

#### Exercise 4.60.

By giving the query

```
(lives-near ?person (Hacker Alyssa P))
```

Alyssa P. Hacker is able to find people who live near her, with whom she can ride to work. On the other hand, when she tries to find all pairs of people who live near each other by querying

```
(lives-near ?person-1 ?person-2)
```

she notices that each pair of people who live near each other is listed twice; for example,

```
(lives-near (Hacker Alyssa P) (Fect Cy D))  
(lives-near (Fect Cy D) (Hacker Alyssa P))
```

Why does this happen? Is there a way to find a list of people who live near each other, in which each pair appears only once? Explain.

#### Answer.


Alyssa's trouble arises from the fact that `lives-near` fails to get rid of symmetrical pairs while consulting the data base which makes the same fact match twice by the query system. So we must guarantee that symmetry on pairs be eliminated before querying. A plausible strategy is to stipulate a topological order among the employees. For example, we may assign each individual a unique ID:

```
(id (Warbuks Oliver) 001)  
(id (Aull Dewitt) 002)  
(id (Bitdiddle Ben) 011)  
(id (Hacker Alyssa P) 012)  
(id (Fect Cy D) 013)  
(id (Tweakit Lem E) 014)  
(id (Reasoner Louis) 015)  
(id (Scrooge Eben) 021)  
(id (Cratchet Robert) 022)
```

and modify `lives-near` to fetch entries of which pairs are assigned with designated order:

```
(rule (lives-near ?person-1 ?person-2)  
      (and (address ?person-1 (?town . ?rest-1))  
            (address ?person-2 (?town . ?rest-2))  
            (not (same ?person-1 ?person-2))  
            (id ?person-1 ?id-1)  
            (id ?person-2 ?id-2)  
            (lisp-value > ?id-1 ?id-2)))
```

---

\*. Creative Commons  2014, Lawrence X. Amlord (颜世敏, aka 颜序).  
Email address: informlarry@gmail.com