   1: returned T
 0: returned T
T
```

Alternative View of Wang Function

(`wang` *KB* *Query*)
Returns `T` if the *Query* follows from the *KB*
`NIL` otherwise.

2.3 Clause Form Propositional Logic

2.3.1 Clause Form Syntax part 1

Atomic Propositions:

- Any letter of the alphabet
- Any letter with a numeric subscript
- Any alphanumeric string.

Literals:

If P is an atomic proposition, P and $\neg P$ are literals.
P is called a **positive literal**
$\neg P$ is called a **negative literal**.

2.3.1 Clause Form Syntax part 2

Clauses: If $L_1, \ldots, L_n$ are literals
then the set $\{L_1, \ldots, L_n\}$ is a clause.

Sets of Clauses: If $C_1, \ldots, C_n$ are clauses
then the set $\{C_1, \ldots, C_n\}$ is a set of clauses.

2.3.2 Clause Form Semantics
Atomic Propositions

Intensional: $[P]$ is some proposition in the domain.

Extensional: $\llbracket P \rrbracket$ is either True or False.

2.3.2 Clause Form Semantics

Literals

Positive Literals: The meaning of P as a literal is the same as it is as an atomic proposition.

Negative Literals:

Intensional:
$[\neg P]$ means that it is not the case that $[P]$.

Extensional: $[\![\neg P]\!]$ is True if $[\![P]\!]$ is False.
$[\![\neg P]\!]$ is False if $[\![P]\!]$ is True.

2.3.2 Clause Form Semantics

Clauses

Intensional:

$[\{L_1, \ldots, L_n\}] = [L_1]$ and/or ... and/or $[L_n]$.

Extensional:

$[\![\{L_1, \ldots, L_n\}]\!]$ is True
if at least one of $[\![L_1]\!]$, ..., $[\![L_n]\!]$ is True.
It is False if $[\![L_1]\!]$ and ... and $[\![L_n]\!]$ are all False.

2.3.2 Clause Form Semantics
Sets of Clauses

Intensional:
$[\{C_1, \ldots, C_n\}] = [C_1]$ and ... and $[C_n]$.

Extensional:
$[\![\{C_1, \ldots, C_n\}]\!]$ is True if $[\![C_1]\!]$ and ... and $[\![C_n]\!]$ are all True.
It is False if at least one of $[\![C_1]\!]$, ..., $[\![C_n]\!]$ is False.

2.3.3 Clause Form Syntactic Inference: Resolution

Notion of Proof: A set of clauses constitutes a proof.

Assumptions: The proof is initialized with a set of assumption clauses $AC_1, \ldots, AC_n$.

Rule of Inference: A clause may be added to a set of clauses if justified by **resolution**.

Derived Clause: If clause CQ has been added to a set of clauses initialized with the set of assumption clauses $AC_1, \ldots, AC_n$ by one or more applications of resolution, then $AC_1, \ldots, AC_n \vdash CQ$.

Resolution

$$\frac{\{P, L_1, \ldots, L_n\}, \{\neg P, L_{n+1}, \ldots, L_m\}}{\{L_1, \ldots, L_n, L_{n+1}, \ldots, L_m\}}$$

Resolution is sound, but not complete!

Example Derivation

1.	$\{\neg(\textit{John is up}), \neg(\textit{John is down})\}$	*Assumption*
2.	$\{(\textit{John is down}), (\textit{Betty is down})\}$	*Assumption*
3.	$\{(\textit{John is up})\}$	*Assumption*
4.	$\{\neg(\textit{John is down})\}$	*R,1,3*
5.	$\{(\textit{Betty is down})\}$	*R,2,4*

Example of Incompleteness

$\{P\} \models \{P, Q\}$

but

Resolution does not apply to $\{\{P\}\}$.

2.3.4 Semantic Inference Contradictory Sets of Clauses

- Notice that $\{\{P\},\{\neg P\}\}$ is contradictory.

- Notice that resolution applies to $\{\{P\},\{\neg P\}\}$ producing $\{\}$, the **empty clause**.

- If a set of clauses is contradictory, repeated application of resolution is **guaranteed** to produce $\{\}$.

Implications

- $\{P_1, \ldots, P_n, Q_1, \ldots, Q_m\}$ is contradictory.

- means $(P_1 \wedge \ldots \wedge P_n \wedge Q_1 \wedge \ldots \wedge Q_m)$ is False for all situations.

- means whenever $(P_1 \wedge \ldots \wedge P_n)$ is True, $(Q_1 \wedge \ldots \wedge Q_m)$ is False.

- means whenever $(P_1 \wedge \ldots \wedge P_n)$ is True $\neg(Q_1 \wedge \ldots \wedge Q_m)$ is True.

- means $P_1, \ldots, P_n \models \neg(Q_1 \wedge \ldots \wedge Q_m)$.

Negation and Clauses

- $\neg\{L_1, \ldots, L_n\} = \{\{\neg L_1\}, \ldots, \{\neg L_n\}\}$.

- $\neg L = \begin{cases} \neg A & \text{if } L = A \\ A & \text{if } L = \neg A \end{cases}$

Resolution Refutation

To decide if $C_1, \ldots, C_n \models CQ$:

1. Let $S = \{C_1, \ldots, C_n\} \cup \neg CQ$
2. Repeatedly apply resolution to clauses in S.
3. If generate $\{\}$, $C_1, \ldots, C_n \models CQ$.
4. If reach point where no new clause can be generated, but $\{\}$ has not appeared, $C_1, \ldots, C_n \not\models CQ$.

Example 1

To decide if $\{P\} \models \{P, Q\}$

$S = \{\{P\}, \{\neg P\}, \{\neg Q\}\}$

$$\begin{array}{lll} 1. & \{P\} & Assumption \\ 2. & \{\neg P\} & From\ query\ clause \\ 3. & \{\} & R, 1, 2 \end{array}$$

Example 2

To decide if

$$\{\neg(John\ is\ up), \neg(John\ is\ down)\},$$
$$\{(John\ is\ down), (Betty\ is\ down)\},$$
$$\{(John\ is\ up)\} \qquad \models \{(Betty\ is\ down)\}$$

1.	$\{\neg(John\ is\ up), \neg(John\ is\ down)\}$	$Assumption$
2.	$\{(John\ is\ down), (Betty\ is\ down)\}$	$Assumption$
3.	$\{(John\ is\ up)\}$	$Assumption$
4.	$\{\neg(Betty\ is\ down)\}$	$From\ query\ clause$
5.	$\{(John\ is\ down)\}$	$R, 2, 4$
6.	$\{\neg(John\ is\ up)\}$	$R, 1, 5$
7.	$\{\}$	$R, 3, 6$

3 Predicate Logic

3.1 Seesaw World

Propositional Logic

$$\begin{array}{ll} \textit{John is up} & \textit{Betty is up} \\ \textit{John is down} & \textit{Betty is down} \end{array}$$

related only by domain rules.

Predicate Logic

$$\begin{array}{ll} Up(John) & Up(Betty) \\ Down(John) & Down(Betty) \end{array}$$

shows two properties and two individuals.

Seesaw World Domain Rules

Propositional Logic

John is up $\Leftrightarrow \neg$*John is down*
John is down $\Leftrightarrow$ *Betty is up*
Betty is up $\Leftrightarrow \neg$*Betty is down*

Predicate Logic

$$\forall x(Up(x) \Leftrightarrow \neg Down(x))$$
$$\forall x, y(Up(x) \Leftrightarrow Down(y))$$

shows the generalization.

Need for $\neq$

But

$$\forall x, y(Up(x) \Leftrightarrow Down(y)) \Rightarrow (Up(John) \Leftrightarrow Down(John))$$

So Domain Rules are:

1. $\forall x(Up(x) \Leftrightarrow \neg Down(x))$

2. $\forall x, y((x \neq y) \Rightarrow (Up(x) \Leftrightarrow Down(y)))$

3. $(John \neq Betty) \wedge (Betty \neq John)$

3.2 The “Standard” Predicate Logic

3.2.1 Syntax of the "Standard" Predicate Logic Atomic Symbols

Individual Constants:

- Any letter of the alphabet (preferably early),
- any (such) letter with a numeric subscript,
- any character string not containing blanks nor other punctuation marks.

For example: a, B_{12}, $John$, *John's_mother-in-law*.

Atomic Symbols, Part 2

Variables:

- Any letter of the alphabet (preferably late),
- any (such) letter with a numeric subscript.

For example: u, v_6.

Skolem Constants:

- Any letter of the alphabet (preferably early),
- any (such) letter with a numeric subscript.

Atomic Symbols, Part 3

Function Symbols:

- Any letter of the alphabet (preferably early middle)
- any (such) letter with a numeric subscript
- any character string not containing blanks.

For example: f, g_2.

Atomic Symbols, Part 4

Skolem Function Symbols:

- Any letter of the alphabet (preferably early middle)
- any (such) letter with a numeric subscript.

Predicate Symbols:

- Any letter of the alphabet (preferably late middle),
- any (such) letter with a numeric subscript,
- any character string not containing blanks.

For example: P, Q_4, odd.

Atomic Symbols, Part 5

Each Function Symbol, Skolem Function Symbol, and Predicate Symbol must have a particular **arity**.

Use superscript for explicit arity.

For example: $\mathit{mother\text{-}of}^1$, Up^1, g_2^3, $\mathit{family\text{-}of}^2$

Atomic Symbols, Part 6

In any specific predicate logic language

Individual Constants,
Variables,
Skolem Constants,
Function Symbols,
Skolem Function Symbols,
Predicate Symbols

must be disjoint.

Terms

- Every individual constant, every Skolem constant, and every variable is a term.

- If f^n is a function symbol or Skolem function symbol of arity n,
  and $t_1, \ldots, t_n$ are terms,
  then $f^n(t_1, \ldots, t_n)$ is a term.
  (The superscript may be omitted if no confusion results.)
  For example: *family-of*^{2}*(John, mother-of*^{1}*(Sally))*

- Nothing else is a term.

Atomic Formulas

If P^n is a predicate symbol of arity n,

and $t_1, \ldots, t_n$ are terms,

then $P^n(t_1, \ldots, t_n)$ is an atomic formula.
E.g.: *child-in*^{2}*(Sally, family-of*^{2}*(John, mother-of*^{1}*(Sally)))*

(The superscript may be omitted if no confusion results.)

Well-Formed Formulas (wffs):

- Every atomic formula is a wff.
- If P is a wff, then so is $\neg P$.
- If P and Q are wffs, then so are
  $P \wedge Q \quad P \vee Q$
  $P \Rightarrow Q \quad P \Leftrightarrow Q$

Well-Formed Formulas (wffs):

If P is a wff and x is a variable, then $\forall x P$ and $\exists x P$ are wffs.

$\forall$ called the **universal quantifier**.

$\exists$ called the **existential quantifier**.

P is called the **scope** of quantification.

Free and Bound Variables

Every occurrence of x in P, not in the scope of some other occurrence of $\forall x$ or $\exists x$, is said to be **free** in P and **bound** in $\forall x P$ and $\exists x P$.

Every occurrence of every variable other than x that is free in P is also free in $\forall x P$ and $\exists x P$.

$$\forall x[P(x,y) \Leftrightarrow [(\exists x \exists z Q(x,y,z)) \Rightarrow R(x,y,)]]$$

Open, Closed, and Ground

A wff with a free variable is called **open**.

A wff with no free variables is called **closed**,

An expression with no variables is called **ground**.

3.2.2 Substitutions

Syntax

Pairs: t/v

- t is any term
- v is any variable

Substitutions: $\{t_1/v_1, \ldots, t_n/v_n\}$

- $i \neq j \Rightarrow v_i \neq v_j$

Terminology

$\sigma = \{t_1/v_1, \ldots, t_n/v_n\}$

t_i is a **term** in σ

v_i is a **variable of** σ

Say $t_i/v_i \in \sigma$ and $v_i \in \sigma$,
but not $t_i \in \sigma$

Note: x is not a variable of $\{x/y\}$,
i.e. $x/y \in \{x/y\}, y \in \{x/y\}, x \notin \{x/y\}$

Substitution Application

For expression $\mathcal{A}$ and substitution $\sigma = \{t_1/v_1, \ldots, t_n/v_n\}$

$\mathcal{A}\sigma$: replace every free occurrence of each v_i in $\mathcal{A}$ by t_i

E.g.: $P(x, y)\{x/y, y/x\} = P(y, x)$

Substitution Composition

$P\sigma\tau = ((P\sigma)\tau) = P(\sigma \circ \tau)$

Let $\sigma = \{t_1/v_1, \ldots, t_n/v_n\}$

$\sigma \circ \tau = \{t_1\tau/v_1, \ldots, t_n\tau/v_n\} \uplus \tau$

$\sigma \uplus \tau = \sigma \cup \{t_i/v_i \mid (t_i/v_i \in \tau) \wedge v_i \notin \sigma\}$

E.g.: $\{x/y, f(a,y)/z\} \circ \{u/y, g(v)/w\} = \{x/y, f(a,u)/z, g(v)/w\}$

Implementation of Substitutions

Represent $\{t_1/v_1, \ldots, t_n/v_n\}$ as `((`v_1 `.` t_1`)` $\cdots$ `(`v_n `.` t_n`))`.

```
(defun apply-sub (substitution wff)
  (sublis substitution wff))

(defun compose (sigma tau)
  (append
   (mapcar #'(lambda (pair)
               (cons (first pair)
                     (apply-sub tau (rest pair))))
           sigma)
   (remove-if #'(lambda (pair)
                  (member (first pair) sigma :key #'first))
              tau)))
```

3.2.3 Semantics of the “Standard” Predicate Logic

Assumes a **Domain**, $\mathcal{D}$, of

- individuals,
- functions on individuals,
- sets of individuals,
- relations on individuals

Let $\mathcal{I}$ be set of all individuals in $\mathcal{D}$.

Semantics of Constants

Individual Constants:
$[\![a]\!]$ is some particular individual in $\mathcal{I}$.

Skolem Constants: If sc created for Skolemization of A,
$[\![sc]\!]$ is the individual described by A.

Semantics of Functional Terms

Function Symbols: $[\![f^n]\!]$ is some function in $\mathcal{D}$,

$[\![f^n]\!]$: $\underbrace{\mathcal{I} \times \cdots \times \mathcal{I}}_{n \text{ times}} \to \mathcal{I}$

Skolem Function Symbols:

If sf^n is created for the Skolemization of the wff A,
then $[\![sf^n]\!]$ is the n-ary function described by A.

Functional Terms:

If f^n is some function symbol or Skolem function symbol,
and $t_1, \ldots, t_n$ are ground terms,
then $[\![f^n(t_1, \ldots, t_n)]\!] = [\![f^n]\!]([\![t_1]\!], \ldots, [\![t_n]\!])$.

Semantics of Predicate Symbols

Predicate Symbols:

- $[\![P^1]\!]$ is some particular subset of $\mathcal{I}$.
- $[\![P^n]\!]$ is some particular subset of the relation

$$\underbrace{\mathcal{I} \times \cdots \times \mathcal{I}}_{n \text{ times}}.$$

Semantics of Atomic Formulas

Atomic Formulas:

- If P^1 is some unary predicate symbol, and t is some ground term, then $[\![P^1(t)]\!]$ is True if $[\![t]\!] \in [\![P^1]\!]$, and False otherwise.

- If P^n is some n-ary predicate symbol, and $t_1, \ldots, t_n$ are ground terms, then $[\![P^n(t_1, \ldots, t_n)]\!]$ is True if $\langle [\![t_1]\!], \ldots, [\![t_n]\!] \rangle \in [\![P^n]\!]$, and False otherwise.

Semantics of WFFs, Part 1

- $[\![\neg P]\!]$ is True if $[\![P]\!]$ is False. Otherwise, it is False.

- $[\![P \wedge Q]\!]$ is True if $[\![P]\!]$ is True and $[\![Q]\!]$ is True. Otherwise, it is False.

- $[\![P \vee Q]\!]$ is False if $[\![P]\!]$ is False and $[\![Q]\!]$ is False. Otherwise, it is True.

- $[\![P \Rightarrow Q]\!]$ is False if $[\![P]\!]$ is True and $[\![Q]\!]$ is False. Otherwise, it is True.

Semantics of WFFs, Part 2

- $[\![P \Leftrightarrow Q]\!]$ is True if $[\![P]\!]$ and $[\![Q]\!]$ are both True, or both False. Otherwise, it is False.

- $[\![\forall x P]\!]$ is True if $[\![P\{t/x\}]\!]$ is True for every ground term, t. Otherwise, it is False.

- $[\![\exists x P]\!]$ is True if there is some ground term, t such that $[\![P\{t/x\}]\!]$ is True. Otherwise, it is False.

Intensional Semantics of Seesaw World 1

Individual Constants:

John: The individual person named John.

Betty: The individual person named Betty.

Intensional Semantics of Seesaw World 2

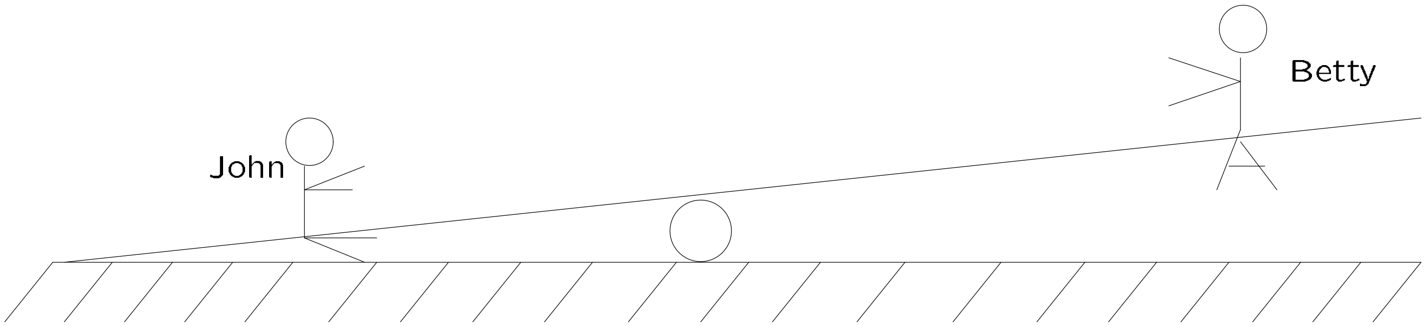
Predicate Symbols:

Up$^1(x)$**:** The individual $[x]$ is in the higher position on his/her end of the seesaw.

Down$^1(x)$**:** The individual $[x]$ is in the lower position on his/her end of the seesaw.

$x \neq y$**:** $[x]$ and $[y]$ are different individuals.

Extensional Semantics of a Seesaw World Situation



$[\![John]\!]$ = the individual person named John.
$[\![Betty]\!]$ = the individual person named Betty.
$[\![Up]\!] = \{$Betty$\}$.
$[\![Down]\!] = \{$John$\}$.
$[\![\neq]\!] = \{\langle$John, Betty $\rangle, \langle$ Betty, John$\rangle\}$.

3.2.4 Syntactic Inference Hilbert-Style

Axioms

(A1). $(\mathcal{A} \Rightarrow (\mathcal{B} \Rightarrow \mathcal{A}))$

(A2). $((\mathcal{A} \Rightarrow (\mathcal{B} \Rightarrow \mathcal{C})) \Rightarrow ((\mathcal{A} \Rightarrow \mathcal{B}) \Rightarrow (\mathcal{A} \Rightarrow \mathcal{C})))$

(A3). $((\neg\mathcal{B} \Rightarrow \neg\mathcal{A}) \Rightarrow ((\neg\mathcal{B} \Rightarrow \mathcal{A}) \Rightarrow \mathcal{B}))$

(A4). $\forall x\mathcal{A} \Rightarrow \mathcal{A}\{t/x\}$
where t is any term containing no variable occurrence that gets bound as a result of the substitution.

(A5). $(\forall x(\mathcal{A} \Rightarrow \mathcal{B})) \Rightarrow (\mathcal{A} \Rightarrow \forall x\mathcal{B})$
if $\mathcal{A}$ is a wff containing no free occurrences of x.

Hilbert-Style Rules of Inference for "Standard" Predicate Logic

$$\frac{\mathcal{A}, \mathcal{A} \Rightarrow \mathcal{B}}{\mathcal{B}}$$

$$\frac{\mathcal{A}}{\forall x \mathcal{A}}$$

Note: $\exists x \mathcal{A}$ is just an abbreviation of $\neg \forall x \neg \mathcal{A}$.

Fitch-Style Natural Deduction

4 Additional Rules of Inference*

$$
\begin{array}{r|ll}
i. & \underline{sc} & SkI \\
 & \vdots & \\
j. & A & \\
\multicolumn{1}{r}{j+1.} & \multicolumn{1}{l}{\forall x A\{x/sc\}} & \forall I, i\text{–}j
\end{array}
\qquad
\begin{array}{r|ll}
i. & \forall x A & \\
i+1. & A\{t/x\} & \forall E, i
\end{array}
$$

$$
\begin{array}{r|ll}
i. & A(t) & \\
i+1. & \exists x A(x) & \exists I, i
\end{array}
\qquad
\begin{array}{r|ll}
i. & \exists x A & \\
i+1. & A\{sc/x\} & \exists E, i
\end{array}
$$

*Adapted from D. Kalish and R. Montague, *Logic: Techniques of Formal Reasoning*, (New York: Harcourt, Brace & World), 1964, pp. 99–101.

Example Fitch-Style Proof

1.	$\neg\forall x A(x)$	Hyp
2.	$\quad \neg\exists x \neg A(x)$	Hyp
3.	$\quad\quad s_1$	SkI
4.	$\quad\quad\quad \neg A(s_1)$	Hyp
5.	$\quad\quad\quad \exists x \neg A(x)$	$\exists I, 4$
6.	$\quad\quad\quad \neg\exists x \neg A(x)$	$Reit, 2$
7.	$\quad\quad \neg\neg A(s_1)$	$\neg I$, 4–6
8.	$\quad\quad A(s_1)$	$\neg E, 7$
9.	$\quad \forall x A(x)$	$\forall I$, 3–8
10.	$\quad \neg\forall x A(x)$	$Reit, 1$
11.	$\neg\neg\exists x \neg A(x)$	$\neg I$, 2–10
12.	$\exists x \neg A(x)$	$\neg E, 11$
13.	$\neg\forall x A(x) \Rightarrow \exists x \neg A(x)$	$\Rightarrow I$, 1–12

Seesaw Situation Derivation

1.	$\forall x(Up(x) \Leftrightarrow \neg Down(x))$ $\wedge \forall x \forall y((x \neq y) \Rightarrow (Up(x) \Leftrightarrow Down(y)))$ $\wedge (John \neq Betty) \wedge (Betty \neq John)$ $\wedge Up(John)$	Hyp
2.	$Up(John)$	$\wedge E, 1$
3.	$(John \neq Betty)$	$\wedge E, 1$
4.	$\forall x \forall y((x \neq y) \Rightarrow (Up(x) \Leftrightarrow Down(y)))$	$\wedge E, 1$
5.	$\forall y((John \neq y) \Rightarrow (Up(John) \Leftrightarrow Down(y)))$	$\forall E, 4$
6.	$((John \neq Betty) \Rightarrow (Up(John) \Leftrightarrow Down(Betty)))$	$\forall E, 5$
7.	$Up(John) \Leftrightarrow Down(Betty))$	$\Rightarrow E, 3, 6$
8.	$Down(Betty)$	$\Leftrightarrow E, 2, 7$

3.2.5 Semantic Inference Without Functions

- n Individual Constants.
- Predicate P^j yields n^j ground atomic propositions.
- k_j predicates of arity j yields $\sum_j (k_j \times n^j)$ ground atomic propositions.
- So $2^{\sum_j (k_j \times n^j)}$ situations (lines of truth table).
- Seesaw World has $2^{(2\times 2^1 + 1\times 2^2)} = 2^8 = 256$ situations.

Seesaw World Situations

$Up(John)$	T	T	F
$Up(Betty)$	T	F	T
$Down(John)$	T	F	T
$Down(Betty)$	T	T	F
$John \neq John$	T	F	F
$John \neq Betty$	T	T	T
$Betty \neq John$	T	T	T
$Betty \neq Betty$	T	F	F
$\forall x(Up(x) \Leftrightarrow \neg Down(x))$	F	T	T
$\forall x \forall y((x \neq y) \Rightarrow (Up(x) \Leftrightarrow Down(y)))$	T	T	T
$John \neq Betty$	T	T	T
$Betty \neq John$	T	T	T
Q	?	?	?

Feasability of Wang's Algorithm

- n Individual Constants.
- $\mathcal{A}$ has k quantified variables.
- $\mathcal{A} \Leftrightarrow$ Boolean combination of k^n ground wffs.
- E.g.:

$$
\begin{array}{l}
\forall x \exists y (x \neq y) \\
\Leftrightarrow \exists y (John \neq y) \wedge \exists y (Betty \neq y) \\
\Leftrightarrow ((John \neq John) \vee (John \neq Betty)) \\
\quad \wedge ((Betty \neq John) \vee (Betty \neq Betty))
\end{array}
$$

Semantic Inference With Functions

Add to Seesaw World:
$playmate(x)$: The individual playing with the individual x.
$mother(x)$: The mother of individual x.

Example wffs:
$\forall x(Up(x) \Rightarrow Down(playmate(playmate(x))))$
$\forall x(mother(x) \neq x)$
$\forall x(Chicken(mother(x)) \Rightarrow Chicken(x))$

Number of ground terms, propositions, situations: ∞

Inference methods only semi-decision procedures.

3.3 Clause Form Predicate Logic

3.3.1 Syntax of Clause Form Predicate Logic Atomic Symbols

Individual Constants:

- Any letter of the alphabet (preferably early),
- any (such) letter with a numeric subscript,
- any character string not containing blanks nor other punctuation marks.

For example: a, B_{12}, $John$, *John's_mother-in-law*.

Atomic Symbols, Part 2

Variables:

- Any letter of the alphabet (preferably late),
- any (such) letter with a numeric subscript.

For example: u, v_6.

Atomic Symbols, Part 3

Function Symbols:

- Any letter of the alphabet (preferably early middle)
- any (such) letter with a numeric subscript
- any character string not containing blanks.

For example: f, g_2.
Use superscript for explicit arity.

Atomic Symbols, Part 4

Predicate Symbols:

- Any letter of the alphabet (preferably late middle),
- any (such) letter with a numeric subscript,
- any character string not containing blanks.

For example: P, Q_4, odd.
Use superscript for explicit arity.

Terms

- Every individual constant and every variable is a term.
- If f^n is a function symbol of arity n,
  and $t_1, \ldots, t_n$ are terms,
  then $f^n(t_1, \ldots, t_n)$ is a term.
  (The superscript may be omitted if no confusion results.)
- Nothing else is a term.

Atomic Formulas

If P^n is a predicate symbol of arity n,

and $t_1, \ldots, t_n$ are terms,

then $P^n(t_1, \ldots, t_n)$ is an atomic formula.

(The superscript may be omitted if no confusion results.)

Literals and Clauses

Literals: If P is an atomic formula,
then P and $\neg P$ are literals.

Clauses: If $L_1, \ldots, L_n$ are literals,
then the set $\{L_1, \ldots, L_n\}$ is a clause.

Sets of Clauses: If $C_1, \ldots, C_n$ are clauses,
then the set $\{C_1, \ldots, C_n\}$ is a set of clauses.

3.3.2 Semantics of Clause Form Predicate Logic

- Individual Constants, Function Symbols, Predicate Symbols, Ground Terms, and Ground Atomic Formulas as for Standard Predicate Logic.

- Ground Literals, Ground Clauses, and Sets of Clauses as for Clause Form Propositional Logic.

Semantics of Open Clauses

If clause C contains variables $v_1, \ldots, v_n$,
then $C\{t_1/v_1, \ldots, t_n/v_n\}$ is a **ground instance** of C if it contains no more variables.

If C is an open clause,
$[\![C]\!]$ is True if every ground instance of C is True.
Otherwise, it is False.

3.3.3 Direct Inference
Rules of Inference
Version 1

Resolution: $$\frac{\{P, L_1, \ldots, L_n\}, \{\neg P, L_{n+1}, \ldots, L_m\}}{\{L_1, \ldots, L_n, L_{n+1}, \ldots, L_m\}}$$

Universal Instantion (temporary): $$\frac{C}{C\sigma}$$

Example Derivation

1.	$\{\neg Up(x), \neg Down(x)\}$	$Assumption$
2.	$\{Down(y), Down(z), y = z\}$	$Assumption$
3.	$\{John \neq Betty\}$	$Assumption$
4.	$\{Up(John)\}$	$Assumption$
5.	$\{\neg Up(John), \neg Down(John)\}$	$UI, 1, \{John/x\}$
6.	$\{\neg Down(John)\}$	$R, 4, 5$
7.	$\{Down(John), Down(z), John = z\}$	$UI, 2, \{John/y\}$
8.	$\{Down(z), John = z\}$	$R, 6, 7$
9.	$\{Down(Betty), John = Betty\}$	$UI, 8, \{Betty/z\}$
10.	$\{Down(Betty)\}$	$R, 3, 9$

Note: Clause 7 not ground.

Motivation for a Shortcut

$$\{P(x), L_1(x), \ldots, L_n(x)\} \quad \{\neg P(y), L_{n+1}(y), \ldots, L_m(y)\}$$

$$\downarrow \{a/x, a/y\} \qquad\qquad \downarrow \{a/x, a/y\}$$

$$\{P(a), L_1(a), \ldots, L_n(a)\} \quad \{\neg P(a), L_{n+1}(a), \ldots, L_m(a)\}$$

$$\{L_1(a), \ldots, L_n(a), L_{n+1}(a), \ldots, L_m(a)\}$$

Most General Unifier

A most general unifier (**mgu**), of $\mathcal{A}$ and $\mathcal{B}$

is a substitution, μ,

such that $\mathcal{A}\mu = \mathcal{B}\mu$

$=$ a most general common instance of $\mathcal{A}$ and $\mathcal{B}$

Rule of Inference
Version 2

Resolution: $$\frac{\{A, L_1, \ldots, L_n\}, \{\neg B, L_{n+1}, \ldots, L_m\}}{\{L_1\mu, \ldots, L_n\mu, L_{n+1}\mu, \ldots, L_m\mu\}}$$

where μ is an mgu of A and B.

Example Derivation Revisited

1.	$\{\neg Up(x), \neg Down(x)\}$	$Assumption$
2.	$\{Down(y), Down(z), y = z\}$	$Assumption$
3.	$\{John \neq Betty\}$	$Assumption$
4.	$\{Up(John)\}$	$Assumption$
5.	$\{\neg Down(John)\}$	$R, 1, 4, \{John/x\}$
6.	$\{Down(z), John = z\}$	$R, 2, 5, \{John/y\}$
7.	$\{Down(Betty)\}$	$R, 3, 6, \{Betty/z\}$

Unification

To find the mgu of $\mathcal{A}$ and $\mathcal{B}$.

Some Examples:

$\mathcal{A}$	$\mathcal{B}$	mgu	mgci
$P(a,b)$	$P(a,b)$	$\{\}$	$P(a,b)$
$P(a)$	$P(b)$	$FAIL$	
$P(a,x)$	$P(y,b)$	$\{a/y, b/x\}$	$P(a,b)$
$P(a,x)$	$P(y,g(y))$	$\{a/y, g(a)/x\}$	$P(a,g(a))$
$P(x,f(x))$	$P(y,y)$	$FAIL$	

Unification Algorithm

```
(defun unify (A B &optional mu)
  (cond ((eql mu 'FAIL) 'FAIL)
        ((eql A B) mu)
        ((variablep A) (unifyVar A B mu))
        ((variablep B) (unifyVar B A mu))
        ((or (atom A) (atom B)) 'FAIL)
        ((/= (length A) (length B)) 'FAIL)
        (t (unify (rest A)
                  (rest B)
                  (unify (first A) (first B) mu)))))
```

UnifyVar

```
(defun unifyVar (var term subst)
  (if (var-in-substp var subst)
      (unify (term-of-var-in-subst var subst) term subst)
    (let ((newterm (apply-sub subst term)))
      (cond ((eql var newterm) subst)
            ((occursIn var newterm) 'FAIL)
            (t (compose subst `((,var . ,newterm))))))))
```

Program Assertion

If original $\mathcal{A}$ and $\mathcal{B}$ have no variables in common,

then throughout the above program

no substitution will have one of its variables occurring in one of its terms.

Therefore, for any expression $\mathcal{E}$ and any substitution σ formed in the above program, $\mathcal{E}\sigma\sigma = \mathcal{E}\sigma$.

3.3.4 Refutation Inference Example

To decide if

$\{\neg Up(x), \neg Down(x)\}$, $\{Down(y), Down(z), y = z\}$,
$\{John \neq Betty\}$, $\{Up(John)\}$ $\models \{Down(Betty)\}$

1.	$\{\neg Up(x), \neg Down(x)\}$	$Assumption$
2.	$\{Down(y), Down(z), y = z\}$	$Assumption$
3.	$\{John \neq Betty\}$	$Assumption$
4.	$\{Up(John)\}$	$Assumption$
5.	$\{\neg Down(Betty)\}$	$From\ query\ clause$
6.	$\{Down(y), y = Betty\}$	$R, 2, 5, \{Betty/z\}$
7.	$\{Down(John)\}$	$R, 3, 6, \{John/y\}$
8.	$\{\neg Up(John)\}$	$R, 1, 7, \{John/x\}$
9.	$\{\}$	$R, 4, 8, \{\}$

Predicate Refutation Resolution is Incomplete

1. $\{P(u), P(v)\}$
2. $\{\neg P(x), \neg P(y)\}$
3. $\{P(w), \neg P(z)\} \quad R, 1, 2, \{u/x, w/v, z/y\}$

$\vdots$

Rules of Inference
Version 3

Resolution: $$\frac{\{A, L_1, \ldots, L_n\}, \{\neg B, L_{n+1}, \ldots, L_m\}}{\{L_1\mu, \ldots, L_n\mu, L_{n+1}\mu, \ldots, L_m\mu\}}$$

where μ is an mgu of A and B.

Factoring: $$\frac{\{A, B, L_1, \ldots, L_n\}}{\{A\mu, L_1\mu, \ldots, L_n\mu\}}$$

where μ is an mgu of A and B.
(Note: Special case of UI.)

Refutation with Resolution & Factoring is Complete

1.	$\{P(u), P(v)\}$	
2.	$\{\neg P(x), \neg P(y)\}$	
3.	$\{P(w)\}$	$F, 1, \{w/u, w/v\}$
4.	$\{\neg P(z)\}$	$F, 2, \{z/x, z/y\}$
5.	$\{\}$	$R, 3, 4, \{w/z\}$

Efficiency Rules

Tautology Elimination: If clause C contains literals L and $\neg L$, delete C from the set of clauses.

Pure-Literal Elimination: If clause C contains a literal A $(\neg A)$ and no clause contains a literal $\neg B$ (B) such that A and B are unifiable, delete C from the set of clauses.

Subsumption Elimination: If the set of clauses contains clauses C_1 and C_2 such that there is a substitution σ for which $C_1\sigma \subseteq C_2$, delete C_2 from the set of clauses.

These rules delete unhelpful clauses.

Subsumption may be required to cut infinite loops.

Strategies

Unit Preference: Resolve shorter clauses before longer clauses.

Set of Support: One clause in each pair being resolved must descend from the query.

Many others

These are heuristics for finding {} faster.

3.4 Translating Standard Wffs into Clause Form
Useful Meta-Theorems

- If A is (an occurrence of) a subformula of B, and $\models A \Leftrightarrow C$, then $\models B \Leftrightarrow B\{C/A\}$

- $\forall x_1(\cdots \forall x_n(\cdots \exists y A(x_1, \ldots, x_n, y) \cdots) \cdots)$ is consistent if and only if $\forall x_1(\cdots \forall x_n(\cdots A(x_1, \ldots, x_n, f^n(x_1, \ldots, x_n) \cdots) \cdots)$ is, where f^n is a new Skolem function.
(Note: use a new Skolem constant instead of $f^0()$)

Translating Standard Wffs into Clause Form

Step 1

Eliminate occurrences of $\Leftrightarrow$ using

$$\models (A \Leftrightarrow B) \Leftrightarrow ((A \Rightarrow B) \wedge (B \Rightarrow A))$$

From: $\forall x[parent(x) \Leftrightarrow (person(x) \wedge \exists y(person(y) \wedge child(y,x)))]$

To:

$$\forall x[(parent(x) \Rightarrow (person(x) \wedge \exists y(person(y) \wedge child(y,x))))$$
$$\wedge((person(x) \wedge \exists y(person(y) \wedge child(y,x))) \Rightarrow parent(x))]$$

Translation Step 2

Eliminate occurrences of $\Rightarrow$ using

$$\models (A \Rightarrow B) \Leftrightarrow (\neg A \vee B)$$

From:

$$\forall x[(parent(x) \Rightarrow (person(x) \wedge \exists y(person(y) \wedge child(y, x))))$$
$$\wedge((person(x) \wedge \exists y(person(y) \wedge child(y, x))) \Rightarrow parent(x))]$$

To:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge \exists y(person(y) \wedge child(y, x))))$$
$$\wedge(\neg(person(x) \wedge \exists y(person(y) \wedge child(y, x))) \vee parent(x))]$$

Translation Step 3

Translate to *miniscope* form using

$$\models \neg(A \wedge B) \Leftrightarrow (\neg A \vee \neg B) \qquad \models \neg(A \vee B) \Leftrightarrow (\neg A \wedge \neg B)$$
$$\models \neg\forall x A \Leftrightarrow \exists x \neg A \qquad \models \neg\exists x A \Leftrightarrow \forall x \neg A$$

From:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge \exists y(person(y) \wedge child(y,x))))$$
$$\quad \wedge(\neg(person(x) \wedge \exists y(person(y) \wedge child(y,x))) \vee parent(x))]$$

To:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge \exists y(person(y) \wedge child(y,x))))$$
$$\quad \wedge((\neg person(x) \vee \forall y(\neg person(y) \vee \neg child(y,x))) \vee parent(x))]$$

Translation Step 4

Rename apart: If any two quantifiers bind the same variable, rename all occurrences of one of them.

From:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge \exists y(person(y) \wedge child(y,x))))$$
$$\wedge((\neg person(x) \vee \forall y(\neg person(y) \vee \neg child(y,x))) \vee parent(x))]$$

To:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge \exists y(person(y) \wedge child(y,x))))$$
$$\wedge((\neg person(x) \vee \forall z(\neg person(z) \vee \neg child(z,x))) \vee parent(x))]$$

Translation Step 5

Skolemize

From:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge \exists y(person(y) \wedge child(y,x))))$$
$$\wedge((\neg person(x) \vee \forall z(\neg person(z) \vee \neg child(z,x))) \vee parent(x))]$$

To:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge (person(f(x)) \wedge child(f(x),x))))$$
$$\wedge((\neg person(x) \vee \forall z(\neg person(z) \vee \neg child(z,x))) \vee parent(x))]$$

Translation Step 6

Discard all occurrences of "$\forall x$" for any variable x.

From:

$$\forall x[(\neg parent(x) \vee (person(x) \wedge (person(f(x)) \wedge child(f(x), x))))$$
$$\wedge((\neg person(x) \vee \forall z(\neg person(z) \vee \neg child(z, x))) \vee parent(x))]$$

To:

$$[(\neg parent(x) \vee (person(x) \wedge (person(f(x)) \wedge child(f(x), x))))$$
$$\wedge((\neg person(x) \vee (\neg person(z) \vee \neg child(z, x))) \vee parent(x))]$$

Translation Step 7

CNF: Translate into Conjunctive Normal Form, using

$$\models (A \vee (B \wedge C)) \Leftrightarrow ((A \vee B) \wedge (A \vee C))$$
$$\models (A \wedge (B \vee C)) \Leftrightarrow ((A \wedge B) \vee (A \wedge C))$$

From:

$$[(\neg parent(x) \vee (person(x) \wedge (person(f(x)) \wedge child(f(x), x))))$$
$$\wedge((\neg person(x) \vee (\neg person(z) \vee \neg child(z, x))) \vee parent(x))]$$

To:

$$[((\neg parent(x) \vee person(x))$$
$$\wedge((\neg parent(x) \vee person(f(x)))$$
$$\wedge(\neg parent(x) \vee child(f(x), x)))))$$
$$\wedge((\neg person(x) \vee (\neg person(z) \vee \neg child(z, x))) \vee parent(x))]$$

Translation Step 8

Discard extra parentheses using the associativity and commutativity of $\wedge$ and $\vee$.

From:

$$[((\neg parent(x) \vee person(x))$$
$$\wedge((\neg parent(x) \vee person(f(x)))$$
$$\wedge(\neg parent(x) \vee child(f(x), x))))$$
$$\wedge((\neg person(x) \vee (\neg person(z) \vee \neg child(z, x))) \vee parent(x))]$$

To:

$$[(\neg parent(x) \vee person(x))$$
$$\wedge(\neg parent(x) \vee person(f(x)))$$
$$\wedge(\neg parent(x) \vee child(f(x), x))$$
$$\wedge(\neg person(x) \vee \neg person(z) \vee \neg child(z, x) \vee parent(x))]$$

Translation Step 9

Turn each disjunction into a clause,
and the conjunction into a set of clauses.

From:

$$[(\neg parent(x) \lor person(x))$$
$$\land(\neg parent(x) \lor person(f(x)))$$
$$\land(\neg parent(x) \lor child(f(x), x))$$
$$\land(\neg person(x) \lor \neg person(z) \lor \neg child(z, x) \lor parent(x))]$$

To:

$$\{\{\neg parent(x), person(x)\},$$
$$\{\neg parent(x), person(f(x))\},$$
$$\{\neg parent(x), child(f(x), x)\},$$
$$\{\neg person(x), \neg person(z), \neg child(z, x), parent(x)\}\}$$

Translation Step 10

Rename the clauses apart
so that no variable occurs in more than one clause.

From:

$$\begin{aligned}
&\{\{\neg parent(x), person(x)\},\\
&\ \{\neg parent(x), person(f(x))\},\\
&\ \{\neg parent(x), child(f(x), x)\},\\
&\ \{\neg person(x), \neg person(z), \neg child(z, x), parent(x)\}\}
\end{aligned}$$

To:

$$\begin{aligned}
&\{\{\neg parent(x_1), person(x_1)\},\\
&\ \{\neg parent(x_2), person(f(x_2))\},\\
&\ \{\neg parent(x_3), child(f(x_3), x_3)\},\\
&\ \{\neg person(x_4), \neg person(z_4), \neg child(z_4, x_4), parent(x_4)\}\}
\end{aligned}$$

Use of Translation

$$A_1, \ldots, A_n \models B$$

iff

The translation of $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ into a set of clauses is contradictory.

Implemented Example

```
(prove
 '((all x ((parent x)
          <=> ((person x)
               & (exists y ((person y) & (child y x))))))
   (person John) (person Betty) (child John Betty))
 '(parent Betty))

 1 ((PERSON JOHN))                         Assumption
 2 ((PERSON BETTY))                        Assumption
 3 ((CHILD JOHN BETTY))                    Assumption
 4 ((~ (PARENT ?3)) (PERSON ?3))           Assumption
 5 ((~ (PARENT ?4)) (PERSON (S2 ?4)))      Assumption
 6 ((~ (PARENT ?5)) (CHILD (S2 ?5) ?5))    Assumption
 7 ((~ (PERSON ?6)) (~ (PERSON ?7))
    (~ (CHILD ?7 ?6)) (PARENT ?6))         Assumption
 8 ((~ (PARENT BETTY)))                    From Query
```

Resolution Steps

```
 1 ((PERSON JOHN))           Assumption
 2 ((PERSON BETTY))          Assumption
 3 ((CHILD JOHN BETTY))      Assumption
 7 ((~ (PERSON ?6))
    (~ (PERSON ?7))
    (~ (CHILD ?7 ?6))
    (PARENT ?6))             Assumption
 8 ((~ (PARENT BETTY)))      From Query
 9 ((~ (PERSON BETTY))
    (~ (PERSON ?8))
    (~ (CHILD ?8 BETTY))) R  8  7 ((?6 . BETTY))
13 ((~ (PERSON JOHN))
    (~ (PERSON BETTY)))      R  9  3 ((?8 . JOHN))
18  ((~ (PERSON JOHN)))      R 13  2 NIL
21  NIL                      R 18  1 NIL
QED
```

Connections

Modus Ponens

$$\frac{A, A \Rightarrow B}{B}$$

Resolution

$$\frac{\{A\}, \{\neg A, B\}}{\{B\}}$$

Modus Tollens

$$\frac{A \Rightarrow B, \neg B}{\neg A}$$

Resolution

$$\frac{\{\neg A, B\}, \{\neg B\}}{\{\neg A\}}$$

Disjunctive Syllogism

$$\frac{A \vee B, \neg A}{B}$$

Resolution

$$\frac{\{A, B\}, \{\neg A\}}{\{B\}}$$

Chaining

$$\frac{A \Rightarrow B, B \Rightarrow C}{A \Rightarrow C}$$

Resolution

$$\frac{\{\neg A, B\}, \{\neg B, C\}}{\{\neg A, C\}}$$

More Connections

Clause
$\{\neg A_1, \ldots, \neg A_n, C_1, \ldots, C_m\}$

Rule
$(A_1 \wedge \cdots \wedge A_n) \Rightarrow (C_1 \vee \cdots \vee C_m)$

Horn Clause
$\{\neg A_1, \ldots, \neg A_n, C\}$

Rule
$(A_1 \wedge \cdots \wedge A_n) \Rightarrow C$

Prolog Clause
C :- $A_1, \ldots, A_n$

Set of Support

Back-chaining

3.5 Asking Wh Questions

Given

$$\forall x[parent(x) \Leftrightarrow (person(x) \wedge \exists y(person(y) \wedge child(y,x)))]$$
$$person(John)$$
$$person(Betty)$$
$$child(John, Betty)$$

Ask: "Who is a parent?"

Answer via constructive proof of $\exists x \; parent(x)$.

Try to Answer Wh Question

```
(prove
 '((all x ((parent x)
          <=> ((person x)
               & (exists y ((person y) & (child y x))))))
   (person John) (person Betty) (child John Betty))
 '(exists x (parent x)))

 1 ((PERSON JOHN))                              Assumption
 2 ((PERSON BETTY))                             Assumption
 3 ((CHILD JOHN BETTY))                         Assumption
 4 ((~ (PARENT ?3)) (PERSON ?3))                Assumption
 5 ((~ (PARENT ?4)) (PERSON (S2 ?4)))           Assumption
 6 ((~ (PARENT ?5)) (CHILD (S2 ?5) ?5))         Assumption
 7 ((~ (PERSON ?6)) (~ (PERSON ?7))
   (~ (CHILD ?7 ?6)) (PARENT ?6))               Assumption
 8 ((~ (PARENT ?9)))                            From Query
```

Resolution Steps

```
 9 ((~ (PERSON ?10))
    (~ (PERSON ?11))
    (~ (CHILD ?11 ?10)))  R  8  7 ((?9 . ?6))

14 ((~ (PERSON JOHN))
    (~ (PERSON BETTY)))   R  9  3 ((?10 . BETTY)
                                   (?11 . JOHN))

16 ((~ (PERSON JOHN)))    R 14  2 NIL

17 NIL                    R 16  1 NIL

QED
```

The ANSWER Predicate

Instead of query $\exists x_1 \cdots \exists x_n P(x_1, \ldots, x_n)$,

and refutation resolution with $\{\neg P(x_1, \ldots, x_n)\}$

until $\{\}$,

use $\forall x_1 \cdots \forall x_n (P(x_1, \ldots, x_n) \Rightarrow ANSWER(P(x_1, \ldots, x_n)))$

and do direct resolution with

$$\{\neg P(x_1, \ldots, x_n), ANSWER(P(x_1, \ldots, x_n))\}$$

until $\{(ANSWER \ldots) \cdots (ANSWER \ldots)\}$.

Using the ANSWER Predicate

```
(prove
 '((all x ((parent x)
          <=> ((person x)
               & (exists y ((person y) & (child y x))))))
   (person John) (person Betty) (child John Betty))
 '(exists x (parent x)))

 1  ((PERSON JOHN))                               Assumption
 2  ((PERSON BETTY))                              Assumption
 3  ((CHILD JOHN BETTY))                          Assumption
 4  ((~ (PARENT ?3)) (PERSON ?3))                 Assumption
 5  ((~ (PARENT ?4)) (PERSON (S2 ?4)))            Assumption
 6  ((~ (PARENT ?5)) (CHILD (S2 ?5) ?5))          Assumption
 7  ((~ (PERSON ?6)) (~ (PERSON ?7))
     (~ (CHILD ?7 ?6)) (PARENT ?6))               Assumption
 8  ((~ (PARENT ?9)) (ANSWER (PARENT ?9))) From Query
```

Resolution Steps

```
 9  ((ANSWER (PARENT ?10))
     (~ (PERSON ?10))
     (~ (PERSON ?11))
     (~ (CHILD ?11 ?10)))      R  8  7 ((?9 . ?6))

14  ((~ (PERSON JOHN))
     (~ (PERSON BETTY))
     (ANSWER (PARENT BETTY)))  R  9  3 ((?10 . BETTY)
                                         (?11 . JOHN))

19  ((ANSWER (PARENT BETTY))
     (~ (PERSON JOHN)))        R 14  2 NIL

22  ((ANSWER (PARENT BETTY)))  R 19  1 NIL

QED
```

Disjunctive Answers

```
(prove '((on a b)(on b c)
         (red a) (green c)
         ((red b) or (green b)))
       '(exists x
         (exists y ((red x) & ((green y) & (on x y))))))

 1 ((ON A B))                                        Assumption
 2 ((ON B C))                                        Assumption
 3 ((RED A))                                         Assumption
 4 ((GREEN C))                                       Assumption
 5 ((RED B) (GREEN B))                               Assumption
 6 ((~ (RED ?2)) (~ (GREEN ?3))
    (~ (ON ?2 ?3))
    (ANSWER
     ((RED ?2) & ((GREEN ?3) & (ON ?2 ?3))))) From Query
```

Resolution Steps

```
 9 ((ANSWER
    ((RED A) & ((GREEN ?4) & (ON A ?4))))
   (~ (ON A ?4)) (~ (GREEN ?4)))         R  6  3 ((?2 . A))

10 ((ANSWER
    ((RED ?5) & ((GREEN C) & (ON ?5 C))))
   (~ (ON ?5 C)) (~ (RED ?5)))           R  6  4 ((?3 . C))

13 ((~ (RED B))
   (ANSWER
    ((RED B) & ((GREEN C) & (ON B C))))) R 10  2 ((?5 . B))

16 ((ANSWER
    ((RED B) & ((GREEN C) & (ON B C))))
   (GREEN B))                            R 13  5 NIL
```

Resolution Finished

```
20 ((~ (ON A B))
    (ANSWER
     ((RED A) & ((GREEN B) & (ON A B))))
    (ANSWER
     ((RED B) & ((GREEN C) & (ON B C))))) R  9 16 ((?4 . B))

22 ((ANSWER
     ((RED B) & ((GREEN C) & (ON B C))))
    (ANSWER
     ((RED A) & ((GREEN B) & (ON A B))))) R 20  1 NIL

QED
```

Multiple Clauses From Query

```
(prove '((on a b)(on b c)
         (red a) (green c)
         ((red b) or (green b)))
       '(exists x ((red x) or (green x))))

 1 ((ON A B))                              Assumption
 2 ((ON B C))                              Assumption
 3 ((RED A))                               Assumption
 4 ((GREEN C))                             Assumption
 5 ((RED B) (GREEN B))                     Assumption
 6 ((~ (RED ?1))
    (ANSWER ((RED ?1) OR (GREEN ?1))))    From Query
 7 ((~ (GREEN ?2))
    (ANSWER ((RED ?2) OR (GREEN ?2))))    From Query
 8 ((ANSWER ((RED A) OR (GREEN A))))      R  6  3 ((?1 . A))
QED
```

Wh Questions in Rule-Based Systems

Given rule $\forall\overline{x}(A(\overline{x}) \Rightarrow C(\overline{x}))$

Ask $C(\overline{y})$?

Backchain to subgoal $A(\overline{x})\mu$, where μ is the mgu of $C(\overline{x}))$ and $C(\overline{y}))$

Moral: Unification is generally needed in backward chaining systems.

Unification is correct pattern matching when both structures have variables.

Forward Chaining & Unification

Forward chaining generally matches a ground fact with rule antecedents.

Forward chaining does not generally require unification.

4 Summary

- **Logic** is the study of correct reasoning.

- There are many systems of logic (logics).

Parts of a Logic

- Syntax
- Semantics
  - Intensional: Formalizing a Domain
  - Extensional: Specifying a Situation
- Syntactic Inference Rules
- Semantic Inference Method

Soundness is the essence of "correct reasoning."

Relations Among Inference Methods

Syntax	**Derivation**		**Theoremhood**
	$A_1, \ldots, A_n \vdash P$	$\Leftrightarrow$	$\vdash A_1 \wedge \ldots \wedge A_n P$
	$\Downarrow\Uparrow$		$\Downarrow\Uparrow$
	$A_1, \ldots, A_n \models P$	$\Leftrightarrow$	$\models A_1 \wedge \ldots \wedge A_n P$
Semantics	**Logical Implication**		**Validity**

Inference Methods

- Direct, syntactic methods apply rules of inference to axioms (logical or non-logical).

- Refutation, semantic methods attempt to
  - fail at attempt to find a situation that satisfies the negation of the query.
  - or show the negation of the query is contradictory. (There is no situation that satisfies it.)

- Resolution (plus Factoring) is a complete refutation semantic inference method.

Unification

- Unification is a least-commitment method of choosing a substitution for Universal Instantiation ($\forall E$).

- Unification is correct pattern matching when both structures have variables.

- Unification is generally needed in backward chaining systems.

Effect of Functions

- Inclusion of functions makes inference hard (semi-decidable).
- Existential quantifiers lead to functions (Skolem functions).

AI-Logic Connections

AI	Logic
rules or domain rules	non-atomic assumptions or non-logical axioms
inference engine procedures	rules of inference
knowledge base	proof

Relevance of Logic

Any system that consists of

- a collection of symbol structures

- a set of procedures for adding new structures to that collection based on what's already in there

is a logic.

Do the symbol structures have a consistent semantics?

Are the procedures sound relative to that semantics?