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**

2009 Solutions

(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**

(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: ↓↓←↑△⃝→↓◎□⃝↓↓←↑△⃝→↓◎□⃝↓↓←↑△⃝→↓◎

(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** $\Rightarrow$ **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