ROBBIE'S WORLD – THE KB RULES: If Robbie remembered (stored) ALL he knew as just facts, that could just be stored in a big database (or data file) – and computers do fairly well searching big DB's fast. But that would be very SPACE-consuming for "all the knowledge about his entire world" as well as very TIME-consuming to search through all those facts if this was a realistic sized real world problem. We also wouldn't think of Robbie as being very INTELLIGENT - just a fast, accurate, reliable, pre-programmed robot from DB-land, good at recalling (searching) simple facts he's just "memorized" (stored in his database). He also couldn't deal with other new arrangements of these objects without changing that DB.

So, Robbie needs RULES about how his world "works", what things "mean", common sense knowledge, things to take for granted, stuff "everybody knows", "laws of physics", etc. These KB RULES describe how any world of this sort in general works (which of course, incorporates how this particular configuration of objects works). Where ever possible, Robbie stores things as RULES rather than FACTS.

Knowledge makes Robbie intelligent. Knowledge is more efficient and useful than data. One rule (e.g., a formula) can suffice for 1000's of facts. Knowledge is about variables rather than constants – making it generic and reusable. For example, if you had a different DB (different collection of objects) or even just a different dynamic DB (same objects, but in a different configuration), then this KB must be able to work with that as well. It can be applied in completely new situations, making Robbie adaptable. Knowledge is generalizable – it allows Robbie to deduce new unstated knowledge or new facts. Robbie will also be more adaptable to humans if he knows about "common sense" terminology for synonyms, antonyms, slang and abstractions (and maybe some Spanish or Chinese?). So your KB needs to include RULES about:

1) more general RELATIONSHIPS based on any objects' RELATIONSHIPS:
   (somewhereLeft, below, clear, ... ETC.)
2) more general PROPERTIES based on objects' PROPERTIES:
   (material – since many objects in this world are largely based on their TYPE – e.g.,
   All marbles are made of glass.
   (hollow, bounces, breakable, crushable, transparent, . . . etc.)
   or combination of properties (flatOnTop, weight, . . . etc.)
3) SYNNONYMS ("same meaning") for various RELATIONSHIPS & PROPERTIES
   (including both specific and general) (left, under, tiny, huge, . . . etc.)
4) ANTONYMS ("opposite meaning") for various RELATIONSHIPS & PROPERTIES
   (specific and general) (directlyRight, somewhereRight, immediatelyRight, above, ... etc.)
5) terms for CATEGORIZING objects (an object vs. a thing, something that exists)
6) terms for more GENERALIZED properties based on objects' PROPERTIES: (dark/light, ...)
7) CAPABILITIES of objects based on their PROPERTIES and current state
   - CURRENT capabilities (canRoll, canPutSomethingOnIt, . . . etc.)
   [based on shape and dynamic property values right NOW, if that's relevant]
   - POTENTIAL capabilities (couldRoll, couldPutSomethingOnIt, . . . etc.)
   [based on shape, nevermind if it's currently in the right orientation and/or if it happens to currently have something on it already]
   [NOTE that for some shaped objects, current orientation is important to check, and for some, not
   e.g., cylinders vs. spheres]

KB RULES vs. QUERIES

Do NOT add rules to the KB for specific query-handling – e.g., if some query asks about "black granite", do NOT put a rule in the KB for:

```
blackGranite(X) :-
        color(X, black),
        material(X, granite).
```

Let the QUERY rule include specific individual properties:

```
question246(X) :-
        color(X, black),
        material(X, granite).
```

NOT

```
question246(X) :-
        blackGranite(X).
```

KB RULES – Special Handling for ROBBIE & THE TABLE? For every rule you write, decide if you have to specifically INCLUDE or specifically EXCLUDE Robbie and/or the table. Or, you might need additional rules for (i.e., an "or-rule") for handling Robbie and perhaps another one for the table. Sometimes Robbie and the table will just fit into the regular rules along with the regular objects. Make sure the queries give the right answer. And this should NOT be the responsibility of the query itself (to include/exclude Robbie and/or the table) – it's the responsibility of the "concept definition" in the KB. For example,

X has a flat side IF
its SHAPE is A cube or A cylinder or A disk.
theTable has a flat side (by it's very existence as a table, nevermind considering its shape)

CONCEPTUAL EFFECIENCY: As with human's storage, store knowledge EFFICIENTLY and ECONOMICALLY.

- store knowledge as RULES rather than FACTS where possible,
  e.g., things are flatOnTop if they have certain shapes
  rather than listing all the categories (aBlock, …) or all the names (theBlock, …)
- define rules in terms of other rules to shorten them;
  e.g., somewhereRight is just the opposite of somewhereLeft
  (write a detailed description of somewhereLeft, but then
  don't write a detailed description of somewhereRight)
- define rules in terms of more general rules rather than in terms of more specific rules
  e.g., things which have a rounded edge could roll rather than listing specific shapes

As with human knowledge, rules should be general enough so they'd work in new situations - i.e., with a different number of objects in many/any different configurations (e.g., 35 objects altogether, 15 objects across, some stacked 7 high, all prisms and balls, etc.) - by simply changing the facts.