2009 Solutions

(G) Sk8r

Languages are everywhere… even in places where you don't expect them.

Consider the "combo rules" of *P-Little's Triple-I XTreem Hyp0th3tica7 Sk8boarding Game*. In it, players press a series of buttons (left, right, down, up, circle, triangle, square, and X) to make their on-screen avatar perform skateboard tricks that illustrate pro boarder P-Little's "Triple-I" philosophy of Insane, Ill-Advised, and Impossible According to the Laws of Physics.

Underneath, the game is using the methods of computational linguistics to turn this "little language" of button presses into tricks and combos. The game uses a simple ***shift-reduce parser*** to parse button "words" into combo "sentences".

As each button-press comes in, the corresponding symbols are placed, in order, in a buffer:

1. ↑
2. ↑ ←
3. ↑ ← ⊡
4. ↑ ← ⊡ ⊗

If, at any point, the *rightmost* symbols in this buffer match any of the patterns on the next page, they are removed and replaced with a new symbol indicating a combo. So, since SX corresponds to an "ollie", we replace it with the new symbol **Ollie**.

5. ↑ ← **Ollie**
6. ↑ ← **Ollie** ⊡
7. ↑ ← **Ollie** ⊡ ⊡
8. ↑ ← **Ollie** ⊡ ⊡ ⊗
9. ↑ ← **Ollie** ⊡ **Ollie**

(G) Sk8r

More complex combos can then be built out of simpler combos. You see in rule (e) below that **Ollie** and **Nollie** can be joined by S to make a new combo. There are also ***rule schemas*** that can create new combos out of *any* kind of combo. Rule (j) below says that *any* combo (represented by a), whether it's an Ollie or an Inverted-360-Kickflip, can be joined with itself by a S to make a Double combo:

10. ↑ ← **Double-Ollie**

	If the right side of the input matches…		*Replace it with…*
a.	← ↑ △	⇨	**Backside-180**
b.	→ ↓ ◎	⇨	**Frontside-180**
c.	□ ⊗	⇨	**Ollie**
d.	⊗ □	⇨	**Nollie**
e.	**Nollie** □ **Ollie**	\ ⇨	**Woolie**
f.	↓ ↓	⇨	**Crouch**
g.	**Backside-180 Frontside-180**	⇨	**Backside-360**
h.	**Crouch Backside-360**	⇨ v	**360-Kickflip**
i.	↓ **a** ↑	⇨	**Inverted-a**

2009 Solutions

(G) Sk8r

j. **a □⃝ a** ⇨ **Double-a**

k. **Double-a □⃝ a** ⇨ **Triple-a**

l. **Double-a □⃝ Double-a** ⇨ **Quadruple-a**

m. **a □⃝ Inverted-a** ⇨ **Atomic-a**

Complex combos can get pretty involved. Here are a few combos from the manual to give you an idea:

Inverted-Nollie: ↓⊗□⃝↑
Double-Inverted-Woolie: ↓⊗□⃝□⃝□⃝⊗↑□⃝↓⊗□⃝□⃝□⃝⊗↑
Inverted-Triple-Backside-180: ↓←↑△⃝□⃝←↑△⃝□⃝←↑△⃝↑
Atomic-Double-Frontside-180: →↓◎□⃝→↓◎□⃝↓→↓◎□⃝→↓◎↑
Inverted-Backside-360: ↓←↑△⃝→↓◎↑
Triple-360-Kickflip: ↓↓←↑△⃝→↓◎□⃝↓↓←↑△⃝→↓◎□⃝↓↓←↑△⃝→↓◎

2009 Solutions

(G) Sk8r

1. How would you perform an “Inverted-Atomic-Backside-360”?

↓←↑△⃝→↓◎□⃝↓←↑△⃝→↓◎↑↑

2. How about an “Atomic-Atomic-Ollie”?

□⃝⊗□⃝↓□⃝⊗↑□⃝↓□⃝⊗□⃝↓□⃝⊗↑↑

3. The shift-reduce rules given on the other page are incomplete. Using the descriptions of advanced combos in the manual, can you fill in the missing pieces? State them as concisely as possible.

4. During playtesting, the testers discover that even though combos like “Quadruple-Ollie” and “Quadruple-Inverted-Woolie” are listed in the manual, the game can never actually recognize any Quadruple combo that the player performs. Why not? How could you fix the game so that it can?

Consider the sequence of button presses that would make up a Quadruple-Ollie (or Quadruple-anything).

□⃝⊗□⃝□⃝⊗□⃝□⃝⊗□⃝□⃝⊗

Considering each button in turn, the parser first turns the first two symbols into an Ollie, then that and the subsequent Ollie into a Double-Ollie:

1. □⃝
2. □⃝ ⊗
3. Ollie
4. Ollie □⃝
5. Ollie □⃝ □⃝
6. Ollie □⃝ □⃝ ⊗
7. Ollie **S** Ollie
8. Double-Ollie

(G) Sk8r

When the parser comes across the next Ollie, it then combines it with the previous Double-Ollie to make a Triple-Ollie

9. Double-Ollie ⊡
10. Double-Ollie ⊡ ⊡
11. Double-Ollie ⊡ ⊡ ⊗
12. Double-Ollie ⊡ Ollie
13. Triple-Ollie

However, there's no way to turn a Triple-Ollie into a Quadruple-Ollie. You can never get a sequence that runs Double-Ollie S Double-Ollie, because the first half of the second Double-Ollie would have already combined with the previous symbols to create a

14. Triple-Ollie ⊡
15. Triple-Ollie ⊡ ⊡
16. Triple-Ollie ⊡ ⊡ ⊗
17. Triple-Ollie ⊡ Ollie

In order to make a Quadruple-Ollie possible, then, we should rewrite the Quadruple-a rule so that

1. Triple-**a** ⊡ **a** ⇒ **Quadruple-a**

5. What other types of combinations of the listed combos can never actually be pulled off by the player, and why not?

There are a large (in fact, infinite) number of possible combos that the parser can never actually parse, due to the fact that it recognizes some sub-sequence of the combo as a different combo and reduces it, rendering that sub-sequence unusable by the original rule.

(G) Sk8r

For example, you can never perform a Double-Nollie (or any further iteration of Nollies), because the parser recognizes a spurious Ollie inside of it:

⊗ ⊡ ⊡ ⊗ ⊡ Þ NOllie Ollie ⊡

Likewise, *any* Inverted-Inverted-a, as well as anything built on it (like an Atomic-Inverted-a), will fail, because the parser always recognizes two consecutive 4s as a Crouch:

↓ ↓ **a** ↑ ↑ Þ crouch **a** ↑ ↑

The same goes for any sort of Inversion of a Crouch or any move beginning in a Crouch, such as the Inverted-360-Kickflip. The first **4,** which should be part of the Inversion part, is instead reduced along with the first 4 of the Crouch to make a spurious Crouch, and the leftovers are interpreted incorrectly as an Inverted-Backside-360:

↓↓↓←↑◬→↓◎↑ Þ crouch ↓←↑◬→↓◎↑
Þ crouch ↓ Backside-360 ↑
Þ crouch Inverted-Backside-360

(H) LinearB

	𐀒 𐀜 𐀰	ko-no-so
	𐀀 𐀖 𐀛 𐀰	a-mi-ni-so
	𐀞 𐀂 𐀵	pa-i-to
	𐀶 𐀪 𐀰	tu-ri-so
H1	𐀓 𐀈 𐀛 𐀀	ku-do-ni-a
	𐀀 𐀞 𐀲 𐀷	a-pa-ta-wa
	𐀬 𐀑 𐀵	ru-ki-to
	𐀄 𐀲 𐀜	u-ta-no
	𐀓 𐀠 𐀪 𐀍	ku-pi-ri-jo
H2	𐀶 𐀛 𐀊	tu-ni-ja

(H) LinearB

H3

𐀀	a
𐀈	do
𐀂	i
𐀊	ja
𐀍	jo
𐀑	ki
𐀒	ko
𐀓	ku
𐀖	mi
𐀛	ni
𐀜	no
𐀞	pa
𐀠	pi
𐀪	ri
𐀬	ru
𐀰	so
𐀲	ta
𐀵	to
𐀶	tu
𐀄	u
𐀷	wa

(H) LinearB

	𐀒 𐀷	‘girl’	ko-wa
	𐀞 𐀲	‘all’	pa-ta
H4	𐀵 𐀰	‘this’	to-so
	𐀓 𐀖 𐀜	‘cumin’	ku-mi-no
	𐀪 𐀜	‘linen’	ri-no

2009 Solutions

(I) BgAdj

Below are phrases in Bulgarian and their translations into English:

	Bulgarian phrase	**English translation**
1	červẹni yạbəlki	red apples
2	kọsteni iglị	bone needles
3	studẹni napịtki	cold drinks
4	dosạdni decạ	boring children
5	obiknovẹn čovẹk	ordinary person
6	gnẹvni dụmi	angry words
7	červẹn plod	red fruit
8	lẹnen plat	linen fabric
9	sọčni plodovẹ	juicy fruits
10	kọžni zabolyạvaniya	skin diseases
11	gnẹven sədiyạ	angry judge
12	rịbeni kyuftẹta	fish croquettes
13	kirpịčeni kə̣šti	adobe houses
14	kọženi rəkavịci	leather gloves
15	lẹsen ịspit	easy exam
16	cẹnni knịgi	precious books
17	sọčen grẹypfrut	juicy grapefruit
18	cẹnen predmẹt	precious object

In Bulgarian, the adjectives agree in number with the nouns they qualify. Thus, a noun in plural is accompanied by an adjective in plural, and a noun in singular is accompanied by an adjective in singular.

The formation of the plural of the nouns is complex and cannot be deduced from the data presented in this problem. The specific noun plural form is not relevant to the formation of the plural of the adjective which accompanies it.

(I) BgAdj

I. The problem presents only a partial picture of the formation of the plural of the adjectives. The three rules could be formulated in various ways, essentially equivalent to the the following:

A.If the singular ends in **–ẹn** (stressed), then just add **–i**: **červẹn** : **červẹni**. (15%)

B.If the adjective indicates from what matter the noun is made, then just add **–i**: **kọsten** : **kọsteni**. (25%)

C.In all other cases, drop the final **e** and add **–i** : **gnẹven** : **gnẹvni**. (10%)

II. In brackets, after each adjective, the rule symbol is given (this will vary according to the order in which the rules are listed!):

19	**obiknovẹni** (A) procedụri	ordinary procedures
20	**lẹsni** (C) urọci	easy lessons
21	**rịbni** (C) restorạnti	fish restaurants
22	**kọstni** (C) zabolyạvaniya	bone diseases
23	**lẹneni** (B) čaršạfi	linen sheets

For each correctly formed plural, 10% of the full score awarded. Nothing accorded if error whatsoever; score awarded if there is misspelling irrelevant to the problem, e.g. **obikoveni** instead of **obiknoveni**.

(J) HypoHmongdriac

1. ___ be lost
2. ___ beef
3. ___ beverage
4. ___ bovine* livestock
5. ___ chicken (the animal)
6. ___ dog (the animal)
7. ___ filthy animals; filth
8. ___ filthy language
9. ___ flesh; meat
10. ___ hurt
11. ___ internal organs; soul
12. ___ language
13. ___ liver (the organ)
14. ___ livestock
15. ___ lose heart (“liver”); lose one’s wits; panic
16. ___ lose life to water; drown
17. ___ lose money (“silver”)
18. ___ lungs
19. ___ money
20. ___ small, non-bovine livestock
21. ___ pig (the animal)
22. ___ poetic genre (“money-language”)
23. ___ silver
24. ___ suffer from a headache (“brain-ache”)
25. ___ suffer from grief (“liver-ache”)
26. ___ suffer from lung disease (“lung-ache”)
27. ___ water
28. ___ water-buffalo liver
29. ___ wealth
30. ___ whisky
31. ___ young female
32. ___ young sow

(J) HypoHmongdriac

Solution:
28, be lost; 17, beef; 6, beverage; 15, bovine livestock; 13, chicken; 10, dog; 12, fithy animals; 23, filthy language; 18, flesh; 32, hurt; 3, internal organs; 24, language; 1, liver; 16 livestock; 25, lose heart; 27, lose life to water; 26, lose money; 2, lungs; 8, money; 14, small livestock; 11, pig; 22, poetic genre; 7, silver; 30, suffer from a headache; 29, suffer from grief; 31, suffer from lung disease; 4, water; 21, water-buffalo liver; 9, wealth; 5, whisky; 20, young female; 19, young sow

To solve this problem, it is important to realize that both of the two collections of words can be seen as networks, where words are connected by hyponymy relationships, and that these two networks must have equivalent shapes. However since "matching up" a whole network (or "graph") of this kind with another is difficult even for a computer, solving this problem requires noting that the graphs are largely composed of smaller graphs with a tree-like shape. These are much simpler to deal with.

For example, you might observe that there are exactly two components of the graph where three words are hyponymns of a single word (like a tree with three branches) for both the Hmong and English collections. This allows you to infer that 25-28 and 29 -32 must be either 'be lost' and the 'lose' words or 'hurt' and the 'suffer' words. You can determine how to match them by noting that only one of the roots in the Hmong words does not occur elsewhere (*hlwb*) and that only one of the English meanings does not occur elsewhere ('brain'). This suggests that 29-32 must be the 'hurt/suffer' group and 25-28 must be the 'lost/lose' group. Furthermore, since *sab* occurs in both of these groups, and since 'liver' occurs in both groups, *sab* must be 'liver', *sab-twm* must be 'water-buffalo liver' and *twm* must mean 'water buffalo'.

For example, you might observe that there are exactly two components of the graph where three words are hyponymns of a single word (like a tree with three branches) for both the Hmong and English collections. This allows you to infer that 25-28 and 29 -32 must be either 'be lost' and the 'lose' words or 'hurt' and the 'suffer' words.

(J) HypoHmongdriac

You can determine how to match them by noting that only one of the roots in the Hmong words does not occur elsewhere (*hlwb*) and that only one of the English meanings does not occur elsewhere ('brain'). This suggests that 29-32 must be the 'hurt/suffer' group and 25-28 must be the 'lost/lose' group. Furthermore, since *sab* occurs in both of these groups, and since 'liver' occurs in both groups, *sab* must be 'liver', *sab-twm* must be 'water-buffalo liver' and *twm* must mean 'water buffalo'.

This will lead you to the livestock tree in 12-16 and the realization that Hmong compounds are of at least two types. In one type, the meaning of the whole is the meaning of the first part modified by the second part (as in *sab-twm*). In the second type, the meaning of the whole is a general category including the meaning of both parts (that is, both parts are hyponymns of the whole). Knowing that *twm* is 'water buffalo' should allow you to guess that *nyuj-twm* is 'bovine livestock' since 'water buffalo' is a hyponym of only 'livestock' and 'bovine livestock', 'bovine livestock' is a hyponymn of 'livestock' and *nyuj-twm* is a hyponymn of *qab-npua-nyuj-twm*. We can now see that 3, 6, 9, and 12 are all compounds of the second type, and reason from what is known about their parts that 3 and 6 must be 'internal organs; soul' and 'beverage'. We see that 12 must be 'filthy animal; filth' since it occurs embedded inside of a type one compound that can only mean 'filthy language' (23). Therefore, 14 must be 'small, non-bovine livestock'.

By applying similar logic to the remaining cases, you will arrive at the answer given above.

(K) Dyirbal

Word orders:

- SOV – in case the subject is a pronoun
- OSV – in case the subject is not a pronoun

Pronouns:

- ŋinda = you
- ŋađa = I

Definite articles:

- bayi, placed before subjects of intransitive verbs and objects of transitive verbs
- baŋgul, placed before subjects of transitive verbs.

The suffix -ŋgu is placed on the subject of transitive verbs, when the subject is not a pronoun.

Assignment 1. Give the English translations:

bayi ñalŋga banagañu = **The boy returned.**
bayi yaṛa baŋgul yuṛiŋgu walmbin. = **The kangaroo waked the man.**
ŋinda bayi yuṛi buṛan. = **You saw the kangaroo.**

Assignment 2. Give the Dyirbal translations:

You sat. = **ŋinda ñinañu**
I caught the kangaroo. = **ŋađa bayi yuṛi ñiman.**
The father waked the man = **bayi yaṛa baŋgul ŋumaŋgu walmbin.**

(L) YakDuDray

L1. (16/25 points: two points for each correct match except for the blank)

1	2	3	4	5	6	7	8	9
D	E	H	B	C	A	G	F	_

Intermediate data:

A. Kuvi 8-2=6
B. Albanian 10-6=4
C. Farsi 8-3=5
D. Irish 6-5=1
E. Nepali 6-4=2
F. Yiddish 9-1=8
G. Pengo 10-3=7
H. Lithuanian 8-5=3

(5x4)+(9x8)=(5x10)+(6x7)

L2. (9/25 points)
Sample "key insights"

- closer connections among neighboring languages
- consonants more likely to be preserved
- pronunciation may not match spelling
- specific phonological changes, e.g., s-sh, c-p,
- specific patterns for numerals, e.g., 9 starts with N, 4 has T+R in the middle
- use of the title of the problem
- use of the equation
- use of the constraints imposed by the subtractions
- the form for the number 1 changes the most

(M) Orwellspeak

M1. Here is the revised grammar. The changes are relatively small.

Sentence -> PosNounPhrase + Verb + NegNounPhrase
Sentence -> NegNounPhrase + Verb + PosNounPhrase

PosNounPhrase -> PosAdjective + Noun
PosNounPhrase -> PosAdjective + PosNounPhrase
NegNounPhrase -> NegAdjective + Noun
NegNounPhrase -> NegAdjective + NegNounPhrase

Noun -> people
Verb -> love
PosAdjective -> good
PosAdjective -> charming
PosAdjective -> happy
NegAdjective -> bad
NegAdjective -> obnoxious
NegAdjective -> unhappy

Notice that in this grammar, a single Noun does not qualify as a PosNounPhrase or NegNounPhrase. This ensures that the false statement "people love good people" is ungrammatical, since "people" is not a NegNounPhrase.

M2. Could it help to list 1-word bad phrases? No. You can't list any of the 8 vocabulary words without ruling out some legal sentences. (And there is no point in listing words outside that vocabulary, since they will have no effect and you were were asked to keep your list as short as possible.)

How about 2-word bad phrases? There are 25 types of 2-word phrases: the first word can be from any of the 5 categories {START, Noun, Verb, PosAdjective, NegAdjective}, and the second word can be from any

(M) Orwellspeak

of the categories {Noun, Verb, PosAdjective, NegAdjective, END}. Of these 25 types, the following 15 types can never appear in a legal sentence, so we list them as bad phrases:

START Noun (1)
START Verb (1)
START END (1)

Noun Noun (1)
Noun PosA (3)
Noun NegA (3)

Verb Noun (1)
Verb Verb (1)
Verb END (1)

PosA Verb (3)
PosA NegA (9)
PosA END (3)

NegA Verb (3)
NegA PosA (9)
NegA END (3)

The *remaining* 10 types are depicted by the 10 arrows in this graph:
[insert bigram.png here]

By allowing only those 10 types of 2-word phrases, the device so far allows any sentence that corresponds to a path in the graph. Now, where does that leave us? As you can see, this already ensures that

* START must be followed by one or more Adjectives of the same type, and then a Noun. In other words, START must be followed by a PosNounPhrase or NegNounPhrase.

(M) Orwellspeak

* Such a PosNounPhrase or NegNounPhrase may be followed by END, or else may be followed by a Verb and another PosNounPhrase or NegNounPhrase.

However, this still permits illegal utterances like

A1. good people (not a sentence)
B1. good people love good people (not true)
C1. good people love bad people love good people (not a sentence)

and similarly

A2. good charming people
B2. good charming people love good charming people
C2. good charming people love bad obnoxious people love good charming people

We can get rid of some of the A. sentences with the 4-word bad phrases

START PosA Noun END (3)
START NegA Noun END (3)

This is only able to get rid of the shortest A. utterances, such as A1. We would need longer bad phrases to get rid of A2., since every 4-word subsequence of A2. can be part of a legal sentence. No finite list of bad phrases can get rid of all the A. utterances -- even with an upgraded device that allowed 1000-word bad phrases, we would not be able to censor extremely long A. utterances.

Similarly, we can get rid of some of the C. sentences with the 4-word
bad phrases

Verb PosA Noun Verb (3)

(M) Orwellspeak

Verb NegA Noun Verb (3)

Again, this is only able to get rid of the shortest C. utterances, such as C1. We would need longer bad phrases to get rid of C2., and no finite list could get rid of all the C. utterances.

However, we can get rid of *all* of the B. utterances with only the 4-word bad phrases

PosA Noun Verb PosA (9)
NegA Noun Verb NegA (9)

These require successive noun phrases to be of opposite polarity. They work on noun phrases of *any* length, by requiring the first phrase's last adjective to oppose the second phrase's first adjective. For example, we are able to censor B2. because it contains "... charming people love good ..."

The total number of bad phrases above is 73.

M3. Yes. It fails to censor A2. and C2. above.

M4. A single 1-word bad phrase will satisfy the government's stated needs by censoring everything:
START
Or they could use
END

(Or if the device can handle allow 0-word bad phrases, then the single 0-word phrase "" will also censor everything, as it is contained in any utterance; think about it!)

(M) Orwellspeak

You may be interested in some connections to computational linguistics:

* Problem M1 asked you to write a tiny context-free grammar. It is possible to write large context-free grammars that describe a great deal of English or another language. Although the "Opposites Attract" setting was whimsical, you could use similar techniques to ensure that plural noun phrases are not the subjects of singular verbs, and -- for many languages -- that plural noun phrases only contain plural adjectives.

* Problem M2 asked you to approximate the context-free grammar by what is called a 3rd-order Markov model, meaning that the model's opinion of the legality or probability of each word depends solely on the previous 3 words. (That is, the model only considers 4-word phrases.) The graph shown partway through the solution depicts a 1st-order Markov model (which considered only 2-word phrases).

* Problem M3 showed that the Markov model was only an approximation of the context-free grammar -- it did not define exactly the same set of legal sentences. The solution further noted that *no* nth-order Markov model could exactly match this context-free grammar, not even for every large n.

If you know about regular expressions, you may have noticed that the following regular expression *would* be equivalent to the context-free grammar, hence would do a perfect job of censorship.

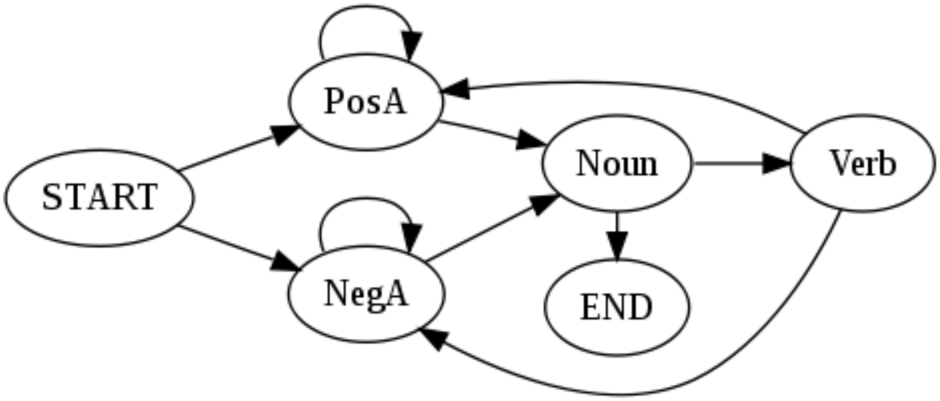
```
  START (  ((PosA)+ Noun Verb (NegA)+ Noun)
        | ((NegA)+ Noun Verb (PosA)+ Noun)  ) END
```

Regular expressions or regular grammars are equivalent to finite-state machines. They are not as powerful as context-free grammars in

(M) Orwellspeak

general, but they are powerful enough to match the "Opposites Attract" grammar. They are essentially equivalent to hidden Markov models, an important generalization of Markov models.

* Problems M3 and M4 together were intended to make you think about how to measure errors. In general, a system that tries to identify bad sentences (or bad poetry or email spam or interesting news stories) may make two kinds of errors: it may identify too many things or too few. Both kinds of errors are bad, and there is a tradeoff: you can generally reduce one kind at the expense of the other kind. The original requirement in problem M2 was to completely avoid the first type of error (i.e., never censor good stuff) while simultaneously trying to avoid the second type of error (censor as much bad stuff as possible). But the revised requirement in problem M4 considered only the second type of error, giving the vendor an incentive to design a dumb system that did horribly on the first type of error. You might conclude that when evaluating a vendor's system or setting requirements for it, you should pay attention to both kinds of error.


PosA
START
NegA
Noun
Verb
END