Problem Set 3: Parsing and Frames

6.034 Fall 2003

`http://www.ai.mit.edu/courses/6.034f/`

Due: Wednesday, October 29th, 2003 – 11:59PM

You will need to copy the following files to your ps3 directory: `ps3.scm`, `ps3-publictest.scm`, `tester.scm`, `stt.scm`. These files can be downloaded off of the webpage or from the 6.034 Athena locker (`/mit/6.034/files/ps3`).

This problem set involves no Scheme coding in the sense of the previous problem sets. You will be asked to perform exercises, the answers to which should be placed in the appropriate places in the file `ps3.scm`*. Public test cases are provided that will test only some parts of each question. These test cases are provided to give examples of correct form but a perfect score on the public test cases does not necessarily mean that your problem set is completely correct. As in previous problem sets, the test cases can be run by executing the* `test-file` *function from within Scheme.* In most cases, questions of syntax or correct formation of solutions can be answered by looking at the public test cases or the documentation provided in `stt.scm`.

Ben Bitdiddle, frustrated from the long hours of work he put in while dealing with Louis' awful code, decided to quit Ur-Bits and look for a better job. Calling up his old pal Alyssa P. Hacker, he managed to get a good referral from her and was soon gainfully employed at a startup company in Cambridge working on question answering problems for the military.

Ben wasn't allowed to know about most of the system, since his clearance hadn't processed yet, but the parser and representation systems weren't classified, and his boss, Milly Speck, put him right to work on those.

"Ben, my boy," said Milly, "Alyssa said you're quite good at getting the most out of simple systems. That's good news for us, because this natural language processing is very hard and we need to get some demos out the door soon."

"One of the big things the military wants is easy access to its databases. If some low-level soldier wants to know something, they shouldn't have to know database language to do queries. Fortunately, most questions are fairly simple, so we can use semantic transition trees to turn easy questions directly into database queries."

"We need to get queries for the COUNTRIES database and the WEAPONS database up and running for a demo in two days. Unfortunately, our main STT designer is out right now for paternity leave. Can you do it?"

"Sure!" said Ben, frantically thinking back to 6.034. But to his relief, he found that things were fairly under control. The COUNTRIES database semantic transition tree was mostly done, although it had a few bugs and he was able to write the semantic transition tree for WEAPONS fairly easily.

Problem 1: Semantic Transition Trees

You are provided with an STT grammar for COUNTRIES that has three bugs. **Your task is to find and correct the three small bugs in the COUNTRIES STT grammar and to write an STT grammar for WEAPONS.** We expect that you will be able to do most of what you need for the WEAPONS grammar by copying and editing the COUNTRIES grammar. See the top of `stt.scm` for an explanation of how to format STT grammars.

Hint: *The changes you will need to make to fix the COUNTRIES STT grammar are small. One of the bugs is causing infinite loops: you will probably want to turn on verbose output and find that bug first.*

Your WEAPONS grammar should handle "what" and "get" questions like the COUNTRIES grammar, and all of the vocabulary in the WEAPONS sample.

Here are samples from the COUNTRIES and WEAPONS databases.

Country	**Military-Branch**	**Size**	**Morale**	**Equipment-Quality**
iraq	army	large	low	bad
iraq	navy	tiny	low	bad
us	army	large	medium	excellent
us	navy	large	high	good
us	airforce	large	high	excellent
israel	army	small	high	good

Weapon	**Country**	**Type**	**Size**	**Lethal**
weapon	country	type	size	lethal
m16	us	gun	personal	yes
uzi	israel	gun	personal	yes
m60	us	gun	squad	yes
80mm-howitzer	us	artillery	vehicle	yes
stun-grenade	britain	grenade	personal	no

And here are sample queries for the COUNTRIES database:

- in database COUNTRIES get us army morale
- in database COUNTRIES get the military sizes for iraq
- in database COUNTRIES get the equipment quality for iraq
- in database COUNTRIES what branches does the iraq military have
- in database COUNTRIES what size is the israel navy

Pleased with his work, Milly brought Ben a new assignment. “It looks like you really know what you’re doing. That’s good, because we’re going to need you to do some more STT work, on a more interesting part of the system.”

To Ben’s inquiry, she went on, “What you just did works, but only for the simplest sorts of sentences. It doesn’t know about context, and it can’t deal with most ambiguities. That’s fine when it’s a human talking to a computer, but that’s not what the military’s really interested in.”

“What they really want is to be able to understand what people are saying to each other. Not complicated conversations, but things like news stories. Our big contract is for a situational awareness system that takes in stories from thousands of newspapers all over the world, interprets them, and then acts like a research assistant for intelligence analysts. So if we want to know about tensions in Bolivia, we can just ask it for stories about tensions in Bolivia and it’ll give them to us.”

“Sounds like a good idea,” said Ben, “But what’s so hard about that? Can’t you just look to see whether the word ’tension’ is in a story?”

“Just looking at what words in a story turns out to give very little information about the story. You either end up missing important stories or getting back hundreds of irrelevant ones. Instead, we’re trying to build a system that actually understands the stories it reads. To do that, it turns the story into several different representations used by people and reasons collaboratively between them. Or at least that’s the plan — it’s rather hard to actually do.”

“So what’s my role?” asked Ben.

“You”, said Milly, “are going to build an STT system that translates between two different systems. The big new idea we have is to use English as an intermediate representation. To translate from one representation to another, we render the first representation into simple English, then parse it into the second representation: it simplifies the problem enormously and gives us human-readable output for debugging. I need you to build an STT system for our transition frame to trajectory frame connector.”

“First though, I’d better tell you about our two representations.”

Milly proceeded to explain transition frames and trajectory frames to Ben, along with several examples to help him understand how they worked.

Problem 2: Transition Frames

Transition frame representations take the form of "ladder diagrams." Each column in the ladder corresponds to a step in time (every row of a particular ladder should have the same number of time steps). Each row of the ladder corresponds to the transitions pertaining to a particular relationship.

The specific syntax that we will be using in this problem set is the following:

- The entire ladder is a `transition-space`. This object is a list, the `car` of which is the symbol `transition-space` and the `cdr` of which is a list of `transition-rows`.
- Each `transition-row` is a list, the first element of which is a `transition-relation`, and the second element of which is a `transition-sequence`.
- Each `transition-relation` is a list, the first element of which is a symbol representing the type of relationship being described, and subsequent elements of which are symbols representing the names of the things involved in the relation.
- Each `transition-sequence` is a list of symbols from the set of possible transitions, shown in Table 1.

Transition type	**Symbol to use**
Change	`cha`
Not Change	`ncha`
Increase	`inc`
Not Increase	`ninc`
Decrease	`dec`
Not Decrease	`ndec`
Disappear	`dis`
Not Disappear	`ndis`
Appear	`app`
Not Appear	`napp`

Table 1: Table of typical transitions and the corresponding symbols used in the code

For example, an event such as, "The distance between the ball and the wall decreased and then disappeared while the speed of the ball did not change and then disappeared" would be represented in list structure as the following:

```
'(transition-frame
  ((dist ball wall) (dec dis))
  ((speed ball) (ncha dis)))
```

For this question you are to represent sequences of events as a transition frames using the syntax described above. For each of the following three events, fill in the appropriate definition in `ps3.scm` so that the expression returns the `transition-frame` that you feel most accurately describes the given sequence of events.

Each object and relation type should be a symbol from the following list: `us-army, us-airforce, iraqi-army, israeli-army, quality, morale, size, rank, speed, us-army-general, distance, us-planes, altitude, us-army-equipment`.

Event A:

The US Army grew larger before reaching a steady-state size. Meanwhile, the quality of the US Army's equipment continually declined.

Event B:
Immediately after a sharp decrease in the morale of the US Army, a general of the US Army was demoted.

Event C:
As the US Air Force approached the Iraqi Army, the US planes descended for a while and then leveled out.

Problem 3: Trajectory Frames

For the purposes of this problem, we will be using a simplified version of Jackendoff's *Lexical Conceptual Semantics* representation for our trajectory frames.

- Each event is described by one or more `trajectory-frame` objects.
- Each `trajectory-frame` is a list, the first element of which is the symbol `trajectory-frame`, the second element of which is a symbol representing the object that is moving along this particular trajectory, and the subsequent elements of which are `trajectory-paths` defining the path(s) that the object moves along.
- Each `trajectory-path` is a list, the first element of which is the symbol `path`, the second element of which is a symbol defining the type of path (`to, toward, from, away-from`), and the third element of which is a `trajectory-place`.
- Each `trajectory-place` is a list, the first element of which is the symbol `place`, the second element of which is a symbol defining the type of place (`at, on, under, above, in`), and the third element of which is a symbol representing the particular object that is used to define the place.

For example, an event such as, "The ball went from being on the table to being in the basket" would be represented in list structure as the following:

```
'(trajectory-frame
   ball
   (path from (place on table))
   (path to (place in basket)))
```

For this question you are to represent sequences of events as trajectory frames using the syntax described above. For each of the following three events, fill in the appropriate definition in `ps3.scm` so that the expression returns a list of one or more `trajectory-frames` that you feel most accurately describe the sequence of events. Any symbol that may be needed to define an object or place is in bold.

Event A:
Sam walked from his **office** to the **watercooler** and then over toward the **window**.

Event B:
The **boy** moved further away from the **spider** and crawled under the **table**.

Event C:
Sam, after making his way over to the **window**, stood and watched while outside the **birds** flew playfully from the **windowsill** to the **birdbath**.

Once Milly had explained the representations, she gave Ben some sample sentences to work with, and sent him off to work on his parser. Ben quickly realized that this new problem wasn't very different than before, and set to work coding up a tree to parse English into trajectory frames. The top of the tree and all the terminals of the tree were easy, but when Ben got to the middle, he was stumped.

Panicked, he called Alyssa. “You silly,” she said, “The expressions don’t have to be database queries. They can be any arbitrary sort of expression you want.”

Sheepishly, Ben thanked Alyssa and went back to work, quickly filling in the remaining parts of the problem.

Problem 4: Semantic Transition Trees and Frames

Ben’s tree is supposed to take in English sentences and produce trajectory frames.

Complete Ben’s trajectory tree grammar. You will need to fill in expressions for the trees `?trajectory`, `?path`, `?path-element`, and `?place`.

Problem 5: Topological Sorting

Consider the four graphs shown below. **Using topological sorting, provide a linearization of each hierarchy.** Use alphabetical ordering when the sub-before-super and left-before-right constraints are not enough to determine the next element in your linearization.

For each hierarchy, fill in the appropriate definition in `ps3.scm`. Each definition should return a list, the elements of which correspond to the letters of the nodes in the hierarchy (in proper linearized order). If no linearization exists, an empty list should be returned.

1:

$$\begin{tikzcd}
 & & & E & & & \\
 & F \arrow[urr] & & & & G \arrow[ull] & \\
D \arrow[ur] & & E \arrow[ul] & & D \arrow[ur] & & E \arrow[ul] \\
 & B \arrow[ul] \arrow[ur] & & & & C \arrow[ul] \arrow[ur] & \\
 & & & A \arrow[ull] \arrow[urr] & & &
\end{tikzcd}$$

2:

$$\begin{tikzcd}
 & & & E & & & \\
 & F \arrow[urr] & & & & G \arrow[ull] & \\
D \arrow[ur] & & E \arrow[ul] & & E \arrow[ur] & & F \arrow[ul] \\
 & B \arrow[ul] \arrow[ur] & & & & C \arrow[ul] \arrow[ur] & \\
 & & & A \arrow[ull] \arrow[urr] & & &
\end{tikzcd}$$

3:

$$\begin{tikzcd}
 & & & E & & & \\
 & F \arrow[urr] & & & & G \arrow[ull] & \\
X \arrow[ur] & & Y \arrow[ul] & & L \arrow[ur] & & M \arrow[ul] \\
 & B \arrow[ul] \arrow[ur] & & & & C \arrow[ulllll] \arrow[ul] \arrow[ur] & \\
 & & & A \arrow[ull] \arrow[urr] & & &
\end{tikzcd}$$

4:

$$\begin{tikzcd}
 & & & E & & & \\
 & F \arrow[urr] & & & & G \arrow[ull] & \\
X \arrow[ur] & & Y \arrow[ul] & & L \arrow[ur] & & M \arrow[ul] \\
 & B \arrow[ul] \arrow[ur] \arrow[urrrrr] & & & & C \arrow[ulllll] \arrow[ul] \arrow[ur] & \\
 & & & A \arrow[ull] \arrow[urr] & & &
\end{tikzcd}$$