

## SECTION 2: HOMEWORK SOLUTIONS

This section includes solutions to the homework problems in the course. Note that due to the nature of symbolic logic, there are many problems that can have multiple solutions, especially some of the world building problems in Tarski's World and several of the translation problems later in the course. Thus, these solutions are possible solutions to the problems, but there may be others. For some problems, such as proofs, there are multiple correct solutions presented here.

There are also several Tarski's World problems that require you to evaluate the truth value of a set of sentences; since these problems are designed to increase your understanding of the material and you can find the truth values in Tarski's World, they are not presented here. They are labeled as "Tarski's World Drills."

Problems are numbered  $c-p$ , where  $c$  is the chapter number and  $p$  is the problem number.

For Tarski's World problems where discerning the size of objects is important, the blocks are labeled as **S** (small), **M** (medium), or **L** (large).

### 2.1. Chapter 1 Solutions

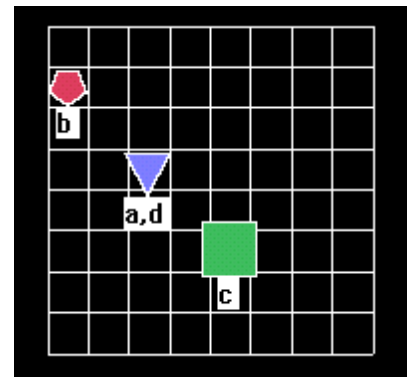
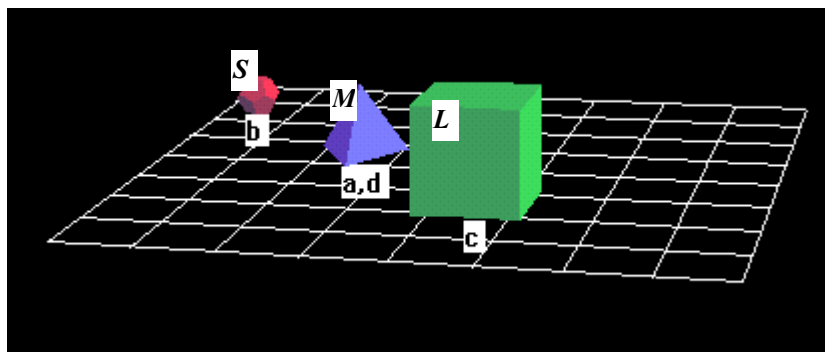
There are no homework problems in this chapter. Instead, read the chapter to get an introduction to what the course will entail as why logic is useful.

### 2.2. Chapter 2 Solutions

**Problem 2-1:** Tarski's World Drill

**Problem 2-2:** Solution looks identical to what is presented in the text.

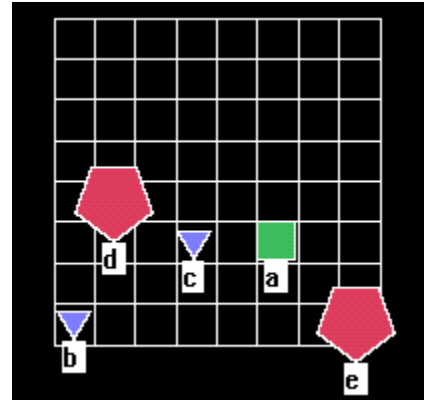
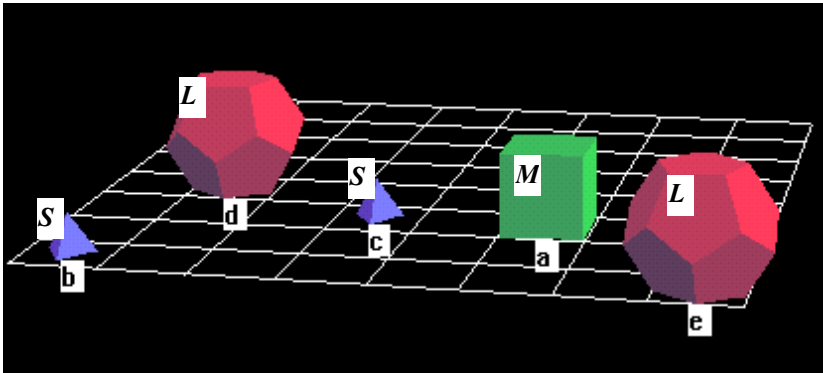
**Problem 2-3:** 2-3.wld:



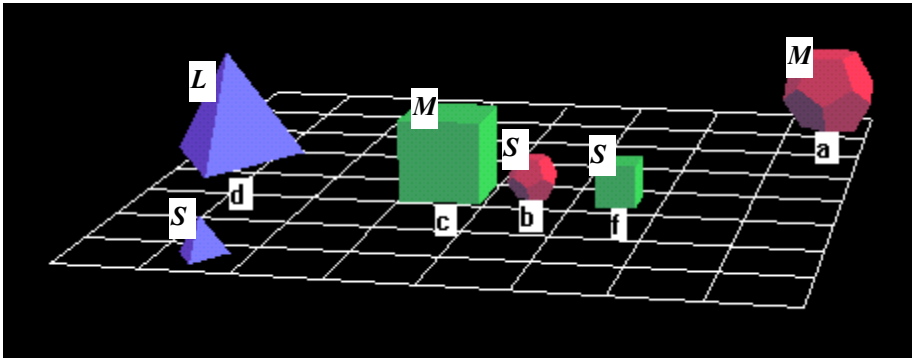
**Problem 2-4:** 2-4.sen:

- |                     |                  |
|---------------------|------------------|
| 1. Cube(a)          | 6. Tet(b)        |
| 2. Smaller(b, a)    | 7. Dodec(e)      |
| 3. Between(c, a, d) | 8. RightOf(e, b) |
| 4. Large(d)         | 9. Smaller(a, e) |
| 5. Larger(e, a)     | 10. BackOf(d, a) |

**Problem 2-4:** continued, 2-4.wld:



**Problem 2-5:** 2-5.wld



**Problem 2-9:**

Given the functional language and the relational language as follows:

	the relational language	the functional language
Names	Claire, Melanie, Jon	Claire, Melanie, Jon
Predicates	TallerThan(x, y) FatherOf(x, y): x is <u>the father of</u> y. x = y	TallerThan(x, y) x = y
Functions		father(x): <u>the father of</u> x

*Translations*

1. FatherOf(Jon, Claire)                  Jon = father(Claire)
2. FatherOf(Jon, Melanie)                Jon = father(Melanie)
3. TallerThan(Melanie, Claire)        TallerThan(Melanie, Claire)

*Which can be translated into atomic sentences of relational language?*

1. YES – FatherOf(Jon, Melanie)
2. NO – There are no function symbols, connectives, or quantifiers in the relational language. Unless an object is named, it is not possible to refer to it. To express this sentence, it is necessary to refer to father(Melanie) and father(Claire), which is not possible.
3. NO – There are no function symbols, connectives, or quantifiers in the relational language. Unless an object is named, it is not possible to refer to it. To express this sentence, it is necessary to refer to father(Claire) and father(Jon), which is not possible.

**Problem 2-10:**

One could use the following translation manual:

	English	FOL
Names	Carl, Sam, Mary	the same
Predicates	$x$ is the same as $y$ $x$ is greater than $y$	$x = y$ $x > y$
Functions	the height of $x$	height ( $x$ )

Some sample translations are:

- *Carl is taller than Sam.*                      height(Carl) > height(Sam)
- *Sam and Mary are the same height.*      height(Sam) = height(Mary)
- *Mary is shorter than Carl.*                 height(Carl) > height(Mary)

**Problem 2-14:**

Given the following two first order languages:

	Language 1		Language 2	
	English	FOL	English	FOL
<b>Names</b>	Max Claire	Max Claire	Max Claire Scruffy Carl	Max Claire Scruffy Carl
<b>Predicates</b>	$x$ gave Scruffy to $y$ $x$ gave Carl to $y$	GaveScruffy( $x, y$ ) GaveCarl( $x, y$ )	$x$ gave $y$ to $z$	Gave( $x, y, z$ )
<b>Functions</b>				

1. In the first language:

- GaveScruffy(Max, Max)      GaveCarl(Max, Max)
- GaveScruffy(Max, Claire)    GaveCarl(Max, Claire)
- GaveScruffy(Claire, Max)    GaveCarl(Claire, Max)
- GaveScruffy(Claire, Claire)   GaveCarl(Claire, Claire)

2. In the second language

- |                             |                                |                                 |                              |
|-----------------------------|--------------------------------|---------------------------------|------------------------------|
| Gave(Max, Max, Max)         | Gave(Max, Max, Claire)         | Gave(Max, Max, Scruffy)         | Gave(Max, Max, Carl)         |
| Gave(Max, Claire, Max)      | Gave(Max, Claire, Claire)      | Gave(Max, Claire, Scruffy)      | Gave(Max, Claire, Carl)      |
| Gave(Max, Scruffy, Max)     | Gave(Max, Scruffy, Claire)     | Gave(Max, Scruffy, Scruffy)     | Gave(Max, Scruffy, Carl)     |
| Gave(Max, Carl, Max)        | Gave(Max, Carl, Claire)        | Gave(Max, Carl, Scruffy)        | Gave(Max, Carl, Carl)        |
| Gave(Claire, Max, Max)      | Gave(Claire, Max, Claire)      | Gave(Claire, Max, Scruffy)      | Gave(Claire, Max, Carl)      |
| Gave(Claire, Claire, Max)   | Gave(Claire, Claire, Claire)   | Gave(Claire, Claire, Scruffy)   | Gave(Claire, Claire, Carl)   |
| Gave(Claire, Scruffy, Max)  | Gave(Claire, Scruffy, Claire)  | Gave(Claire, Scruffy, Scruffy)  | Gave(Claire, Scruffy, Carl)  |
| Gave(Claire, Carl, Max)     | Gave(Claire, Carl, Claire)     | Gave(Claire, Carl, Scruffy)     | Gave(Claire, Carl, Carl)     |
| Gave(Scruffy, Max, Max)     | Gave(Scruffy, Max, Claire)     | Gave(Scruffy, Max, Scruffy)     | Gave(Scruffy, Max, Carl)     |
| Gave(Scruffy, Claire, Max)  | Gave(Scruffy, Claire, Claire)  | Gave(Scruffy, Claire, Scruffy)  | Gave(Scruffy, Claire, Carl)  |
| Gave(Scruffy, Scruffy, Max) | Gave(Scruffy, Scruffy, Claire) | Gave(Scruffy, Scruffy, Scruffy) | Gave(Scruffy, Scruffy, Carl) |
| Gave(Scruffy, Carl, Max)    | Gave(Scruffy, Carl, Claire)    | Gave(Scruffy, Carl, Scruffy)    | Gave(Scruffy, Carl, Carl)    |
| Gave(Carl, Max, Max)        | Gave(Carl, Max, Claire)        | Gave(Carl, Max, Scruffy)        | Gave(Carl, Max, Carl)        |
| Gave(Carl, Claire, Max)     | Gave(Carl, Claire, Claire)     | Gave(Carl, Claire, Scruffy)     | Gave(Carl, Claire, Carl)     |
| Gave(Carl, Scruffy, Max)    | Gave(Carl, Scruffy, Claire)    | Gave(Carl, Scruffy, Scruffy)    | Gave(Carl, Scruffy, Carl)    |
| Gave(Carl, Carl, Max)       | Gave(Carl, Carl, Claire)       | Gave(Carl, Carl, Scruffy)       | Gave(Carl, Carl, Carl)       |

Thus, there are 64 possible sentences. Perhaps the number of expressions is given by  $n^a$  where  $n$  is the number of names and  $a$  is the arity.

3. The first language would need the same four names (Max, Claire, Scruffy, and Carl) and four predicates (GaveScruffy( $x, y$ ), GaveCarl( $x, y$ ), GaveMax( $x, y$ ), and GaveClaire( $x, y$ )) in order to be able to express everything that can be said in the second language.

**Problem 2-15:**

- Claire owned Folly at 2 p.m.
- Claire gave Silly to Max at 2:05 p.m.
- Max is a student.
- Claire erased Folly at 2 p.m.
- Folly belonged to Max at 3:05 p.m.
- 2:00 p.m. is earlier than 2:05 p.m.

Owned(Claire, Folly, 2:00)  
 Gave(Claire, Silly, Max, 2:05)  
 Student(Max)  
 Erased(Claire, Folly, 2:00)  
 Owned(Max, Folly, 3:05)  
 2:00 < 2:05

**Problem 2-16:**

- Owned(Max, Silly, 2:00)
- Erased(Max, Silly, 2:30)
- Gave(Max, Silly, Claire, 2:00)
- (2:00 < 2:00)

Max owned Silly at 2 p.m.  
 Max erased Silly at 2:30 p.m.  
 Max gave Silly to Claire at 2 p.m.  
 2 p.m. is earlier than 2 p.m.

**Problem 2-17:**

	English	FOL
<b>Names</b>	AIDS, influenza Spain, France, Portugal Misery, Company Max, Claire, John, Nancy Jon, Mary Ellen	AIDS, influenza Spain, France, Portugal Misery, Company Max, Claire, John, Nancy Jon, Mary Ellen
<b>Predicates</b>	$x$ is less contagious than $y$ $x$ is between $y$ and $z$ in size $x$ loves $y$ $x$ shook $y$ $x$ is younger than $y$	$x < y$ Between( $x, y, z$ ) Loves( $x, y$ ) Shook( $x, y$ ) Younger( $x, y$ )
<b>Functions</b>	the father of $x$ the eldest child of $x$ and $y$ $x$ 's hand	father( $x$ ) eldestChild( $x, y$ ) hand( $x$ )

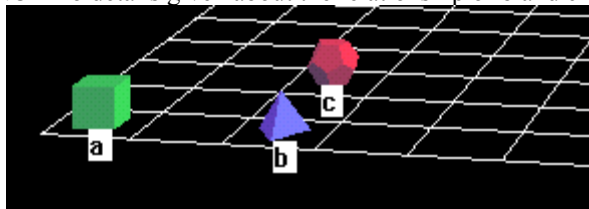
- AIDS < influenza
- Between(Spain, France, Portugal)
- Loves(Misery, Company)
- Shook(Max, hand(father(Claire)))
- Younger(eldestChild(John, Nancy), eldestChild(Jon, Mary Ellen))

**Problem 2-18:**

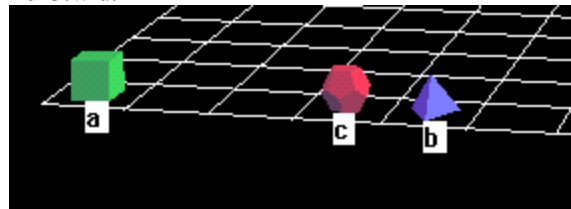
Proof: Suppose that  $a = b$  and  $b = c$  (two given premises). According to Ind Id, we can replace  $b$  in  $a = b$  by  $c$ . We come up with  $a = c$ , as desired.

**Problem 2-19:**

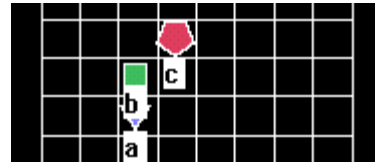
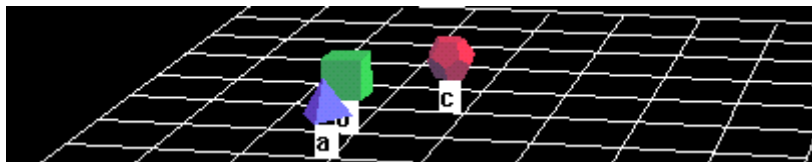
- YES – if  $a$  is to the left of  $b$ , it follows that  $b$  is to the right of  $a$
- YES – if  $a$  is to the left of  $b$ , and  $b = c$ , one can say  $a$  is to the left of  $c$ . It follows that  $c$  is to the right of  $a$ .
- NO – no details given about the relationship of  $b$  and  $c$  – see 2-19-3.wld:



OR



- YES – from the first statement, a front-to-back order is given as  $(b, a)$ . The second statement adds that  $a$  is in front of  $c$ , so  $c$  must be placed after  $a$  in the list:  $(b, a, c)$ . The conclusion holds true in this ordering.
- NO – see 2-19-5.wld:



**Problem 2-20:**

1. Owned(Claire, Folly, 2:00)
  2. Owned(Max, Silly, 2:00)
  3.  $\neg$ Owned(Claire, Silly, 2:00)
- If it is assumed that a disk can only have one owner at a given time, (3) follows from (1) and (2).
  - (2) does not follow from (1) and (3). These statements do not say anything about Silly's owner at 2 p.m. other than that it was not Claire. For example, let *Silly belong to Mary at 2 pm*. So, under this situation, (1) is true, (3) is true, but (2) is false.
  - (1) does not follow from (2) and (3). There is nothing stated about Folly and nothing that prevents Max from owning two disks, as in Owned(Max, Folly, 2:00). For example, let Folly and Silly belong to Max at 2 p.m. Then (2) is true and (3) is true, but (1) is false in this case.

**Problem 2-21:**

- |            |              |
|------------|--------------|
| 1. $a = b$ |              |
| 2. $b = c$ |              |
| 3. $a = c$ | Ind Id: 1, 2 |

**Problem 2-22:**

- |                |              |
|----------------|--------------|
| 1. Likes(a, b) |              |
| 2. $b = c$     |              |
| 3. $c = d$     |              |
| 4. $b = d$     | Ind Id: 2, 3 |
| 5. Likes(a, d) | Ind Id: 1, 4 |

**Problem 2-23:**

- |                     |              |  |
|---------------------|--------------|--|
| 1. Between(a, d, b) |              |  |
| 2. $a = c$          |              |  |
| 3. $e = b$          |              |  |
| 4. Between(c, d, b) | Ind Id: 1, 2 | (using property 1, replacing a with c – replace left of = with right of =) |
| 5. $e = e$          | Refl =       | (no e in 1 or 4 on the left, want it on the left to replace)               |
| 6. $b = e$          | Ind Id: 5, 3 |  |
| 7. Between(c, d, e) | Ind Id: 4, 6 | (using property 4, replacing b (left) with e(right) from 6)                |

**Problem 2-24:**

First, set up a rule: *Transitivity of Smaller (Trans. Sm)*

- |                   |                   |
|-------------------|-------------------|
| #m Smaller (a, b) |                   |
| :                 |                   |
| #n Smaller (b, c) |                   |
| :                 |                   |
| ◇ Smaller (a, c)  | Trans. Sm: #m, #n |

Then, the proof is:

- |                  |                 |
|------------------|-----------------|
| 1. Smaller(a, b) |                 |
| 2. Smaller(c, d) |                 |
| 3. $b = c$       |                 |
| 4. Smaller(a, c) | Ind Id: 1, 3    |
| 5. Smaller(a, d) | Trans. Sm: 4, 2 |

**Problem 2-25:**

First, set up a rule: *Left-Right Symmetry (L-R Sym)*

- |                  |            |
|------------------|------------|
| #m RightOf(a, b) |            |
| :                |            |
| :                |            |
| ◇ LeftOf(b, a)   | L-R Sym: m |

The proof is:

- |                  |              |
|------------------|--------------|
| 1. LeftOf(b, a)  |              |
| 2. $b = c$       |              |
| 3. LeftOf(c, a)  | Ind Id: 1, 2 |
| 4. RightOf(a, c) | L-R Sym: 3   |