
Logic

David Suendermann

<http://suendermann.com>

**Baden-Wuerttemberg Cooperative State University
Stuttgart, Germany**

- **The most up-to-date version of this document as well as auxiliary material can be found online at**

<http://suendermann.com>

- **propositional logic**
- **first-order logic**
- **Prolog**

Outline

- **propositional logic**
- **first-order logic**
- **Prolog**

- **Propositional logic** (aka propositional calculus) is a system of formulas representing **propositions**.
- We are using the following **logical operators** (aka logical connectives):
 - \neg (not)
 - \wedge (and)
 - \vee (or)
 - \rightarrow (if ... then)
 - \leftrightarrow (if and only if (aka iff))

- In order to build valid (well-formulated) formulas (WFFs) in propositional logic, we start with a set of **propositional variables** $P = \{p_1, \dots, p_N\}$.
- Given P , we can derive the set of WWFs F inductively as follows:
 1. $1 \in F$ (true)
 2. $0 \in F$ (false)
 3. if $p \in P$ then $p \in F$ (every variable is a WFF)
 4. if $f \in F$ then $\neg f \in F$ (the negation of a WFF is a WFF)
 5. if $f, g \in F$ then
 - a) $(f \wedge g) \in F$
 - b) $(f \vee g) \in F$
 - c) $(f \rightarrow g) \in F$
 - d) $(f \leftrightarrow g) \in F$

- Assume the following variables

$$P = \{p, q, r\}. \quad (1)$$

- These are WWFs:

p

$(p \wedge q)$

$((\neg p \rightarrow q) \vee (q \rightarrow \neg r))$

these are not:

p^{-1}

$(p \leftrightarrow q)$

$\rightarrow q$

- Outermost parenthesis can be dropped:

$$(p \wedge q) \Leftrightarrow p \wedge q \quad (2)$$

- Consider the following precedences:

operator	precedence
\neg	1 (strongest)
\wedge	2
\vee	3
\rightarrow	4
\leftrightarrow	5 (weakest)

- E.g., we have:

$$(p \wedge q) \rightarrow (q \vee r) \Leftrightarrow p \wedge q \rightarrow q \vee r \quad (3)$$

- Assume operators of the same precedence to be left-associative:

$$(p \rightarrow q) \rightarrow r \Leftrightarrow p \rightarrow q \rightarrow r \quad (4)$$

- **Set parentheses to indicate the order of evaluation:**

$$p \leftrightarrow q \vee r \wedge s \rightarrow t \wedge \neg u \wedge v \leftrightarrow w \quad (5)$$

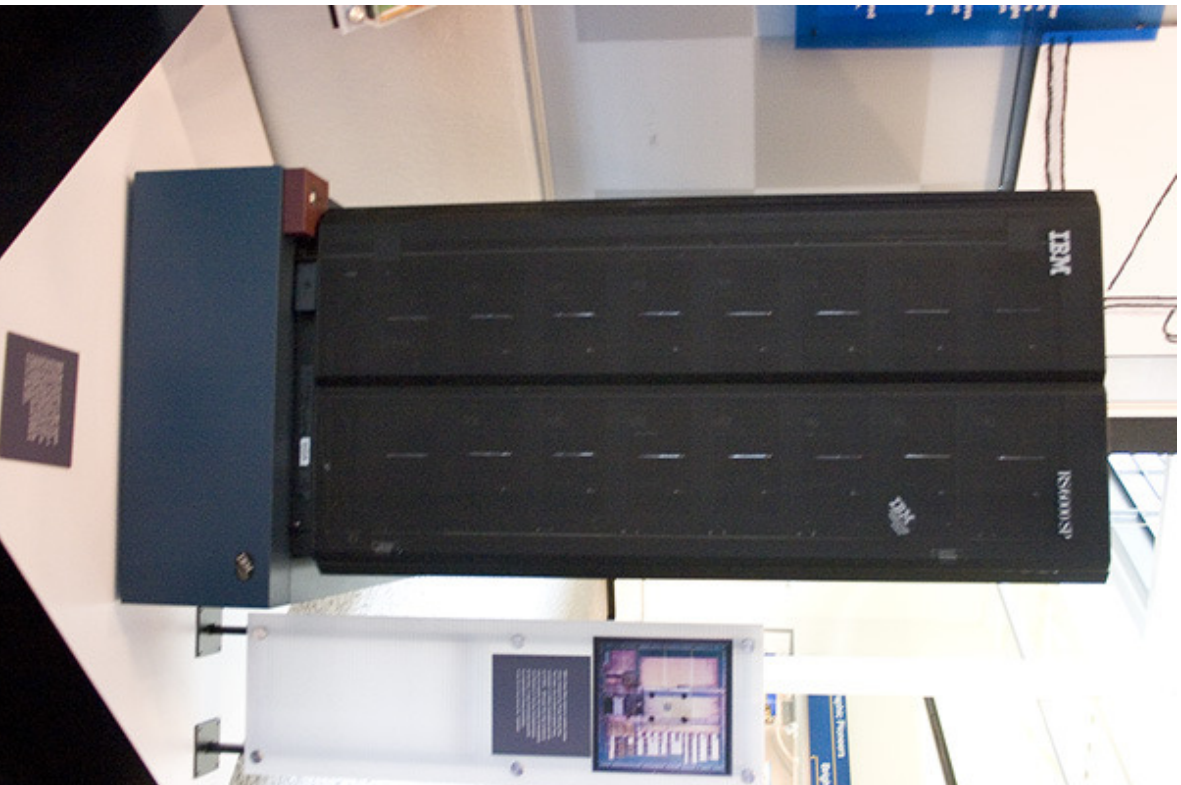
- **Drop as many parentheses as possible:**

$$((\neg\neg\neg(((p \leftrightarrow q) \vee r) \vee s) \vee t) \leftrightarrow u) \wedge v) \wedge w) \wedge x) \quad (6)$$

- **Design and analysis of digital circuits**
 - Several million logical gates (physical realizations of logical connectives) are implemented in nowadays' microprocessors.
 - A circuit comparison algorithm checks whether the propositional formulas representing two circuits are equivalent.
- **Planning**
 - In many planning tasks (such as in logistics, train or airline scheduling), multiple compulsory restrictions apply.
 - These restrictions can often be expressed in terms of propositional logic.
 - Optimal solutions may be determined by means of logical resolution techniques (we will learn about this later in this lecture).

- **Computer-assisted proof**
 - Often, proofs of conjectures in a mathematical discipline may be very complex (100s of pages).
 - If the discipline's axioms are given in form of logical formulas (the knowledge base), the (in)validity of conjectures may be proven by resolution.
 - This approach allows for proofs of a complexity humans are not able to handle (1,000,000s of pages).
- **Game theory**
 - Many one-, two-, or multi-player games can be expressed in terms of formulas of propositional logic.
 - Examples include chess, the 8-puzzle, or the 8-queens puzzle.
 - Again, resolution may be applied to solve these puzzles or derive “optimal” solutions.

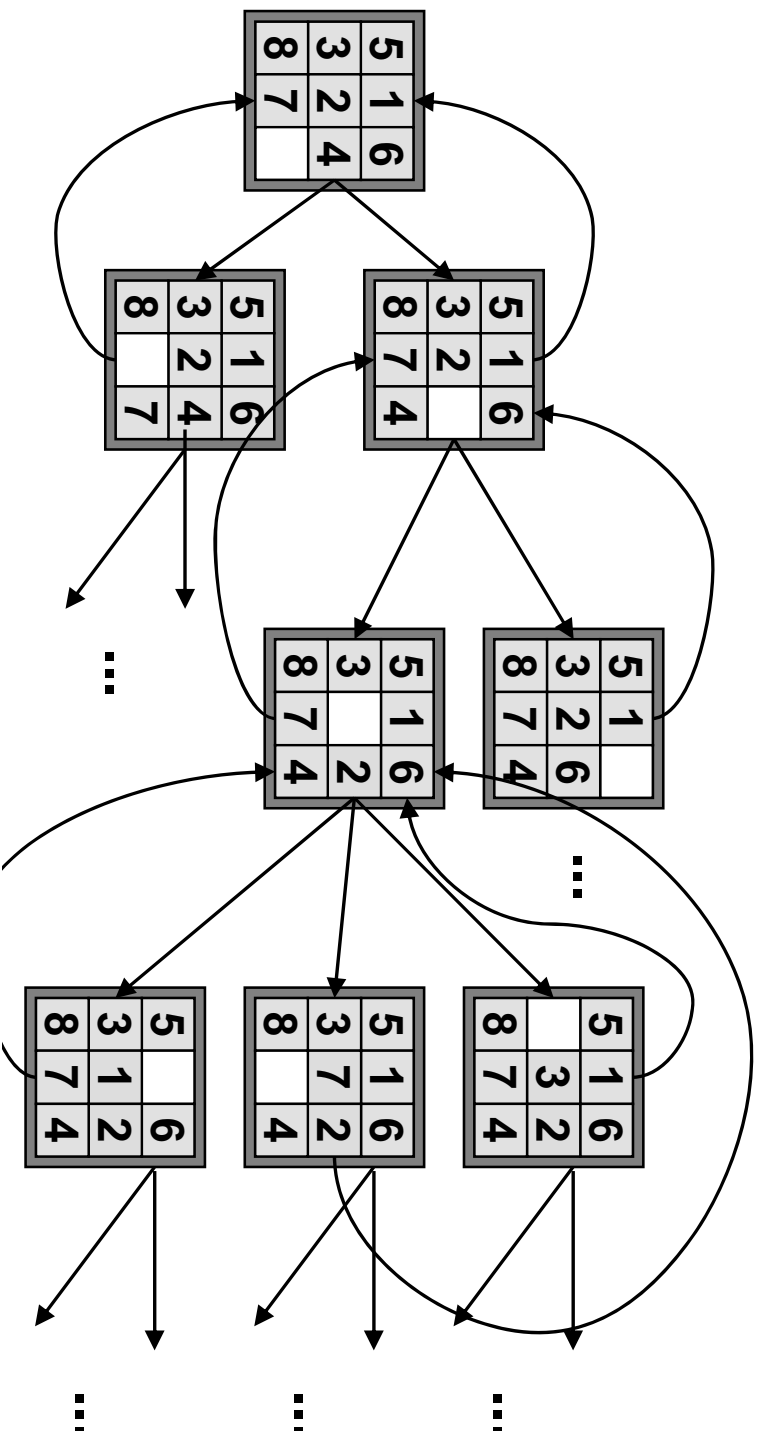
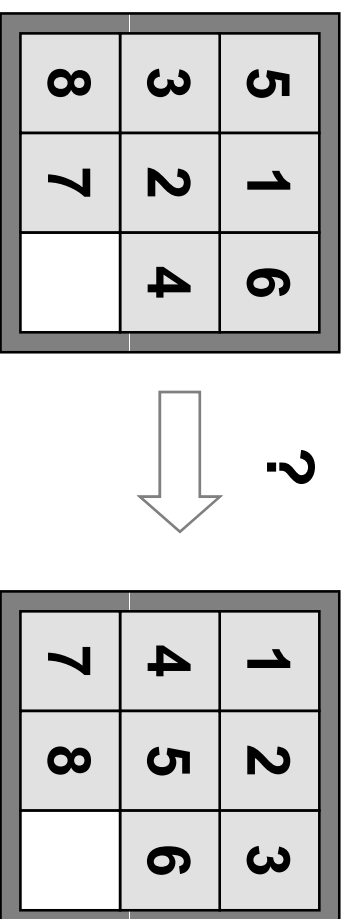
Deep Blue



- chess-playing computer by IBM
- On May 11, 1997, Deep Blue won a six-game match against Garry Kasparov.
- based on **brute-force** computing power (30 nodes with 480 VLSI chess chips)
- written in C under AIX
- The **evaluation function** contained multiple parameters tuned on 700,000 grandmaster games.

- pic:
 - source: <http://flickr.com/photos/22453761@N00/592436598/>
 - author: James the photographer
 - license: Creative Commons Attribution 2.0 Generic

8-puzzle



8 queens puzzle

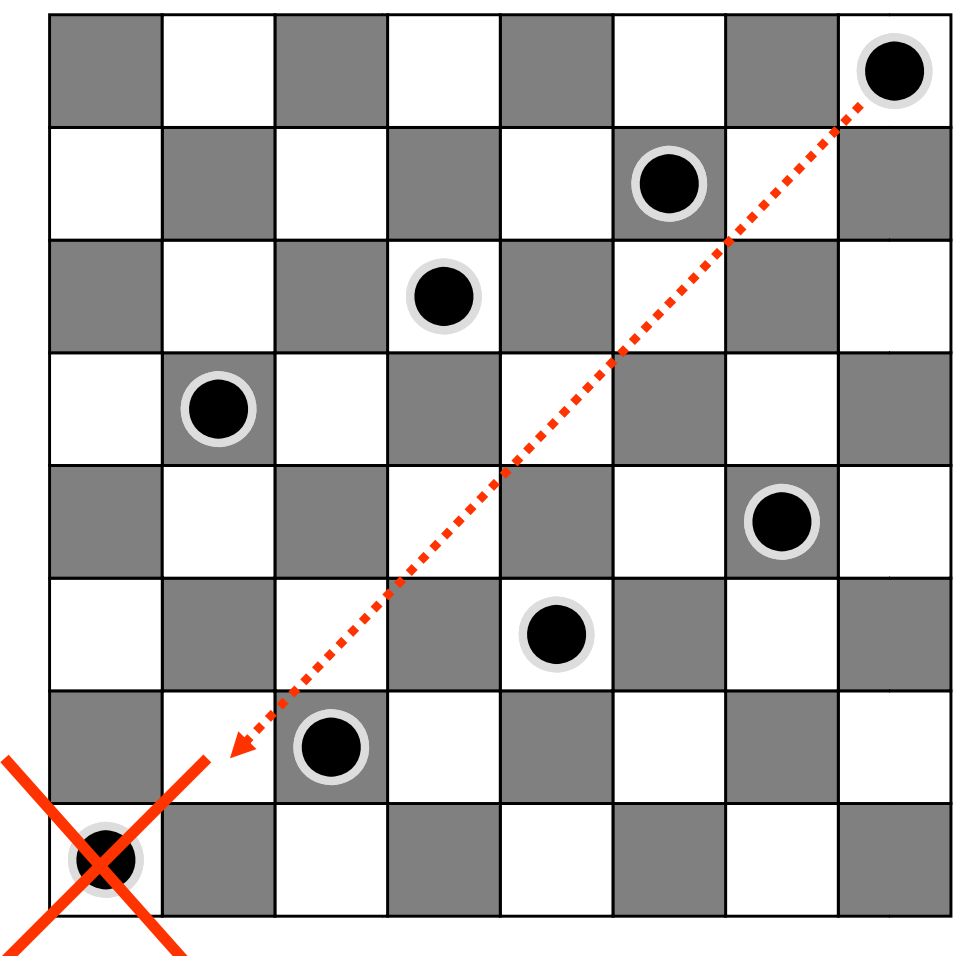
- Place 8 chess queens on a chessboard that no two queens attack each other.

- There are

$$\binom{64}{8} = 4,426,165,368$$

possible arrangements.

- But only 92 solutions.



- So far, we have investigated the **syntax** (structure) of propositional formulas.
- Now, we want to look at the **semantics**, that is the meaning, or truth, of propositional formulas.
- In general, without further knowledge about the variables of a formula, we cannot tell whether a formula is true or false.

- E.g., without knowing the truth value of p and q , the formula

$$p \vee q \tag{7}$$

may be true or false.

- To that end, we introduce the **valuation**, or **interpretation** of a formula as the function

$$I : F \rightarrow \mathbb{B} \quad \text{with} \quad \mathbb{B} = \{0, 1\}. \tag{8}$$

- We define the valuation I inductively as follows:
 1. $I(1) = 1$
 2. $I(0) = 0$
 3. $I(p) \in \mathbb{B}$ for all $p \in P$
(values for all $p \in P$ appearing in the formula have to be provided)
 4. $I(\neg f) = I_{\neg}(I(f))$ for all $f \in F$
 5. $I(f \wedge g) = I_{\wedge}(I(f), I(g))$ for all $f, g \in F$
 6. $I(f \vee g) = I_{\vee}(I(f), I(g))$ for all $f, g \in F$
 7. $I(f \rightarrow g) = I_{\rightarrow}(I(f), I(g))$ for all $f, g \in F$
 8. $I(f \leftrightarrow g) = I_{\leftrightarrow}(I(f), I(g))$ for all $f, g \in F$

Valuation of logical connectives

$$I_{\neg}(p): \quad p$$

0 1

1	0
---	---

$$I_{\wedge}(p, q): \quad p$$

0 1

0	0	0
1	0	1

$$I_{\vee}(p, q): \quad p$$

0 1

0	0	1
1	1	1

$$I_{\rightarrow}(p, q): \quad p$$

0 1

0	1	0
1	1	1

$$I_{\leftrightarrow}(p, q): \quad p$$

0 1

0	1	0
1	0	1

- We are given the formula

$$f := p \rightarrow q \rightarrow (\neg p \rightarrow q \rightarrow q). \quad (9)$$

- Now, we want to evaluate f for all possible interpretations of p and q .

- A handy way to do so is to use a **truth table**:

p	q	$g := \neg p$	$h := g \rightarrow q$	$i := h \rightarrow q$	$j := p \rightarrow q$	$j \rightarrow i$
0	0	1	1	0	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	1
1	1	0	1	1	1	1

with

$$\underbrace{p \rightarrow q}_{j} \rightarrow \underbrace{\left(\underbrace{\neg p}_{g} \rightarrow q \right)}_h \rightarrow \underbrace{q \rightarrow q}_i. \quad (10)$$

- Inspector Watson is called to a jewelry store that has been subject to a robbery where three subjects, Austin, Brian, and Colin, were arrested.

- After evaluation of all facts, this is known:

1. At least one of the subjects is guilty:

$$f_1 := a \vee b \vee c. \quad (17)$$

2. If Austin is guilty he had exactly one accomplice:

$$f_2 := a \rightarrow b \wedge \neg c \vee \neg b \wedge c. \quad (18)$$

3. If Brian is innocent, so is Colin:

$$f_3 := \neg b \rightarrow \neg c. \quad (19)$$

4. If exactly two subjects are guilty, Colin is one of them. Hence, out of three possible pairs of subjects, there is only one impossible:

$$f_4 := \neg(a \wedge b \wedge \neg c). \quad (20)$$

5. If Colin is innocent then Austin is guilty:

$$f_5 := \neg c \rightarrow a. \quad (21)$$

- **Exercise: Who are the culprits? (Hint: conjunctively combine f_1, \dots, f_5)**

- We have seen that Formula 9 is true for every propositional valuation.
- Such a formula is called **tautology** (Ludwig Wittgenstein, 1921).
- Iff f is a tautology this is also denoted as

$$\models f. \quad (22)$$

- **Wittgenstein on logic language and the mystery of the world:**

http://www.youtube.com/watch?v=Pv68v_reEQM

- **Examples:**

1. $\models p \vee \neg p$
2. $\models p \rightarrow p$
3. $\models p \wedge q \rightarrow p$
4. $\models p \rightarrow p \vee q$
5. $\models p \rightarrow 0 \leftrightarrow \neg p$
6. $\models p \wedge q \leftrightarrow q \wedge p$

- **validity:** f is valid $\Leftrightarrow f$ is a tautology, i.e., all interpretations make f true.
- **satisfiability:** At least one interpretation makes f true.
- **unsatisfiability, contradiction:** All interpretations make f false.
- **contingency:** Interpretations of f are contingent upon the truth values of f 's atomic parts. I.e., contingent propositions are neither necessarily true nor necessarily false.

- Two formulas f and g are **equivalent** iff

$$\models f \leftrightarrow g. \quad (23)$$

- **Examples:**

- | | | | |
|----|---|---|--------------------------|
| 1) | $\models \neg 0 \leftrightarrow 1$ | $\models \neg 1 \leftrightarrow 0$ | |
| 2) | $\models p \vee \neg p \leftrightarrow 1$ | $\models p \wedge \neg p \leftrightarrow 0$ | tertium non datur |
| 3) | $\models p \vee 0 \leftrightarrow p$ | $\models p \wedge 1 \leftrightarrow p$ | neutral element |
| 4) | $\models p \vee 1 \leftrightarrow 1$ | $\models p \wedge 0 \leftrightarrow 0$ | |
| 5) | $\models p \wedge p \leftrightarrow p$ | $\models p \vee p \leftrightarrow p$ | idempotency |
| 6) | $\models p \wedge q \leftrightarrow q \wedge p$ | $\models p \vee q \leftrightarrow q \vee p$ | commutativity |
| 7) | $\models p \wedge (q \wedge r) \leftrightarrow (p \wedge q) \wedge r$ | $\models p \vee (q \vee r) \leftrightarrow (p \vee q) \vee r$ | associativity |
| 8) | $\models \neg \neg p \leftrightarrow p$ | | double-negation |
| 9) | $\models p \wedge (p \vee q) \leftrightarrow p$ | $\models p \vee p \wedge q \leftrightarrow p$ | absorption |

Equivalence (cont.)

- 10) $\models p \wedge (q \vee r) \leftrightarrow p \vee q \wedge r \leftrightarrow (p \vee q) \wedge (p \vee r)$ **distributivity**
- 11) $\models \neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$ $\models \neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$ **de Morgan's rules**
- 12) $\models (p \rightarrow q) \leftrightarrow \neg p \vee q$ **elimination of \rightarrow**
- 13) $\models (p \leftrightarrow q) \leftrightarrow (\neg p \vee q) \wedge (\neg q \vee p)$ **elimination of \leftrightarrow**

- We are given the formula

$$f := p \rightarrow q \rightarrow (\neg p \rightarrow q \rightarrow q). \quad (24)$$

Prove that f is a tautology using equivalence rules.

- Transformations such as the ones in the last exercise can be applied algorithmically turning an arbitrary formula into a **normalized** form.
- In order to show this, we need a couple of definitions.
- A propositional formula f is called a **literal** iff either of the following cases applies:
 1. $f = 0$ or $f = 1$.
 2. $f = p$ with $p \in P$ (**positive literal**)
 3. $f = \neg p$ with $p \in P$ (**negative literal**)
- The set of all literals is denoted as L .
- A propositional formula c is called a **clause** iff c has the form
$$c = l_1 \vee \dots \vee l_m$$
where $l_1, \dots, l_m \in L$.
- That is, a clause is a **disjunction** of literals.
- The set of all clauses is denoted as C .

- Due to the associativity, commutativity, and idempotency of the \vee connective, in a clause c , the order of literals is arbitrary, and repeated literals can be dropped.

- This is why we can interpret the literals of c as the elements of a **set**:
 $\{l_1, \dots, l_m\}$. (26)

- Due to the equivalence

$$\models l_1 \vee \dots \vee l_m \vee 0 \leftrightarrow l_1 \vee \dots \vee l_m, \quad (27)$$

we have

$$\models \{l_1, \dots, l_m, 0\} \leftrightarrow \{l_1, \dots, l_m\} \quad (28)$$

(dropping of 0 from a clause).

- A special case is

$$\models \{0\} \leftrightarrow \{\} \quad (29)$$

(the empty clause).

- A clause c is called **trivial**, i.e.

$$\models c \tag{30}$$

iff we have one of the following cases:

1. Due to the equivalence

$$\models l_1 \vee \dots \vee l_m \vee 1 \leftrightarrow 1 \tag{31}$$

we have

$$\models \{l_1, \dots, l_m, 1\} \leftrightarrow \{1\} \tag{32}$$

(clause contains 1).

2. Due to the equivalence

$$\models l_1 \vee \dots \vee l_m \vee p \vee \neg p \leftrightarrow l_1 \vee \dots \vee l_m \vee 1 \tag{33}$$

we have (using Item 1)

$$\models \{l_1, \dots, l_m, p, \neg p\} \leftrightarrow \{1\} \tag{34}$$

(clause contains **complementary literals**).

- A formula f is in **conjunctive normal form (CNF)** iff f has the form

$$f = c_1 \wedge \dots \wedge c_n \quad \text{with} \quad c_1, \dots, c_n \in C. \quad (35)$$

- Due to the associativity, commutativity, and idempotency of the \wedge connective, in a formula f , the order of clauses is arbitrary, and repeated clauses can be discarded.

- This is why we can interpret the clauses of f as the elements of a **set** which, in turn, are sets of literals:

$$\{c_1, \dots, c_n\}. \quad (36)$$

- **The formula**

$$(p \vee q \vee \neg r) \wedge (q \vee \neg r \vee p \vee q) \wedge (\neg r \vee p \vee \neg q) \quad (37)$$

is in CNF, and its set notation is

$$\{\{p, q, \neg r\}, \{p, \neg q, \neg r\}\}. \quad (38)$$

- Due to the equivalence

$$\models c_1 \wedge \dots \wedge c_m \wedge 1 \leftrightarrow c_1 \wedge \dots \wedge c_m, \quad (39)$$

we have

$$\models \{c_1, \dots, c_m, 1\} \leftrightarrow \{c_1, \dots, c_m\} \quad (40)$$

(dropping of 1 from a formula in CNF).

- A special case is

$$\models \{1\} \leftrightarrow \{\} \quad (41)$$

(empty CNF).

- That is, if $f = \{c_1, \dots, c_n\}$ the following cases are equivalent

a) $\models f$

b) $\models c_i$ for all $i \in \{1, \dots, n\}$

c) $c_i \leftrightarrow \{1\}$ for all $i \in \{1, \dots, n\}$

d) $f \leftrightarrow \{\}$

(tautology properties).

- This algorithm turns an arbitrary propositional formula f into CNF:

1. Eliminate \leftrightarrow by Equivalence 13:

$$\models (p \leftrightarrow q) \Leftrightarrow (\neg p \vee q) \wedge (\neg q \vee p) \quad (42)$$

2. Eliminate \rightarrow by Equivalence 12:

$$\models (p \rightarrow q) \leftrightarrow \neg p \vee q \quad (43)$$

3. Simplify the formula using the Equivalences 1, 8, 11:

a) $\models \neg 0 \leftrightarrow 1$

b) $\models \neg 1 \leftrightarrow 0$

c) $\models \neg \neg p \leftrightarrow p$

d) $\models \neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$

e) $\models \neg(p \vee q) \leftrightarrow \neg p \wedge \neg q$

Now, the connective \neg will only appear in front of propositional variables (the formula is in **negation normal form**).

4. Expand the formula using the following variants of the law of distributivity (Equivalence 10):

a) $\models p \vee q \wedge r \leftrightarrow (p \vee q) \wedge (p \vee r)$

b) $\models p \wedge q \vee r \leftrightarrow (p \wedge r) \vee (q \wedge r)$

This is to transform the formula into a conjunction of disjunctions.

5. Transform the formula into set notation.

- Let us demonstrate the algorithm using the formula

$$f ::= p \rightarrow q \rightarrow (\neg p \rightarrow \neg q). \quad (44)$$

1. (skipped)

$$2. \quad \models f \Leftrightarrow \neg p \vee q \rightarrow (\neg p \rightarrow \neg q)$$

$$\Leftrightarrow \neg(\neg p \vee q) \vee (\neg p \rightarrow \neg q)$$

$$\Leftrightarrow \neg(\neg p \vee q) \vee (\neg\neg p \vee \neg q) \quad (45)$$

$$3. \quad \models f \Leftrightarrow \neg(\neg p \vee q) \vee (p \vee \neg q)$$

$$\Leftrightarrow \neg\neg p \wedge \neg q \vee (p \vee \neg q)$$

$$\Leftrightarrow p \wedge \neg q \vee (p \vee \neg q) \quad (46)$$

$$4. \quad \models f \Leftrightarrow (p \vee (p \vee \neg q)) \wedge (\neg q \vee (p \vee \neg q)) \quad (47)$$

$$5. \quad \models f \Leftrightarrow \{\{p, \neg q\}\} \quad (48)$$

Conjunctive normal form: derivation from truth table

- In case there are not too many variables involved, the CNF can also be derived directly from the truth table.
- To demonstrate this, let us consider the above example

$$f := \underbrace{p \rightarrow q}_g \rightarrow \underbrace{(\neg p \rightarrow \neg q)}_h. \quad (49)$$

p	q	g	h	f
0	0	1	1	1
0	1	1	0	0
1	0	0	1	1
1	1	1	1	1

- From those rows where f evaluates to 0, we can derive the CNF as

$$\models f \Leftrightarrow \neg(\neg p \wedge q) \Leftrightarrow p \vee \neg q \quad (50)$$

or, alternatively, from rows turning 1 the **disjunctive normal form (DNF)**

$$\models f \Leftrightarrow \neg p \wedge \neg q \vee p \wedge \neg q \vee p \wedge q. \quad (51)$$

- Transform into CNF:

a) $f ::= p \rightarrow (q \leftrightarrow r)$

b) $g ::= p \rightarrow q \rightarrow r \leftrightarrow \neg s \vee t$

- Proofs are one of the main notions of logic.
- Given
 - a set of formulas $\{f_1, \dots, f_n\}$ (the knowledge base) and
 - a formula g (the conjecture),we ask whether g can be proven based on (or inferred from) the knowledge base.
- That is, we ask whether
$$f_1 \wedge \dots \wedge f_n \rightarrow g \tag{52}$$
is a tautology.
- One possibility to answer this question is to convert Formula 52 into CNF and to check whether **all its clauses are trivial**.

- Given the knowledge base

$$f_1 := p \rightarrow q$$

$$f_2 := q \rightarrow r$$

(53)

prove the conjecture

$$g := p \rightarrow r$$

(54)

- This is, we need to prove whether

$$\models (p \rightarrow q) \wedge (q \rightarrow r) \rightarrow (p \rightarrow r)$$

(55)

- This proof can be done using a truth table or by conversion into CNF.
- This specific rule is called **hypothetical syllogism**.

- Another way to prove a conjecture from a knowledge base is to use **inference rules**.
- An inference rule is a pair consisting of a set of premises $\{f_1, \dots, f_n\}$ and a conclusion g written as

$$\frac{f_1, \dots, f_n}{\therefore g}$$

read as

From f_1, \dots, f_n , we can infer that g .

Inference rules: examples

- **simplification**

$$\frac{p \wedge q}{\therefore p}$$

- **modus tollens**

$$\frac{\neg q, p \rightarrow q}{\therefore \neg p}$$

- **modus ponens**

$$\frac{p \rightarrow q, p}{\therefore q}$$

- **hypothetical syllogism**

$$\frac{p \rightarrow q, q \rightarrow r}{\therefore p \rightarrow r}$$

- **biconditional elimination**

$$\frac{p \leftrightarrow q, p \vee q}{\therefore p \wedge q}$$

- **resolution**

$$\frac{p \vee q, \neg p \vee r}{\therefore q \vee r}$$

- Returning to the set representation of CNF, a handy inference rule for clauses is the **cut rule**.
- Given a propositional variable p and two clauses c_1 and c_2 , the cut rule is:

$$\frac{\{p\} \cup c_1, \{\neg p\} \cup c_2}{\therefore c_1 \cup c_2}$$

- We transform the conventional representation of the cut rule into CNF:

$$\begin{aligned}
 f & := (p \vee c_1) \wedge (\neg p \vee c_2) \rightarrow c_1 \vee c_2 \\
 & \Leftrightarrow \neg((p \vee c_1) \wedge (\neg p \vee c_2)) \vee c_1 \vee c_2 \\
 & \Leftrightarrow \neg(p \vee c_1) \vee \neg(\neg p \vee c_2) \vee c_1 \vee c_2 \\
 & \Leftrightarrow \underbrace{\neg p \wedge \neg c_1 \vee p \wedge \neg c_2 \vee c_1 \vee c_2}_a \\
 & \Leftrightarrow (\neg p \vee a) \wedge (\neg c_1 \vee a) \\
 a & \Leftrightarrow \underbrace{(p \vee c_1 \vee c_2)}_b \wedge \underbrace{(\neg c_2 \vee c_1 \vee c_2)}_c \\
 f & \Leftrightarrow ((\neg p \vee b) \wedge (\neg p \vee c)) \wedge ((\neg c_1 \vee b) \wedge (\neg c_1 \vee c)) \\
 & \Leftrightarrow \{\{\neg p, p, c_1, c_2\}, \{\neg p, \neg c_2, c_1, c_2\}, \\
 & \quad \{\neg c_1, p, c_1, c_2\}, \{\neg c_1, \neg c_2, c_1, c_2\}\} \tag{56}
 \end{aligned}$$

- Since all clauses in f are trivial,

$$\models f. \tag{57}$$

- Using Equivalence 12 (elimination of \rightarrow), the following of our inference rule examples can be regarded as special cases of the cut rule:

- Modus ponens. With $c_1 := \{\}$ and $c_2 := \{q\}$, we have

$$\frac{\{p\} \cup \{\}, \{\neg p\} \cup \{q\}}{\therefore \{\} \cup \{q\}}$$

- Modus tollens (show this).
- Hypothetical syllogism (show this).
- Resolution.

- Given a set of clauses M (the assumptions; a knowledge base) and a single clause f , we define a relation

$$M \vdash f \quad (58)$$

read as

f is provable from M .

- This relation is inductively defined as

- a) M proves all its assumptions:

$$\text{If } f \in M \text{ then } M \vdash f. \quad (59)$$

- b) If two clauses $\{p\} \cup c_1$ and $\{\neg p\} \cup c_2$ are provable from M then the clause $c_1 \cup c_2$ is provable (applying the cut rule):

$$\text{If } M \vdash \{p\} \cup c_1 \text{ and } M \vdash \{\neg p\} \cup c_2 \text{ then } M \vdash c_1 \cup c_2. \quad (60)$$

• We want to show that

$$\{\{\neg p, q\}, \{\neg q, \neg p\}, \{\neg q, p\}, \{q, p\}\} \vdash \{0\}. \quad (61)$$

• This is one way to do so:

1. $\{\{\neg p, q\}, \{\neg q, \neg p\}\} \vdash \{\neg p\}$
2. $\{\{\neg q, p\}, \{q, p\}\} \vdash \{p\}$
3. $\{\{\neg p\}, \{p\}\} \vdash \{\}$

- We are given the following knowledge base

$$f_1 ::= p \rightarrow q$$

$$f_2 ::= q \rightarrow r$$

$$f_3 ::= r \rightarrow s$$

$$f_4 ::= s \rightarrow t$$

$$f_5 ::= t \rightarrow u$$

$$f_6 ::= u \rightarrow p$$

$$f_7 ::= p \vee q \vee r \vee s \vee t \vee u$$

$$f_8 ::= \neg p \vee \neg q \vee \neg r \vee \neg s \vee \neg t \vee \neg u \quad (62)$$

- Show that 0 can be inferred from the knowledge base.

- In multiple applications, it is necessary to determine a variable interpretation rendering all the clauses in a set C_0 true.
- That is, we want to determine whether C_0 is satisfiable and, if so, a **solution**, i.e., a satisfying interpretation.
- Consider the following obvious cases
 - 1) $C_1 = \{\{p\}, \{\neg q\}, \{r\}, \{\neg s\}, \{\neg t\}\}$ (satisfiable)
 - 2) $C_2 = \{\{\}, \{p\}, \{\neg q\}, \{r\}\}$ (unsatisfiable)
 - 3) $C_3 = \{\{p\}, \{\neg q\}, \{\neg p\}\}$ (unsatisfiable)

- A clause is called **unit clause** iff it contains one literal:

$$c = \{p\} \quad \text{or} \quad c = \{\neg p\} \quad \text{with} \quad p \in P. \quad (63)$$

- A set of clauses C_0 is called **trivial** in one of these cases:

- 1) C_0 contains the empty clause:

$$\{\} \in C_0. \quad (64)$$

That is, C_0 is unsatisfiable.

- 2) C_0 contains only unit clauses of different propositional variables:

$$\begin{aligned} \forall c \in C_0 : \exists p \in P : (c = \{p\}) \vee (c = \{\neg p\}) \quad \text{and} \\ \forall p \in P : \neg(\{p\} \in C_0 \wedge \{\neg p\} \in C_0) \end{aligned} \quad (65)$$

That is, C_0 is satisfiable as determined by C_0 's unit clauses.

The algorithm of Davis and Putnam: basic principles

- **The Davis and Putnam (DP) algorithm is to find a satisfying clause given a set of clauses by applying**
 - 1) cut rule**
 - 2) subsumption**
 - 3) case distinction**

- The DP algorithm uses a special case of the cut rule:

$$\frac{\{l_1, \dots, l_n, \neg l\}, \{l\}}{\therefore \{l_1, \dots, l_n\}}$$

reducing the original clause length by one.

- Generally, this operation is performed by the function

$$\text{cut} : 2^C \times L \rightarrow 2^C \tag{66}$$

defined as

$$\text{cut}(C_0, l) = \{c \setminus \{\neg l\} \mid c \in C_0 \wedge \neg l \in c\}. \tag{67}$$

- Assume the example

$$C_0 := \{\{p, q, \neg r\}, \{p\}\}. \quad (68)$$

As we have

$$\models p \rightarrow p \vee q \vee \neg r, \quad (69)$$

we can drop (subsume) $\{p, q, \neg r\}$ from C_0 .

- Formally, we introduce the function

$$\text{sub} : 2^C \times L \rightarrow 2^C \quad (70)$$

that subsumes all clauses in C_0 containing the unit clause $\{l\}$:

$$\text{sub}(C_0, l) = \{c \in C_0 \mid l \notin c\} \cup \{\{l\}\}. \quad (71)$$

- Since cut rule and subsumption have the same argument spaces they can be combined to a general reduction function.
- In doing so, consider that applying the cut rule returns only cut versions of those clauses that originally contained $\neg l$.
- Hence, we add all those clauses *not* containing $\neg l$ before applying subsumption.

- This yields

$$\text{red}(C_0, l) = \text{sub}(C_1 \cup C_2, l) \quad (72)$$

with

$$C_1 = \text{cut}(C_0, l) = \{e \setminus \{\neg l\} \mid e \in C_0 \wedge \neg l \in e\} \quad (73)$$

and

$$C_2 = \{e \in C_0 \mid \neg l \notin e\}. \quad (74)$$

- Assuming C_0 is **not trivial**, we know that applying subsumption to C_1 does not affect any of its clauses since they originally contained $\neg l$, i.e., they **do not** contain l .

- Hence, subsumption can only affect C_2 which is why we can write

$$\begin{aligned} \text{red}(C_0, l) &= \text{sub}(C_1 \cup C_2, l) \\ &= C_1 \cup \text{sub}(C_2, l) \\ &= \{e \setminus \{\neg l\} \mid e \in C_0 \wedge \neg l \in e\} \cup \\ &\quad \{e \in C_0 \mid l \notin e \wedge \neg l \notin e\} \cup \{\{l\}\}. \end{aligned} \tag{75}$$

- This principle is based on the following proposition:

C_0 is satisfiable iff $C_0 \cup \{\{p\}\}$ or $C_0 \cup \{\{\neg p\}\}$ are satisfiable.

Here, C_0 is a set of clauses; p is a propositional variable occurring in C_0 .

- This sounds like a trivial principle since if there is a variable interpretation satisfying C_0 it will include the variable p which must be either 0 or 1 in this specific interpretation.
- The case distinction principle is, however, essential to the DP algorithm as it injects unit clauses required for cut rule and subsumption.

- We are given a set of clauses C_0 and search for a propositional valuation satisfying C_0 .
- 1. Iteratively apply $\text{red}(C_0, l)$ for all $l \in \{p, \neg p \mid p \in P\}$. If C_0 turns out to be trivial, stop.
- 2. Otherwise, choose a variable p from C_0 (should not be a unit clause since those had been tried in 1.)
 - a) Iteratively apply $\text{red}(C_0 \cup \{\{p\}\}, l)$ for all $l \in \{p, \neg p \mid p \in P\}$. If C_0 is found to be satisfiable, stop.
 - b) Iteratively apply $\text{red}(C_0 \cup \{\{\neg p\}\}, l)$ for all $l \in \{p, \neg p \mid p \in P\}$. If C_0 is found to be satisfiable, stop.
 - c) Otherwise, C_0 is not satisfiable.

- We are given

$$C_0 := \{\{p, q, s\}, \{\neg p, r, \neg t\}, \{r, s\}, \{\neg r, q, \neg p\}, \{\neg s, p\}, \{\neg p, \neg q, s, \neg r\}, \{p, \neg q, s\}, \{\neg r, \neg s\}, \{\neg p, \neg s\}\} \quad (76)$$

1. (skipped)— C_0 contains no unit clauses)

2. pick p

- a) $C_1 := \text{red}(C_0 \cup \{\{p\}\}, p)$

$$= \{\{r, \neg t\}, \{r, s\}, \{\neg r, q\}, \{\neg q, s, \neg r\}, \{\neg r, \neg s\}, \{p\}\}$$

$$C_2 := \text{red}(C_1, \neg s)$$

$$= \{\{r, \neg t\}, \{r\}, \{\neg r, q\}, \{\neg q, \neg r\}, \{p\}\}$$

$$C_3 := \text{red}(C_2, r)$$

$$= \{\{r\}, \{q\}, \{\neg q\}, \{\neg s\}, \{p\}\}$$

$$C_4 := \text{red}(C_3, q)$$

$$= \{\{r\}, \{q\}, \{\}, \{\neg s\}, \{p\}\} \quad (\text{unsatisfiable}) \quad (77)$$

$$\text{b) } C_5 := \text{red}(C_0 \cup \{\{\neg p\}\}, \neg p)$$

$$= \{\{q, s\}, \{r, s\}, \{\neg s\}, \{\neg q, s\}, \{\neg r, \neg s\}, \{\neg p\}\}$$

$$C_6 := \text{red}(C_5, \neg s)$$

$$= \{\{q\}, \{r\}, \{\neg s\}, \{\neg q\}, \{\neg p\}\}$$

$$C_7 := \text{red}(C_6, q)$$

$$= \{\{q\}, \{r\}, \{\neg s\}, \{\}, \{\neg p\}\} \quad (\text{unsatisfiable})$$

(78)

c) C_0 is unsatisfiable (contradiction).

- Apply the DP algorithm to the provability exercise with
 - a) $C_0 := \{f_1, \dots, f_8\}$,
 - b) $C_0 := \{f_1, \dots, f_7\}$.

- In order to prove a conjecture based on a knowledge base, we can apply the **resolution technique**:
 1. All sentences in the knowledge base and the **negation** of the sentence to be proven (the **conjecture**) are conjunctively connected.
 2. The resulting sentence is transformed into CNF.
 3. If the **empty clause** can be derived after an application of the DP algorithm (or, alternatively, the provability technique), the original sentence is **unsatisfiable**, i.e., the conjecture follows from the knowledge base.

- Let us revisit the criminal case from earlier.

- Remember, we were given the following facts (our knowledge base):

$$f_1 ::= a \vee b \vee c$$

$$f_2 ::= a \rightarrow b \wedge \neg c \vee \neg b \wedge c$$

$$f_3 ::= \neg b \rightarrow \neg c$$

$$f_4 ::= \neg(a \wedge b \wedge \neg c)$$

$$f_5 ::= \neg c \rightarrow a$$

(79)

- Dr. Watson's gut feeling is that Brian and Colin are the culprits.
- Prove his conjecture using the resolution technique.

- **Using the DP algorithm, show that**
 - the 2 queens puzzle has no solution,
 - the 3 queens puzzle has no solution,
 - the 4 queens puzzle has a solution and which one it is.

- **propositional logic**
- **first-order logic**
- **Prolog**

- **Terms** denote objects.

- Terms are composed of **variables** or **function symbols**:

$$\text{father}(x), \quad \text{mother}(\text{isaac}), \quad x + 7. \quad (80)$$

with the variable x and the function symbols father, mother, isaac, +, 7

- Objects can be put into relation by **predicate symbols** producing **atomic formulas**:

$$\text{IsBrother}(\text{adam}, \text{father}(\text{brian})), \quad x + 7 < x \cdot 7, \quad n \in \mathbb{N}. \quad (81)$$

with the predicate symbols IsBrother, <, ∈

- Formulas can be combined using **logical connectives**:

$$\text{IsHuman}(x) \rightarrow \text{IsMortal}(x), \quad x > 1 \rightarrow x + 7 < x \cdot 7. \quad (82)$$

- Formulas can contain **quantifiers** defining the semantics (scope) of variables:

$$\forall x (\text{IsHuman}(x) \rightarrow \exists y (\text{equals}(y, \text{mother}(x)))). \quad (83)$$

- A **signature** is a quadruple defining a specific system of first-order logic:

$$\Sigma = \langle V, F, P, \text{arity} \rangle. \quad (84)$$

- The members of Σ are

- the set of **variables** V ,
- the set of **function symbols** F ,
- the set of **predicate symbols** P , and
- a function assigning an arity to all function and predicate symbols:

$$\text{arity} : F \cup P \rightarrow \mathbb{N}. \quad (85)$$

- Given a signature Σ , we define the set of Σ **terms** T_Σ inductively as:

1. $\forall x \in V : x \in T_\Sigma$.
2. If $f \in F$ and $n = \text{arity}(f)$ and $t_1, \dots, t_n \in T_\Sigma$ then
$$f(t_1, \dots, t_n) \in T_\Sigma. \quad (86)$$

- **Special case:** If $f \in F$ and $\text{arity}(f) = 0$ then we can write f instead of $f()$ (aka **constant**).

- We are given the signature $\Sigma_a = \langle V, F, P, \text{arity} \rangle$ with

$$V = \{x, y, z\}$$

$$F = \{0, 1, +, \cdot\}$$

$$P = \{=, \leq\}$$

$$\text{arity} = \{\langle 0, 0 \rangle \langle 1, 0 \rangle \langle +, 2 \rangle \langle \cdot, 2 \rangle \langle =, 2 \rangle \langle \leq, 2 \rangle\}$$

- Now, we can derive Σ_a terms as follows:
 1. $x, y, z \in T_{\Sigma_a}$
 2. $0, 1 \in T_{\Sigma_a}$ (0-ary function symbols)
 3. $+(1, x) \in T_{\Sigma_a}$
 4. $\cdot(+(1, x), y) \in T_{\Sigma_a}$
- In the following, we will use an **infix** notation for binary relation and function symbols.
- E.g., Term 4 would read

$$(1 + x) \cdot y \quad (87)$$

- Given the signature $\Sigma = \langle V, F, P, \text{arity} \rangle$, if $p \in P$ and $n = \text{arity}(p)$ and $t_1, \dots, t_n \in \mathcal{T}_\Sigma$ then
$$p(t_1, \dots, t_n) \in A_\Sigma,$$
(88)
the set of all **atomic formulas**.

- **Special case:** If $p \in P$ and $\text{arity}(p) = 0$ then we can write p instead of $p()$ (aka **propositional variable**).

- Continuing with our above example:

$$= (\cdot(+ (1, x), y), 0) \in A_{\Sigma_a}$$
(89)

or in infix notation

$$(1 + x) \cdot y = 0 \in A_{\Sigma_a}.$$
(90)

- Consider an example from calculus:

$$\int_0^x yzdz = \frac{1}{2}yx^2. \quad (91)$$

- Eq. 91 features the three variables x , y and t .

- Trying to substitute variables on both sides of the equation, e.g., x by y

$$\int_0^y yzdz = \frac{1}{2}yy^2 \quad (92)$$

or z by y

$$\int_0^x yydy = \frac{1}{3}x^3 \neq \frac{1}{2}xx^2 \quad (93)$$

shows that there can be two types of variables in formulas: free (x , y) and bound (z) ones.

- The variable z in this example is bound by the differential operator d .

- In first-order logic, variables are bound by the quantifiers \forall and \exists .
- Accordingly, in the following example

$$\forall x, y (P(x) \rightarrow Q(x, f(x), z)) \quad (94)$$

x and y are bound variables, and z is free.

- A formula with no free variables is called a **sentence**.

- Consider the following examples:
 - a) $\forall x(F(x) \rightarrow \exists y(G(y, z)))$
 - b) $\forall x(F(x) \rightarrow \exists y(G(x, y)))$
 - c) $\exists x(y + x \leq y)$
- Which variables are free, and which ones are bound?
- Which of these formulas are sentences?

- **The set of variables $\text{var}(t)$ occurring in a term t is inductively defined as**
 1. $\text{var}(x) := x$ for all $x \in V$.
 2. $\text{var}(f(t_1, \dots, t_n)) := \text{var}(t_1) \cup \dots \cup \text{var}(t_n)$.
- Given a signature Σ , in the following, we denote
 - the set of **formulas** as F_Σ ,
 - the set of all **bound variables** of a formula $f \in F_\Sigma$ as $\text{bound}(f)$, and
 - the set of all **free variables** of f as $\text{free}(f)$.
- These sets are inductively defined as
 1. $0, 1 \in F_\Sigma$ (truth values) and

$$\text{free}(0) = \text{free}(1) = \text{bound}(0) = \text{bound}(1) = \{\}$$
(95)
 2. $\forall f = p(t_1, \dots, t_n) \in A_\Sigma : f \in F_\Sigma$ (atomic formulas) and

$$\text{free}(f) = \text{var}(t_1) \cup \dots \cup \text{var}(t_n); \quad \text{bound}(f) = \{\}$$
(96)

3. $\forall f \in F_{\Sigma} : \neg f \in F_{\Sigma}$ (negation) and

$$\text{free}(\neg f) = \text{free}(f); \quad \text{bound}(\neg f) = \text{bound}(f) \quad (97)$$

4. $\forall f, g \in F_{\Sigma} | \text{free}(f) \cap \text{bound}(g) = \{\} \wedge \text{bound}(f) \cap \text{free}(g) = \{\} :$
 $(f \vee g) \in F_{\Sigma}$ (logical connectives) and

$$\text{free}(f \vee g) = \text{free}(f) \cup \text{free}(g); \quad \text{bound}(f \vee g) = \text{bound}(f) \cup \text{bound}(g) \quad (98)$$

applying to all logical connectives ($\vee, \wedge, \rightarrow, \leftrightarrow$)

5. $\forall f \in F_{\Sigma}, x \in V \setminus \text{bound}(f) : (\forall x(f)), (\exists x(f)) \in F_{\Sigma}$ (quantifiers) and

$$\text{free}(\forall x(f)) = \text{free}(\exists x(f)) = \text{free}(f) \setminus \{x\}; \quad (99)$$

$$\text{bound}(\forall x(f)) = \text{bound}(\exists x(f)) = \text{bound}(f) \cup \{x\} \quad (100)$$

- In literature you encounter expressions of the form

$$\forall x \in R : \exists n \in N : x < n. \quad (101)$$

- These abbreviations can be transformed into the formerly introduced terminology by

$$\forall x \in M : f \quad \stackrel{def}{\iff} \quad \forall x(x \in M \rightarrow f)$$

$$\exists x \in M : f \quad \stackrel{def}{\iff} \quad \exists x(x \in M \wedge f) \quad (102)$$

- Accordingly, Formula 101 can be written as

$$\forall x(x \in R \rightarrow \exists n(n \in N \wedge x < n)). \quad (103)$$

- Sequences of identical quantifiers can be abbreviated:

$$\forall x, y(f) \quad \stackrel{def}{\iff} \quad \forall x(\forall y(f)). \quad (104)$$

- Quantifiers have a higher precedence than logical connectives:

$$\forall x(f) \wedge g \quad \stackrel{def}{\iff} \quad (\forall x(f)) \wedge g. \quad (105)$$

- So far, we have learned how to create (well-formulated) formulas in first-order logic, i.e., we have dealt with **syntax**.
- The next step is the **semantics** of first-order logic.

- To that end, given a signature Σ , we introduce the notion of a **structure**

$$S = \langle U, J \rangle \tag{106}$$

with

1. the **universe** U , a non-empty set containing all the values that can occur when evaluating terms,
2. the **interpretation** J of all function and predicate symbols of Σ .

- Formally, J is defined as

1. Every function symbol $f \in F$ with the arity $n = \text{arity}(f)$ is mapped to an n -ary function

$$f^J : \underbrace{U \times \cdots \times U}_n \rightarrow U. \quad (107)$$

2. Every predicate symbol $p \in P$ with the arity $n = \text{arity}(f)$ is mapped to an n -ary relation

$$p^J \subseteq U^n. \quad (108)$$

3. If $= \in P$ then its interpretation should be natural, i.e.

$$=^J = \{ \langle u, v \rangle \mid u, v \in U \wedge u = v \}. \quad (109)$$

- Using Σ_a as in the above example, we define a structure

$$S_a = \langle U, J \rangle \tag{110}$$

as follows:

1. $U = \{a, b\}$
2. $0^J = a$
3. $1^J = b$
4. $+^J = \{\{\langle a, a \rangle, a \rangle, \langle \langle a, b \rangle, b \rangle, \langle \langle b, a \rangle, b \rangle, \langle \langle b, b \rangle, a \rangle\}$
5. $\cdot^J = \{\{\langle \langle a, a \rangle, a \rangle, \langle \langle a, b \rangle, a \rangle, \langle \langle b, a \rangle, a \rangle, \langle \langle b, b \rangle, b \rangle\}$
6. $=^J = \{\langle a, a \rangle, \langle b, b \rangle\}$
7. $\leq^J = \{\langle a, a \rangle, \langle a, b \rangle, \langle b, b \rangle\}$

- Given a signature Σ and a structure $S = \langle U, J \rangle$, we define a **variable assignment**

$$I : V \rightarrow U \tag{111}$$

assigning a value from the universe U to every variable in V .

- E.g., using the signature Σ_a and the structure S_a , a possible variable assignment is

$$I_a = \{ \langle x, a \rangle, \langle y, b \rangle, \langle z, a \rangle \}. \tag{112}$$

- Furthermore, we introduce the **variable replacement**

$$I[x_1/c_1] \cdots [x_n/c_n](y) = \begin{cases} c_1 & : y = x_1; \\ \vdots & \\ c_n & : y = x_n; \\ I(y) & : \text{otherwise} \end{cases} \quad (113)$$

with $x_1, \dots, x_n, y \in V$ and $c_1, \dots, c_n \in U$ replacing the value of a variable y by c_i if the variable happens to be identical to x_i .

- Using the above example, we have

$$I_a[y/a][z/b] = \{\langle x, a \rangle, \langle y, a \rangle, \langle z, b \rangle\}. \quad (114)$$

- Given a signature Σ , a structure $S = \langle U, J \rangle$, and a variable assignment I , for every term t , its value (written as $S(I, t)$) can be derived inductively as

1. $\forall x \in V : S(I, x) = I(x)$

2. If $f \in F$ and $n = \text{arity}(f)$ and $t_1, \dots, t_n \in \mathcal{T}_\Sigma$ then

$$S(I, f(t_1, \dots, t_n)) = f^J(S(I, t_1), \dots, S(I, t_n)). \quad (115)$$

- **Exercise:** Using the above signature Σ_a , structure S_a , and variable assignment I_a , what is the evaluation of Term 4:

$$S(I_a, (1 + x) \cdot y)? \quad (116)$$

- Given a signature Σ , a structure $S = \langle U, J \rangle$, and a variable assignment I , for every atomic formula $p(t_1, \dots, t_n)$, its value can be derived as

$$S(I, p(t_1, \dots, t_n)) = \begin{cases} 1 & : \langle S(I, t_1), \dots, S(I, t_n) \rangle \in p^J, \\ 0 & : \text{otherwise.} \end{cases} \quad (117)$$

- Exercise: Using the above signature Σ_a , structure S_a , and variable assignment I_a , what is the evaluation of Formula 90:

$$S(I, (1 + x) \cdot y = 0)? \quad (118)$$

- Given a signature Σ , a structure $S = \langle U, J \rangle$, and a variable assignment I , for every formula $f \in F_{\Sigma}$, its value can be derived inductively as
 1. $S(I, 0) = 0; S(I, 1) = 1$
 2. $S(I, \neg f) = I_{\neg}(S(I, f))$
 3. $S(I, f \vee g) = I_{\vee}(S(I, f), S(I, g))$
 4. $S(I, f \wedge g) = I_{\wedge}(S(I, f), S(I, g))$
 5. $S(I, f \rightarrow g) = I_{\rightarrow}(S(I, f), S(I, g))$
 6. $S(I, f \leftrightarrow g) = I_{\leftrightarrow}(S(I, f), S(I, g))$
 7. $S(I, \forall x(f)) = \begin{cases} 1 & : \forall c \in U : S(I[x/c], f) = 1 \\ 0 & : \text{otherwise} \end{cases}$
 8. $S(I, \exists x(f)) = \begin{cases} 1 & : \exists c \in U : S(I[x/c], f) = 1 \\ 0 & : \text{otherwise} \end{cases}$

- Using the above signature Σ_a , structure S_a , and variable assignment I_a , determine the evaluation of the following formulas:
 1. $\exists x((1 + x) \cdot y = 0)$,
 2. $\forall x((1 + x) \cdot y = 0)$,
 3. $\forall x, y((1 + x) \cdot y = 0)$,
 4. $\forall x \exists y((1 + x) \cdot y = 0)$,
 5. $\forall y \exists x((1 + x) \cdot y = 0 \rightarrow \forall z(x \cdot z = 0))$.

- If f is a formula resulting in

$$S(I, f) = 1 \tag{119}$$

for every possible variable assignment I , then f is **universally valid**

- Being the equivalent to a **tautology** in propositional logic, universal validity of a formula f is written as

$$\models f. \tag{120}$$

- If $\text{free}(f) = \{\}$, then $S(I, f)$ does not depend on I . In this case, f is called a **closed formula**.

- For closed formulas, we use an abbreviated terminology:

$$S(f) := S(I, f) \quad \text{if } \text{free}(f) = \{\}. \tag{121}$$

- Also, in the case that

$$S(f) = 1, \tag{122}$$

we say that the structure S is a **model for f** written as

$$S \models f. \tag{123}$$

- Also the notions of **equivalence** and **(un)satisfiability** are inherited from propositional logic.

- Two formulas f and g are **equivalent** iff

$$\models f \leftrightarrow g. \quad (124)$$

- A set of formulas $M \subseteq F_{\Sigma}$ is **satisfiable**, if there is a variable assignment I such that

$$\forall m \in M : S(I, m) = 1. \quad (125)$$

- Otherwise, M is called **unsatisfiable** written as

$$M \models 0. \quad (126)$$

- Our next goal is to define an algorithm that checks whether

$$M \models 0. \tag{127}$$

- In general, this question is **undecidable** (as shown later).

- We will, however, be able to define a **calculus** \vdash for which we have

$$M \vdash 0 \leftrightarrow M \models 0. \tag{128}$$

- This calculus will be based on a **semi-decision algorithm**.
- I.e., if indeed $M \models 0$, the algorithm will eventually discover this fact.
- If, however, M is satisfiable, the algorithm may run eternally.
- **Motivation of semi-decidability: Goldbach conjecture.**

- In order to define the calculus \vdash , we will transform formulas into **first-order clauses**.
- This transformation will make use of the following equivalences (in addition to the ones we know from propositional logic):
 21. $\models \neg \forall x(f) \leftrightarrow \exists x(\neg f)$
 22. $\models \neg \exists x(f) \leftrightarrow \forall x(\neg f)$
 23. $\models \forall x(f) \wedge \forall x(g) \leftrightarrow \forall x(f \wedge g)$
 24. $\models \exists x(f) \vee \exists x(g) \leftrightarrow \exists x(f \vee g)$
 25. $\models \forall x, y(f) \leftrightarrow \forall y, x(f)$
 26. $\models \exists x, y(f) \leftrightarrow \exists y, x(f)$
 27. If $x \in V \wedge x \notin \text{free}(f)$ then
 - a) $\models \forall x(f) \leftrightarrow f$
 - b) $\models \exists x(f) \leftrightarrow f$

28. If $x \in V \wedge x \notin \text{free}(g) \cup \text{bound}(g)$ then

- a) $\models \forall x(f) \vee g \leftrightarrow \forall x(f \vee g)$
- b) $\models \exists x(f) \wedge g \leftrightarrow \exists x(f \wedge g)$
- c) $\models g \vee \forall x(f) \leftrightarrow \forall x(g \vee f)$
- d) $\models g \wedge \exists x(f) \leftrightarrow \exists x(g \wedge f)$

- In certain situations, it is necessary to **rename bound variables**.

- If $f \in F_{\Sigma}$ and $x, y \in V$ then $f[x/y]$ is the formula which we obtain by replacing every occurrence of x by z .

- For example

$$(\forall u \exists v (P(u, v))) [u/z] = \forall z \exists v (P(z, v)). \quad (129)$$

- This leads us to our last equivalence:

29. If $x \in \text{bound}(f) \wedge y \notin \text{free}(f) \cup \text{bound}(f)$ then

$$\models f \leftrightarrow f[x/y]$$

- The above equivalences can be used to rewrite an arbitrary formula such that **all quantifiers appear at the beginning of the formula**.
- This representation is called **prenex normal form**.
- **Example 1:**

$$\begin{aligned} & \forall x(p(x)) \rightarrow \exists x(p(x)) \quad \text{(closed)} \\ \stackrel{12}{\iff} & \neg \forall x(p(x)) \vee \exists x(p(x)) \\ \stackrel{21}{\iff} & \exists x(\neg p(x)) \vee \exists x(p(x)) \\ \stackrel{24}{\iff} & \exists x(\neg p(x) \vee p(x)) \\ \stackrel{2}{\iff} & \exists x(1) \\ \stackrel{27b}{\iff} & \mathbf{1} \quad \text{(universally valid)} \end{aligned} \tag{130}$$

• **Example 2:**

$$\exists x(p(x)) \rightarrow \forall x(p(x)) \quad (\text{closed})$$

$$\stackrel{12}{\iff} \neg \exists x(p(x)) \vee \forall x(p(x))$$

$$\stackrel{21}{\iff} \forall x(\neg p(x)) \vee \forall x(p(x))$$

$$\stackrel{29}{\iff} \forall x(\neg p(x)) \vee \forall y(p(y))$$

$$\stackrel{28a}{\iff} \forall x(\neg p(x) \vee \forall y(p(y)))$$

$$\stackrel{28c}{\iff} \forall x, y(\neg p(x) \vee p(y)) \quad (131)$$

- This formula says, if there is at least one x turning $p(x)$ true then $p(x)$ is always true.
- In turn, if there is *no* x turning $p(x)$ true then $p(x)$ is, logically, always false.
- Consequently, $p(x)$ is independently of x either true or false (a propositional variable).

- The next normalization step requires a wider notion of equivalence.
- We will refer to it as **equisatisfiability**.

- Consider the example formulas

$$f_1 = \forall x \exists y (p(x, y)) \quad (132)$$

and

$$f_2 = \forall x (p(x, s(x))). \quad (133)$$

- f_1 and f_2 are not equivalent.
- They do not even use the same signature (f_2 uses the function symbol s not existing in f_1).

- However, we can relate f_1 and f_2 as follows:
- If a structure S_1 is a model for f_1 , i.e.,

$$S_1(f_1) = 1 \quad (134)$$

then it can be extended to a structure S_2 being a model for f_2 ,

$$S_2(f_2) = 1. \quad (135)$$

- This can be done by defining an interpretation s^J such that

$$p(x, s(x)) \quad (136)$$

is true for all possible x of the universe.

- Formally, two formulas f_1 and f_2 are equisatisfiable if

$$\models S_1(f_1) \leftrightarrow S_2(f_2) \quad (137)$$

also written as

$$f_1 \approx f_2. \quad (138)$$

- We are given a signature $\Sigma = \langle V, F, P, \text{arity} \rangle$ and a closed formula f of the form

$$f = \forall x_1, \dots, x_n \exists y (g(x_1, \dots, x_n, y)). \quad (139)$$

- We choose a new n -ary function symbol $s \notin F$ and extend the signature:
 $\Sigma' = \langle V, F \cup \{s\}, P, \text{arity} \cup \{\langle s, n \rangle\} \rangle. \quad (140)$

- Now, we define the formula f' as follows:

$$f' := \text{skolem}(f) := \forall x_1, \dots, x_n (g(x_1, \dots, x_n, s(x_1, \dots, x_n))). \quad (141)$$

- Here, we have dropped the existential quantifier.
- Every occurrence of the variable y has been replaced by the term $s(x_1, \dots, x_n)$.

- For the resulting formula f' , we have

$$f' \approx f. \quad (142)$$

Skolemization: examples

$$f_1 = \forall x, y \exists z (p(x, y, z)),$$

$$\text{skolem}(f_1) = \forall x, y (p(x, y, s_z^1(x, y)));$$

$$f_2 = \forall x \exists y \forall z (p(x, y, z)),$$

$$\text{skolem}(f_2) = \forall x, z (p(x, s_y^2(x), z));$$

$$f_3 = \exists x \forall y \exists z (p(x, y, z)),$$

$$\text{skolem}(f_3) = \forall y \exists z (p(s_x^3, y, z)),$$

$$\text{skolem}(\text{skolem}(f_3)) = \forall y (p(s_x^3, y, s_z^3(y))) \quad (143)$$

- This algorithm turns an first-order formula f into **clausal normal form**:
 1. Convert f into prenex normal form.
 2. Eliminate all existential quantifiers by skolemization. The result is a formula of the form

$$f' = \forall x_1, \dots, x_n (g) \tag{144}$$

where g contains **no quantifiers**.

3. Since g contains only atomic formulas connected by logical connectives, it can be turned into conjunctive normal form which produces

$$f'' = \forall x_1, \dots, x_n (d_1 \wedge \dots \wedge d_m) \tag{145}$$

where d_i are disjunctions of **literals** (in first-order logic, literals are atomic formulas or their negations).

4. Applying Equivalence 23, the universal quantifiers can be distributed onto all d_i s resulting in the clausal normal form

$$f''' = c_1 \wedge \dots \wedge c_m \quad \text{with} \quad c_i = \forall x_1, \dots, x_n (d_i), \quad (146)$$

a conjunction of the **first-order clauses** c_1, \dots, c_m .

5. This notation can further be simplified by agreeing that all free variables are **implicitly universally quantified**:

$$\begin{aligned} f^{(4)} &= d_1 \wedge \dots \wedge d_m \\ \Leftrightarrow &\{d_1, \dots, d_m\}. \end{aligned} \quad (147)$$

- Turn the following formulas into clausal normal form:
 1. $f_1 = \forall y \exists x ((1 + x) \cdot y = 0 \rightarrow \forall z (x \cdot z = 0))$,
 2. $f_2 = \exists x, y (p(x) \wedge p(y) \leftrightarrow \exists z (p(z)))$,
 3. $\neg f_2$.

- Our goal is to find out whether a formula f is universally valid:

$$\models f. \tag{148}$$

- This is the same as to tell whether f 's negation is unsatisfiable:

$$\{\neg f\} \models 0. \tag{149}$$

- This can be done by

1. transforming $\neg f$ into clausal normal form

$$c_1 \wedge \dots \wedge c_m \approx \neg f, \tag{150}$$

2. trying to prove inconsistency from the set of clauses:

$$\{c_1, \dots, c_m\} \vdash \{\}. \tag{151}$$

- We want to find out whether the formula

$$f := \exists x \forall y (p(x, y)) \rightarrow \forall y \exists x (p(x, y)) \quad (152)$$

is universally valid.

- This can be done by using the provability algorithm:

1. Transforming $\neg f$ into clausal normal form:

1.1 Conversion into prenex normal form:

$$\begin{aligned} & \neg(\exists x \forall y (p(x, y)) \rightarrow \forall y \exists x (p(x, y))) \\ & \stackrel{12}{\iff} \neg(\neg \exists x \forall y (p(x, y)) \vee \forall y \exists x (p(x, y))) \\ & \stackrel{11}{\iff} \exists x \forall y (p(x, y)) \wedge \neg \forall y \exists x (p(x, y)) \\ & \stackrel{21}{\iff} \exists x \forall y (p(x, y)) \wedge \exists y \neg \exists x (p(x, y)) \\ & \stackrel{22}{\iff} \exists x \forall y (p(x, y)) \wedge \exists y \forall x (\neg p(x, y)) \\ & \stackrel{29}{\iff} \exists x \forall y (p(x, y)) \wedge \exists v \forall u (\neg p(u, v)) \\ & \stackrel{28b}{\iff} \exists x (\forall y (p(x, y)) \wedge \exists v \forall u (\neg p(u, v))) \end{aligned}$$

$$\begin{aligned} &\stackrel{28d}{\iff} \exists x, v (\forall y (p(x, y)) \wedge \forall u (\neg p(u, v))) \\ &\stackrel{27a}{\iff} \exists x, v (\forall y (p(x, y)) \wedge \forall y, u (\neg p(u, v))) \\ &\stackrel{23}{\iff} \exists x, v \forall y (p(x, y) \wedge \forall u (\neg p(u, v))) \\ &\stackrel{27a}{\iff} \exists x, v \forall y (\forall u (p(x, y)) \wedge \forall u (\neg p(u, v))) \\ &\stackrel{23}{\iff} \exists x, v \forall y, u (p(x, y) \wedge \neg p(u, v)) \\ &=: f' \end{aligned} \tag{153}$$

1.2 Skolemization:

- f' is of the form
$$f' = \exists x (g(x)) \quad \text{with} \quad g = \exists v \forall y, u (p(x, y) \wedge \neg p(u, v)). \tag{154}$$
- Here, g is 1-ary since there are no universal quantifiers in front of the existential quantifier.

- Consequently, skolemization introduces the 0-ary function s_x replacing x :

$$\begin{aligned}
 f'' &= \text{skolem}(f') \\
 &= g(s_x()) \\
 &= g(s_x) \\
 &= \exists v \forall y, u(p(s_x, y) \wedge \neg p(u, v)).
 \end{aligned}
 \tag{155}$$

- In a second step, the existential quantifier in front of v is replaced by skolemization:

$$\begin{aligned}
 f''' &= \text{skolem}(f'') \\
 &= \forall y, u \underbrace{(p(s_x, y) \wedge \neg p(u, s_v))}_h.
 \end{aligned}
 \tag{156}$$

1.3-5 Clausal normal form:

Since h is already in conjunctive normal form, we have

$$f^{(4)} = p(s_x, y) \wedge \neg p(u, s_v).
 \tag{157}$$

2. Proving inconsistency:

- We are given clausal normal form

$$M := \{ \{ p(s_x, y) \}, \{ \neg p(u, s_v) \} \}. \quad (158)$$

- Since all variables in first-order clauses are implicitly universally bound, we can substitute y by s_v and u by s_x resulting in

$$M \leftrightarrow \{ \{ p(s_x, s_v) \}, \{ \neg p(s_x, s_v) \} \}. \quad (159)$$

- Application of the cut rule gives

$$M \leftrightarrow \{ \}. \quad (160)$$

- Consequently, we have shown that

$$\models f. \quad (161)$$

- In the provability example, we have “guessed” the terms s_x and s_o to be able to apply the cut rule.
- We now want to study a method to do systematically calculate terms which the cut rule can be applied to.

- In order to do so, for a given signature Σ , we define a substitution

$$\sigma = \{\langle x_1, t_1 \rangle, \dots, \langle x_n, t_n \rangle\} =: [x_1 \mapsto t_1, \dots, x_n \mapsto t_n] \quad (162)$$

with

$$x_i \in V, t_i \in T_\Sigma \text{ for } i \in \{1, \dots, n\} \text{ and } x_i \neq x_j \text{ for } i \neq j. \quad (163)$$

- The **domain** of the substitution is defined as

$$\text{dom}(\sigma) = \{x_1, \dots, x_n\}. \quad (164)$$

- Given a term t and a substitution σ , we define the **application of σ to t** (written $t\sigma$) inductively as
 1. $x_i\sigma := t_i$ for $x_i \in \text{dom}(\sigma)$,
 2. $y\sigma := y$ for $y \in V \wedge y \notin \text{dom}(\sigma)$,
 3. $f(s_1, \dots, s_m)\sigma := f(s_1\sigma, \dots, s_m\sigma)$ otherwise.

- **Example:** We are given the substitution

$$\sigma = [x \mapsto c, y \mapsto f(d)]. \quad (165)$$

- Then, we have the following applications:

1. $z\sigma = z$,
2. $f(y)\sigma = f(f(d))$,
3. $h(x, g(y))\sigma = h(c, g(f(d)))$,
4. $\{p(y), q(d, h(z, x))\}\sigma = \{p(f(d)), q(d, h(z, c))\}$.

- A **syntactical equation (SE)** is a construct of the form $s \doteq t$ where s and t are either both terms or both atomic formulas.

- A **system of syntactical equations (SSE)** is a set of SEs:

$$E := \{s_1 \doteq t_1, \dots, s_n \doteq t_n\}. \quad (166)$$

- Given an SSE E and a substitution σ , we define the **application of σ to E** as

$$E\sigma := \{s_1\sigma \doteq t_1\sigma, \dots, s_n\sigma \doteq t_n\sigma\}. \quad (167)$$

- A substitution σ **solves** an SE $s \doteq t$ iff we have $s\sigma = t\sigma$.
- If E is an SSE then the substitution σ is called a **unifier** iff it solves every SE in E .

- **Example: Show that**

$$\sigma = [x_1 \mapsto x_2, x_3 \mapsto f(x_4)] \quad (168)$$

is a unifier of the SSE

$$E = \{p(f(x_4)) \doteq p(x_3), q(x_1, x_2) \doteq q(x_2, x_1)\}. \quad (169)$$

- We now want to study an **algorithm calculating a unifier σ** for a given SSE E .

- A number of **reduction rules** will transform a tuple consisting of an SSE and a substitution into another such tuple:

$$\langle E_1, \sigma_1 \rangle \rightsquigarrow \langle E_2, \sigma_2 \rangle. \quad (170)$$

- Here come the reduction rules:

1. If $y \in V \wedge t \in \mathcal{T}_\Sigma \wedge y \notin \text{var}(t)$ then

$$\langle E \cup \{y \doteq t\}, \sigma \rangle \rightsquigarrow \langle E[y \mapsto t], \sigma \cup \{\langle y, t \rangle\} \rangle. \quad (171)$$

- If an SSE contains an SE of the form $y \doteq t$ where y is a variable not contained in term t then the SE is solved by the substitution $[y \mapsto t]$.
- Consequently, the SE can be dropped in favor of applying the substitution to all other SEs in the SSE and adding it to σ .
- E.g., $x \doteq s(y)$.

2. If $y \in V \wedge t \in T_\Sigma \wedge y \in \text{var}(t) \wedge y \neq t$ then

$$\langle E \cup \{y \doteq t\}, \sigma \rangle \rightsquigarrow \Omega. \quad (172)$$

– If an SSE contains an SE of the form $y \doteq t$ where y is a variable contained in term t but not t itself then the SE is **not solvable**.

– That is because every possible term we can substitute for y will also affect t and render it different.

– E.g., $x \doteq s(x)$.

3. If $y \in V \wedge t \in T_\Sigma \wedge t \notin V$ then

$$\langle E \cup \{t \doteq y\}, \sigma \rangle \rightsquigarrow \langle E \cup \{y \doteq t\}, \sigma \rangle. \quad (173)$$

– Moves the variable to the front.

– E.g., $s(y) \doteq x$.

4. $\langle E \cup \{t \doteq t\}, \sigma \rangle \rightsquigarrow \langle E, \sigma \rangle. \quad (174)$

– Drops trivial SEs.

– E.g., $s(y) \doteq s(y)$.

5. If $f \in F \wedge n = \text{arity}(f) \wedge s_1, \dots, s_n, t_1, \dots, t_n \in \mathcal{T}_\Sigma$ then

$$\begin{aligned} \langle E \cup \{f(s_1, \dots, s_n) \doteq f(t_1, \dots, t_n)\}, \sigma \rangle \\ \rightsquigarrow \langle E \cup \{s_1 \doteq t_1, \dots, s_n \doteq t_n\}, \sigma \rangle. \end{aligned} \quad (175)$$

– From the arguments of an n -ary function (predicate) present on both sides of an SE, n individual SEs are derived.

– E.g., $s(s(x), y) \doteq s(y, z)$.

6. If $f, g \in F \wedge f \neq g \wedge m = \text{arity}(f) \wedge n = \text{arity}(g)$

$\wedge s_1, \dots, s_m, t_1, \dots, t_n \in \mathcal{T}_\Sigma$ then

$$\langle E \cup \{f(s_1, \dots, s_m) \doteq g(t_1, \dots, t_n)\}, \sigma \rangle \rightsquigarrow \Omega. \quad (176)$$

– Different functions (predicates) cannot be unified.

– E.g., $f(x) \doteq g(x)$.

- Given an SSE E , to determine a unifier, we start off with an empty substitution:

$$\langle E, [] \rangle. \tag{177}$$

- Now, we apply the reduction rules until we have either of the following cases:

a) We can derive Ω meaning that E is **not solvable**.

b) We can show that

$$\langle E, [] \rangle \rightsquigarrow \langle \{\}, \sigma \rangle. \tag{178}$$

Here, σ is the **unifier of E** also written as

$$\sigma = \text{mgu}(E) \tag{179}$$

(the **most general unifier**) with the special case

$$\text{mgu}(s, t) := \text{mgu}(\{s \doteq t\}). \tag{180}$$

- We are given an SSE with a single SE:

$$E = \{p(x_1, f(x_4)) \doteq p(x_2, x_3)\} \tag{181}$$

- Now, we attempt to determine a unifier:

$$\begin{aligned} & \langle \{p(x_1, f(x_4)) \doteq p(x_2, x_3)\}, [] \rangle \\ \overset{5}{\rightsquigarrow} & \langle \{x_1 \doteq x_2, f(x_4) \doteq x_3\}, [] \rangle \\ \overset{1}{\rightsquigarrow} & \langle \{f(x_4) \doteq x_3\}, [x_1 \mapsto x_2] \rangle \\ \overset{3}{\rightsquigarrow} & \langle \{x_3 \doteq f(x_4)\}, [x_1 \mapsto x_2] \rangle \\ \overset{1}{\rightsquigarrow} & \langle \{\}, [x_1 \mapsto x_2, x_3 \mapsto f(x_4)] \rangle \end{aligned} \tag{182}$$

- That is, we have

$$\begin{aligned} \text{mgu}(E) &= \text{mgu}(p(x_1, f(x_4)), p(x_2, x_3)) \\ &= [x_1 \mapsto x_2, x_3 \mapsto f(x_4)]. \end{aligned} \tag{183}$$

- **Determine, if possible, a unifier of the following SSEs:**
 - a) $E_1 = \{p(d, x_4) \doteq p(x_2, h(c, d)), q(h(d, x_1)) \doteq q(x_4)\}$,
 - b) $E_2 = \{p(h(x_1, c)) \doteq p(x_2), q(x_2, d) \doteq q(h(d, c), x_4)\}$,
 - c) $E_3 = \{p(h(x_2, d)) \doteq p(h(x_1, d)), q(x_1, c) \doteq q(h(x_2, d), c)\}$.

- The first-order calculus makes use of **two inference rules** we will define next.

- The first-order **resolution rule** expects

1. two first-order clauses c_1 and c_2 and
2. two atomic formulas $p(s_1, \dots, s_n)$ and $p(t_1, \dots, t_n)$ whose syntactical equation is solvable, i.e., we can determine the unifier

$$\mu = \text{mgu}(p(s_1, \dots, s_n), p(t_1, \dots, t_n)). \quad (184)$$

- Then, this is the definition of the resolution rule:

$$\frac{c_1 \cup \{p(s_1, \dots, s_n)\}, \quad c_2 \cup \{\neg p(t_1, \dots, t_n)\}}{\therefore c_1\mu \cup c_2\mu}$$

- The resolution rule makes use of the cut rule and the **substitution rule**

$$\frac{c}{\therefore c\sigma}$$

where c is a first-order clause and σ is a substitution.

- When applying the resolution rule, it is sometimes necessary to **rename variables**.

- In the following first-order clause set

$$M = \{\{p(x)\}, \{\neg p(f(x))\}\}, \quad (185)$$

unification will result in Ω since both involved clauses use the same variable x .

- Substituting x by y in one of the clauses does not change semantics

$$M' = \{\{p(x)\}, \{\neg p(f(y))\}\} \quad (186)$$

simply because

$$\forall x(\neg p(f(x))) \stackrel{29}{\Leftrightarrow} \forall y(\neg p(f(y))). \quad (187)$$

- Now, we are able to determine the unifier

$$\mu = \text{mgu}(p(x), p(f(y))) = [x \mapsto f(y)] \quad (188)$$

- In conclusion, we can prove contradiction by applying the resolution rule:

$$\{\{p(x)\}, \{\neg p(f(y))\}\} \stackrel{\text{res. } [x \mapsto f(y)]}{\vdash} \{\}. \quad (189)$$

- Consider the following example

$$M = \{\{p(x), p(y)\}, \{\neg p(x), \neg p(y)\}\} \quad (190)$$

- Application of the resolution rule using $c_1 = p(x)$ and $c_2 = \neg p(x)$ and the unifier $\mu = \text{mgu}(p(x), p(y)) = \{\}$ produces

$$\begin{aligned} & \{\{p(x), p(y)\}, \{\neg p(x), \neg p(y)\}\} \\ \text{res. } \{\} & \vdash \{\{p(y), \neg p(y)\}\} \\ & \vdash 1. \end{aligned} \quad (191)$$

- This is why, as in propositional logic, we have to apply **case distinction** to account for multiple occurrences of negated literals:

- Pick $\{p(x)\}$:
- $$\{\{p(x)\}, \{p(x), p(y)\}, \{\neg p(x), \neg p(y)\}\}$$

$$\text{res. } \{\} \vdash \{\{p(x)\}, \{p(x), p(y)\}, \{\neg p(y)\}\}$$

$$\text{res. } [x \mapsto y] \vdash \{\{\}, \{p(x), p(y)\}\}$$

$$\vdash 0. \quad (192)$$

• **Pick** $\neg\{p(x)\} \leftrightarrow \neg\forall x(p(x)) \leftrightarrow \exists x(\neg p(x)) \approx \neg p(s) \leftrightarrow \{\neg p(s)\}$:

$$\{\{\neg p(s)\}, \{p(x), p(y)\}, \{\neg p(x), \neg p(y)\}\}$$

$$\text{res. } [x \mapsto s] \vdash \{\{\neg p(s)\}, \{p(y)\}, \{\neg p(x), \neg p(y)\}\}$$

$$\text{res. } [y \mapsto s] \vdash \{\{\}, \{\neg p(x), \neg p(y)\}\}$$

$$\vdash 0. \tag{193}$$

• **So, in conclusion, we have**

$$\{\{p(x), p(y)\}, \{\neg p(x), \neg p(y)\}\} \vdash 0. \tag{194}$$

- In general, we account for multiple occurrences of unifiable predicates in a clause by means of **factorization rules**.

- Given

1. the first-order clause c and
2. two atomic formulas $p(s_1, \dots, s_n)$ and $p(t_1, \dots, t_n)$ whose syntactical equation is solvable, i.e., we can determine the unifier

$$\mu = \text{mgu}(p(s_1, \dots, s_n), p(t_1, \dots, t_n)). \quad (195)$$

- Then, these are the definitions of the factorization rules:

$$\frac{c \cup \{p(s_1, \dots, s_n), p(t_1, \dots, t_n)\}}{\therefore c\mu \cup \{p(s_1, \dots, s_n)\mu\}}$$

and

$$\frac{c \cup \{\neg p(s_1, \dots, s_n), \neg p(t_1, \dots, t_n)\}}{\therefore c\mu \cup \{\neg p(s_1, \dots, s_n)\mu\}}$$

- Returning to the above example, application of the factorization rules produces

$$\{\{p(x), p(y)\}, \{\neg p(x), \neg p(y)\}\}$$

$$\text{fact.} \vdash \{\{p(y)\}, \{\neg p(x), \neg p(y)\}\}$$

$$\text{fact.} \vdash \{\{p(y)\}, \{\neg p(y)\}\}$$

$$\text{res.} \square \vdash 0.$$

(196)

- Can you derive contradiction from the following clause sets?
 - a) $M_1 = \{\{\neg p(y, y)\}, \{p(f(x), y), p(y, g(v))\}\}$
 - b) $M_2 = \{\{p(x, y), p(g(u), v), p(z, v)\}, \{\neg p(x, y), \neg p(g(u), v)\}\}$
 - c) $M_3 = \{\{p(x, f(z)), p(g(u), v), p(g(u), f(z))\}\}$
 - d) $M_4 = \{\{p(f(g(u)), v), \neg p(f(z), v)\}, \{\neg p(x, y), \neg p(g(u), v)\}\}$

- Similarly to the function cut introduced for the definition of the DP algorithm in propositional logic, we introduce the function **res** for first-order logic.
- The function **res** applies the resolution rule to a set of clauses C_0 with respect to a literal (unit clause).
- As opposed to cut, it does not drop any clauses from C_0 but simply adds conclusions from the resolution rule to the clause set.
- The reason that clauses are not dropped is that different atomic formulas may be based on the same predicate but different arguments which may produce a different unifier and, hence, a different result clause.
- If the original clause would have been dropped, this different clause would have been missed.

- **Example 1:**

$$\begin{aligned} \text{res}(\{p(x), \neg q(f(t))\}, \{q(x)\}) = \\ \{p(x), \neg q(f(t))\}, \{p(f(t))\}, \{q(x)\} \end{aligned} \quad (197)$$

- **Example 2:**

$$\begin{aligned} \text{res}(\{p(y), \neg q(x)\}, \{p(f(y)), \neg q(x)\}, \{\neg p(x)\}) = \\ \{p(y), \neg q(x)\}, \{\neg q(x)\}, \{p(f(y)), \neg q(x)\}, \{\neg q(f(y))\}, \{\neg p(x)\} \end{aligned} \quad (198)$$

- **Example 3 (infinite loop):**

$$C_0 := \{\{\neg p(x), p(f(x))\}, \{p(x)\}\}$$

$$C_1 := \text{res}(C_0, p(x))$$

$$= \{\{\neg p(x), p(f(x))\}, \{p(x)\}\}$$

$$= \{\{\neg p(x), p(f(x))\}, \{p(x)\}\}$$

$$C_2 := \text{res}(C_1, p(f(y)))$$

$$= \{\{\neg p(x), p(f(x))\}, \{p(f(f(y)))\}, \{p(x)\}\}$$

$$C_3 := \text{res}(C_2, p(f(f(y)))) \dots \quad (199)$$

- We have now learned all the pieces necessary to apply first-order calculus in a systematic way.
- Let us reiterate on the involved steps by means of an example.
- This is our knowledge base:
 - a) *Every dragon is happy if all his children know how to fly.*
 - b) *Every red dragon knows how to fly.*
 - c) *Children of red dragons are red.*
- **Now, let us try to prove whether or not**
 - d) *All red dragons are happy.*

- To get started, we define a **signature** $\Sigma_a = \langle V, F, P, \text{arity} \rangle$ with

$$V = \{x, y\}$$

$$F = \{\}$$

$$P = \{r, f, h, c\}$$

$$\text{arity} = \{\langle r, 1 \rangle \langle f, 1 \rangle \langle h, 1 \rangle \langle c, 2 \rangle\}$$

- We assume that the **universe** contains all dragons.
- The predicates have the following **interpretations**:
 - $r(x)$ is 1 iff x is red.
 - $f(x)$ is 1 iff x knows how to fly.
 - $h(x)$ is 1 iff x is happy.
 - $c(x, y)$ is 1 iff x is y 's child.

- Formalizing knowledge base and claim:

a) $f_1 ::= \forall x(\forall y(c(y, x) \rightarrow f(y)) \rightarrow h(x))$

b) $f_2 ::= \forall x(r(x) \rightarrow f(x))$

c) $f_3 ::= \forall x(\exists y(c(x, y) \wedge r(y)) \rightarrow r(x))$

d) $f_4 ::= \forall x(r(x) \rightarrow h(x))$

- Now, to see whether the claim d) can be derived from the knowledge base a) to c) is to prove that

$$f ::= f_1 \wedge f_2 \wedge f_3 \rightarrow f_4 \quad (200)$$

is **universally valid**.

- This is the same as to prove that $\neg f$ is a **contradiction**:

$$\begin{aligned}\neg f &\leftrightarrow \neg(f_1 \wedge f_2 \wedge f_3 \rightarrow f_4) \\ &\leftrightarrow \neg(\neg(f_1 \wedge f_2 \wedge f_3) \vee f_4) \\ &\leftrightarrow \neg\neg(f_1 \wedge f_2 \wedge f_3) \vee \neg f_4 \\ &\leftrightarrow \neg\neg(f_1 \wedge f_2 \wedge f_3) \wedge \neg f_4 \\ &\leftrightarrow f_1 \wedge f_2 \wedge f_3 \wedge \neg f_4\end{aligned}\tag{201}$$

- Next, we need to turn $\neg f$ into a set of clauses.
- This can be done by turning f_1 , f_2 , f_3 , and $\neg f_4$ individually into **clausal normal form**.

First-order calculus: example (cont.)

$$\begin{aligned} f_1 &\leftrightarrow \forall x(\forall y(c(y, x) \rightarrow f(y)) \rightarrow h(x)) \\ &\leftrightarrow \forall x(\forall y(\neg c(y, x) \vee f(y)) \rightarrow h(x)) \\ &\leftrightarrow \forall x(\neg \forall y(\neg c(y, x) \vee f(y)) \vee h(x)) \\ &\leftrightarrow \forall x(\exists y(\neg(\neg c(y, x) \vee f(y)))) \vee h(x)) \\ &\leftrightarrow \forall x(\exists y(\neg \neg c(y, x) \wedge \neg f(y)) \vee h(x)) \\ &\leftrightarrow \forall x(\exists y(c(y, x) \wedge \neg f(y)) \vee h(x)) \\ &\approx \forall x(\exists y(c(y, x) \wedge \neg f(y) \vee h(x))) \\ &\leftrightarrow \forall x(c(s(x), x) \wedge \neg f(s(x))) \vee h(x)) \\ &\leftrightarrow \forall x((c(s(x), x) \vee h(x)) \wedge (\neg f(s(x)) \vee h(x))) \\ &\leftrightarrow \{\{c(s(x), x), h(x)\}, \{\neg f(s(x)), h(x)\}\} \end{aligned} \tag{202}$$
$$\begin{aligned} f_2 &\leftrightarrow \forall x(r(x) \rightarrow f(x)) \\ &\leftrightarrow \forall x(\neg r(x) \vee f(x)) \\ &\leftrightarrow \{\{\neg r(x), f(x)\}\} \end{aligned} \tag{203}$$

First-order calculus: example (cont.)

$$\begin{aligned} f_3 &\leftrightarrow \forall x(\exists y(c(x, y) \wedge r(y)) \rightarrow r(x)) \\ &\leftrightarrow \forall x(\neg\exists y(c(x, y) \wedge r(y)) \vee r(x)) \\ &\leftrightarrow \forall x(\forall y(\neg(c(x, y) \wedge r(y)))) \vee r(x) \\ &\leftrightarrow \forall x(\forall y(\neg c(x, y) \vee \neg r(y))) \vee r(x) \\ &\leftrightarrow \forall x, y(\neg c(x, y) \vee \neg r(y) \vee r(x)) \\ &\leftrightarrow \{\{\neg c(x, y), \neg r(y), r(x)\}\} \end{aligned} \tag{204}$$

$$\begin{aligned} \neg f_4 &\leftrightarrow \neg\forall x(r(x) \rightarrow h(x)) \\ &\leftrightarrow \neg\forall x(\neg r(x) \vee h(x)) \\ &\leftrightarrow \exists x(\neg(\neg r(x) \vee h(x))) \\ &\leftrightarrow \exists x(r(x) \wedge \neg h(x)) \\ &\approx r(t) \wedge \neg h(t) \\ &\leftrightarrow \{\{r(t)\}, \{\neg h(t)\}\} \end{aligned} \tag{205}$$

- Applying the DP algorithm:

$$C_0 := \{ \{c(s(x), x), h(x)\}, \{\neg f(s(x)), h(x)\}, \{\neg r(x), f(x)\}, \{\neg c(x, y), \neg r(y), r(x)\}, \{r(t)\}, \{\neg h(t)\} \}$$

$$C_1 := \text{res}(C_0, r(t)) \\ = \{ \{c(s(x), x), h(x)\}, \{\neg f(s(x)), h(x)\}, \{\neg r(x), f(x)\}, \{f(t)\}, \{\neg c(x, y), \neg r(y), r(x)\}, \{\neg c(x, t), r(x)\}, \{r(t)\}, \{\neg h(t)\} \}$$

$$C_2 := \text{res}(C_1, \neg h(t)) \\ = \{ \{c(s(x), x), h(x)\}, \{c(s(t), t)\}, \{\neg f(s(x)), h(x)\}, \{\neg f(s(t))\}, \{\neg r(x), f(x)\}, \{f(t)\}, \{\neg c(x, y), \neg r(y), r(x)\}, \{\neg c(x, t), r(x)\}, \{r(t)\}, \{\neg h(t)\} \} \quad (206)$$

$$\begin{aligned}
 C_3 &::= \text{res}(C_2, c(s(t), t)) \\
 &= \{\{c(s(x), x), h(x)\}, \{c(s(t), t)\}, \{\neg f(s(x)), h(x)\}, \{\neg f(s(t))\}, \\
 &\quad \{\neg r(x), f(x)\}, \{f(t)\}, \{\neg c(x, y), \neg r(y), r(x)\}, \\
 &\quad \{\neg r(t), r(s(t))\}, \{\neg c(x, t), r(x)\}, \{r(s(t))\}, \{r(t)\}, \{\neg h(t)\}\} \\
 C_4 &::= \text{res}(C_3, r(s(t))) \\
 &= \{\{c(s(x), x), h(x)\}, \{c(s(t), t)\}, \{\neg f(s(x)), h(x)\}, \{\underline{\neg f(s(t))}\}, \\
 &\quad \{\neg r(x), f(x)\}, \{\underline{f(s(t))}\}, \{\neg c(x, y), \neg r(y), r(x)\}, \\
 &\quad \{\neg c(x, s(t)), r(x)\}, \{\neg r(t), r(s(t))\}, \{\neg c(x, t), r(x)\}, \{r(s(t))\}, \\
 &\quad \{r(t)\}, \{\neg h(t)\}\} \quad (\text{unsatisfiable}) \quad (207)
 \end{aligned}$$

We know that

Everybody loves only those people who do not love a gardener.

Now, prove that

Gardeners do not love themselves.

Solutions to selected exercises

• Transform into CNF:

$$g := \underbrace{p \rightarrow q \rightarrow r}_a \leftrightarrow \underbrace{\neg s \vee t}_b \quad (208)$$

$$1. \quad g \Leftrightarrow \underbrace{(\neg a \vee b)}_c \wedge \underbrace{(\neg b \vee a)}_d \quad (209)$$

$$2. \quad a \Leftrightarrow \neg p \vee q \rightarrow r \quad (210)$$
$$\Leftrightarrow \neg(\neg p \vee q) \vee r$$

$$3. \quad a \Leftrightarrow p \wedge \neg q \vee r \quad (211)$$

$$\neg a \Leftrightarrow \neg(p \wedge \neg q) \wedge \neg r \quad (212)$$
$$\Leftrightarrow (\neg p \vee q) \wedge \neg r$$

$$\neg b \Leftrightarrow s \wedge \neg t \quad (213)$$

Conjunctive normal form: solution to exercise (cont.)

$$4. \quad c \Leftrightarrow (\neg p \vee q) \wedge \neg r \vee b$$

$$\Leftrightarrow (\neg p \vee q \vee b) \wedge (\neg r \vee b) \quad (214)$$

$$\Leftrightarrow (\neg p \vee q \vee \neg s \vee t) \wedge (\neg r \vee \neg s \vee t)$$

$$d \Leftrightarrow s \wedge \neg t \vee a$$

$$\Leftrightarrow \underbrace{(s \vee a)}_e \wedge \underbrace{(\neg t \vee a)}_f \quad (215)$$

$$e \Leftrightarrow p \wedge \neg q \vee r \vee s$$

$$\Leftrightarrow (p \vee r \vee s) \wedge (\neg q \vee r \vee s) \quad (216)$$

$$f \Leftrightarrow (p \vee r \vee \neg t) \wedge (\neg q \vee r \vee \neg t) \quad (217)$$

$$5. \quad g \Leftrightarrow \{\{\neg p, q, \neg s, t\}, \{\neg r, \neg s, t\}, \{p, r, s\}, \{\neg q, r, s\},$$

$$\{p, r, \neg t\}, \{\neg q, r, \neg t\}\} \quad (218)$$

- Using the above signature Σ_a , structure S_a , and variable assignment I_a , determine the evaluation of the following formulas:

1. $\underbrace{\exists x((1 + x) \cdot y = 0)}_f$

Evaluating the formula using Rule 8:

$$S(I_a[x/b], f) = 1 \rightarrow S(I_a, \exists x(f)) = 1 \quad (219)$$

2. $\forall x((1 + x) \cdot y = 0)$

Evaluating the formula using Rule 7:

$$S(I_a[x/a], f) = 0 \rightarrow S(I_a, \forall x(f)) = 0 \quad (220)$$

3. $\forall x, y((1 + x) \cdot y = 0)$

Evaluating the formula using Rule 7:

$$S(I_a[x/a][y/b], f) = 0 \rightarrow S(I_a, \forall x, y(f)) = 0 \quad (221)$$

4. $\forall x\exists y((1 + x) \cdot y = 0)$

Evaluating the formula using Rules 7 and 8:

$$\begin{aligned} S(I_a[x/a][y/a], f) &= 1 \\ \wedge S(I_a[x/b][y/a], f) &= 1 \\ \rightarrow S(I_a, \forall x\exists y(f)) &= 1 \end{aligned} \quad (222)$$

$$5. \forall y \exists x ((1 + x) \cdot y = 0 \rightarrow \underbrace{\forall z (x \cdot z = 0)}_g))$$

Due to

$$(g \leftrightarrow 1) \rightarrow (f \rightarrow g \leftrightarrow 1), \quad (223)$$

it is enough to show that

$$\forall y \exists x (g) \leftrightarrow 1 \quad \text{that is} \quad \forall y \exists x \forall z (h) \leftrightarrow 1. \quad (224)$$

Evaluating the formula using Rules 7 and 8:

$$\begin{aligned} & S(I_a[x/a][y/a][z/a], h) = 1 \\ \wedge & S(I_a[x/a][y/a][z/b], h) = 1 \\ \wedge & S(I_a[x/a][y/b][z/a], h) = 1 \\ \wedge & S(I_a[x/a][y/b][z/b], h) = 1 \\ \rightarrow & S(I_a, \forall y \exists x (f \rightarrow g)) = 1 \end{aligned} \quad (225)$$

- We know that

Everybody loves only those people who do not love a gardener.

- Let us use the predicates
 - $l(x, y)$ which is 1 iff x loves y and
 - $g(x)$ which is 1 iff x is a gardener.

- Let us further use an auxiliary predicate
 - $n(x)$ which is 1 iff x does not love a gardener.

- Using $n(x)$'s definition, the above axiom could be written as

Every x loves only those y for which $n(y)$

which can be formally expressed as

$$f := \forall x, y (l(x, y) \rightarrow n(y)). \quad (226)$$

- Expressing $n(y)$ in terms of $l(y, z)$ and $g(z)$:

$$n(y) \leftrightarrow \neg \exists z (l(y, z) \wedge g(z)). \quad (227)$$

- Hence, Formula 226 becomes

$$\begin{aligned} f &\leftrightarrow \forall x, y (l(x, y) \rightarrow \neg \exists z (l(y, z) \wedge g(z))) \\ &\leftrightarrow \forall x, y (\neg l(x, y) \vee \neg \exists z (l(y, z) \wedge g(z))) \\ &\leftrightarrow \forall x, y (\neg l(x, y) \vee \forall z (\neg (l(y, z) \wedge g(z)))) \\ &\leftrightarrow \forall x, y, z (\neg l(x, y) \vee \neg l(y, z) \vee \neg g(z)) \\ &\leftrightarrow \{\{\neg l(x, y), \neg l(y, z), \neg g(z)\}\} \end{aligned} \quad (228)$$

- Our conjecture is that

Gardeners do not love themselves

which can be formally expressed as

$$g := \forall x (g(x) \rightarrow \neg l(x, x)). \quad (229)$$

- To prove g , we need to conjunctively combine f with g 's negation, so, let us attack the latter now:

$$\begin{aligned} \neg g &\leftrightarrow \neg(\forall x(g(x) \rightarrow \neg l(x, x))) \\ &\leftrightarrow \neg(\forall x(\neg g(x) \vee \neg l(x, x))) \\ &\leftrightarrow \exists x(\neg(\neg g(x) \vee \neg l(x, x))) \\ &\leftrightarrow \exists x(g(x) \wedge l(x, x)) \\ &\approx g(s) \wedge l(s, s) \\ &\leftrightarrow \{\{g(s)\}, \{l(s, s)\}\} \end{aligned} \tag{230}$$

- Applying the DP algorithm:

$$C_0 := \{\{\neg l(x, y), \neg l(x, z), \neg g(z)\}, \{g(s)\}, \{l(s, s)\}\}$$

$$\begin{aligned} C_1 &:= \text{res}(C_0, g(s)) \\ &= \{\{\neg l(x, y), \neg l(x, z), \neg g(z)\}, \{\neg l(x, y), \neg l(x, s)\}, \\ &\quad \{g(s)\}, \{l(s, s)\}\} \end{aligned}$$

$$\begin{aligned} C_2 &:= \text{res}(C_1, l(s, s)) \\ &= \{\{\neg l(x, y), \neg l(x, z), \neg g(z)\}, \{\neg l(s, z), \neg g(z)\}, \\ &\quad \{\neg l(x, y), \neg l(x, s)\}, \{\underline{\neg l(s, s)}\}, \{\underline{g(s)}\}, \{\underline{l(s, s)}\}\} \quad (231) \end{aligned}$$

- This proves that the conjecture is true.