Sour Grapes is phonotactically complex

Andrew Lamont · University of Massachusetts Amherst · alamont@linguist.umass.edu
LSA 2019 · January 3

Sour Grapes is an unattested spreading/harmony process. Its non-existence has been argued to reflect its computational complexity as a function (Gainor et al., 2012; Heinz & Lai, 2013). I argue against this view, and show that Sour Grapes is not exceptional as a function. I also show that a computational account of its nonexistence is viable, as licit strings derived from Sour Grapes have exceptionally complex phonotactics.

1 What is Sour Grapes?

Sour Grapes is an unattested process predicted in parallel Optimality Theory (Prince & Smolensky, 1993/2004) when an AGREE constraint is used to motivate iterative spreading (Wilson, 2003, 2006; McCarthy, 2003, 2010). It is characterized by a feature that spreads completely through some domain, or, if complete spreading would be blocked, does not spread at all.

The tableaux below illustrate Sour Grapes, using progressive nasal spreading à la McCarthy (2010). In addition to the phonological strings, a schematic representing different segment types is given: M is a trigger, A is a target, and S is a blocker. This representation greatly improves the clarity of and simplifies the computational analysis.

Tableaux (1-2) illustrate nasal spreading in strings without blockers. AGREE penalizes adjacent segments with different values of [nasal], and IDENT penalizes changes in [nasal]. Spreading the feature [nasal] word-internally (1) and word-finally (2) satisfies AGREE. The symbol \( \triangleright \) explicitly marks the right-edge of the string.

(1) Word-internal spreading: \( /M\text{a}^nM/ \rightarrow [M\text{M}^nM] \)

<table>
<thead>
<tr>
<th></th>
<th>AGREE</th>
<th>IDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. mawam</td>
<td>W 2</td>
<td>L</td>
</tr>
<tr>
<td>b. máwam</td>
<td>W 2</td>
<td>L 1</td>
</tr>
<tr>
<td>c. máwam</td>
<td>W 2</td>
<td>L 2</td>
</tr>
<tr>
<td>\rightarrow d. māwām</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

(2) Word-final spreading: \( /M\text{a}^n/ \rightarrow [M\text{M}^nK] \)

<table>
<thead>
<tr>
<th></th>
<th>AGREE</th>
<th>IDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. mawa</td>
<td>W 1</td>
<td>L</td>
</tr>
<tr>
<td>b. máwa</td>
<td>W 1</td>
<td>L 1</td>
</tr>
<tr>
<td>c. māwa</td>
<td>W 1</td>
<td>L 2</td>
</tr>
<tr>
<td>\rightarrow d. māwā</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

The constraint *NASALFRICATIVE penalizes nasal fricatives. Ranked above AGREE, the constraint that motivates spreading, *NASALFRICATIVE blocks spreading [nasal] onto fricatives (3e-f). This leaves candidates (a-d) which tie on AGREE – each violating it once. The choice of which candidate is optimal falls to IDENT, which prefers the fully faithful candidate (a). Thus, the realization of a target depends on the presence of a trigger to its left, which may be arbitrarily distant, and the absence of a blocker to its right, which may also be arbitrarily distant. Crucially, blocking by fricatives is limited to the first nasal to their left; long strings may have multiple domains of spreading, e.g., \( /\text{MAM ASM AM SAM A/} \rightarrow [\text{MM M M M M M}] \). As (4) illustrates, spreading between two nasals improves on AGREE, and the final fricative only blocks spreading from the second /m/.

(3) Blocking: \( /M\text{a}^nS\text{a}^n/ \rightarrow [M\text{M}^nS\text{a}^n] \)

<table>
<thead>
<tr>
<th></th>
<th>*NASFRC</th>
<th>AGREE</th>
<th>IDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>\rightarrow a. mawasa</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b. máwasa</td>
<td>1</td>
<td>W 1</td>
<td></td>
</tr>
<tr>
<td>c. māwasa</td>
<td>1</td>
<td>W 2</td>
<td></td>
</tr>
<tr>
<td>d. māwasa</td>
<td>1</td>
<td>W 3</td>
<td></td>
</tr>
<tr>
<td>e. māwās</td>
<td>W 1</td>
<td>1</td>
<td>W 4</td>
</tr>
<tr>
<td>f. māwās</td>
<td>W 1</td>
<td>L</td>
<td>W 5</td>
</tr>
</tbody>
</table>

(4) Spreading and blocking: \( /M\text{a}^nM\text{a}^nS/ \rightarrow [M\text{M}^nM\text{a}^nS] \)

<table>
<thead>
<tr>
<th></th>
<th>*NASFRC</th>
<th>AGREE</th>
<th>IDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>\rightarrow a. maamaas</td>
<td>W 3</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>b. mámaas</td>
<td>W 3</td>
<td>L 1</td>
<td></td>
</tr>
<tr>
<td>c. māmāas</td>
<td>W 3</td>
<td>L 2</td>
<td></td>
</tr>
<tr>
<td>d. māmāas</td>
<td>1</td>
<td>W 3</td>
<td></td>
</tr>
<tr>
<td>e. māmās</td>
<td>1</td>
<td>W 4</td>
<td></td>
</tr>
<tr>
<td>f. māmās</td>
<td>W 1</td>
<td>L</td>
<td>W 5</td>
</tr>
</tbody>
</table>

---

\(^1\) The term Sour Grapes was originally used by Padgett (1997) to describe feature geometric assimilation where either all features of a given category assimilate or none do.

\(^2\) Cf. blocking at a distance in bounded harmony (Walker, 2010; Kimper, 2012).

\(^3\) Cf. the characterization by O’Hara & Smith (2018a); Smith & O’Hara (2019).
2 Sour Grapes is a weakly deterministic function

Sour Grapes is a function, i.e., a mapping from inputs to outputs. Heinz & Lai (2013) prove that it is not subsequential, meaning that it cannot be modeled by a deterministic finite-state transducer (FST). They conjecture that Sour Grapes is not weakly deterministic, meaning it cannot be modeled by composing two subsequential FSTs that apply in opposite directions, and that this excess complexity explains its non-existence. This explanation falls short for two reasons. First, there are attested processes whose application depends on non-local information on both sides of a target (Jardine, 2016a; McCollum et al., 2018). Second, as I show below, Sour Grapes is, in fact, weakly deterministic.

Feeding any input over \{M, A, S\} first through the left subsequential FST in (5) and then the right subsequential FST in (6) models its Sour Grapes output. I wrote a Python script to verify the analysis over all strings \{M, A, S\} \leq 19 (1,743,392,199 total). There are two contexts for spreading [nasal]: \text{M}^+ \text{M} and \text{M}^+ \text{X}. The first FST (5) spreads in bounded contexts with one or two targets: \text{MAM} \rightarrow \text{MMM}, \text{MA}\text{X} \rightarrow \text{MM}\text{X}, \text{MAAM} \rightarrow \text{MMMM}, \text{MA}\text{AX} \rightarrow \text{MM}\text{AX}. In unbounded contexts with three or more targets, it leaves a message in the form of \text{MAM} to be interpreted by the second machine: \text{MA}^+\text{AAAM} \rightarrow \text{MA}^+\text{MAAM}, \text{MA}^+\text{AAA} \rightarrow \text{MA}^+\text{MAAX}. Removing all \text{MAM} strings that were present in the input guarantees the message is unambiguous. The second FST (6) reads this message, and spreads [nasal] leftwards until it reaches another \text{M}: \text{MA}^+\text{MM} \rightarrow \text{M}^+\text{MMMM}. This markup strategy is similar to those used in weakly deterministic analyses of similarly complex processes (O’Hara & Smith, 2018a,b; McCollum et al., 2018; Smith & O’Hara, 2019).

An example derivation is given in (7-8) mapping the input /MAMASMAAA/ onto the output [MMMMASMMMM]. The input contains a local context \text{MAM}, a blocking context \text{MAS}, and an unbounded context \text{MAAAA}.

3 Sour Grapes defines a non-counting language

Sour Grapes is not exceptionally complex as a function. However, as I show below, the set of phonotactically well-formed strings it defines is exceptionally complex. We can define the language associated with Sour Grapes as those strings that map onto themselves. For example, the string \text{MAAMASMAAA} maps onto \text{MMMMASAMMM} and is in the language; the string \text{MMMMASAMMM} maps onto itself and is in the language. The language corresponds to the set of phonotactically licit strings in a phonology where Sour Grapes is the only process.

All strings that consist only of A, M, or S belong to the language. As do strings that consist only of M and S. Phonotactically illicit strings contain both A and M, e.g. \text{MAA} is not in the language as it maps onto \text{MMM}. The

4For space, I do not give more detailed definitions – see Heinz & Lai (2013); Jardine (2016a) for clear discussion of these classes.

5Available at https://github.com/lmaoaml/Sour-Grapes

6This strategy does not work for any regular relation; Lamont et al. (2019) prove that the weakly deterministic class is subregular.
are licit, but set of substrings up to length \( k \) MA closed under suffix substitution (McNaughton & Papert, 1971):  

\[
(11) \begin{align*}
& \text{a.} \quad /b´a-ka-mu-londolol-a/ \\
& \text{b.} \quad /b´a-ka-londolol-a=k´o/ \\
\end{align*}
\]

---

Bickmore, 2015). The phonotactic language bans spans of high tones across fewer than three moras, and, when a Grapes. In words without final high-toned moras, the rightmost high tone spreads to the right edge (11a);  

\[
(12) \begin{align*}
& \text{(10) } \\
& \text{Jardine (2016a) notes that high tone spreading in Copperbelt Bemba bears a strong resemblance to Sour Grapes. In words without final high-toned moras, the rightmost high tone spreads to the right edge (11a); in words with final high-toned moras, a ternary spread occurs instead (11b) (Bickmore \\& Kula, 2013; Kula \\& Bickmore, 2015). The phonotactic language bans spans of high tones across fewer than three moras, and, when a high tone is present in the string, final low tones. A logical definition of this language is given in (12). This uses strings, negation, and disjunction, and so belongs to the Locally Testable (LT) class (Rogers \\& Pullum, 2011).}
\]

\[
(11) \begin{align*}
& \text{a.} \quad /b´a-ka-mu-londolol-a/ \rightarrow [b´ak´am´ul´o´ond´ol´ol´a] \quad \text{‘they will introduce him/her’} \\
& \text{b.} \quad /b´a-ka-londolol-a=k´o/ \rightarrow [b´akal´ondolol´alk´o] \quad \text{‘they will introduce’} \\
\end{align*}
\]

---

(12) \((\neg H) \lor (\neg LHL \land \neg LHHL \land \neg L\kappa)\)  

Dominant/antidominant \([ATR]\) harmony in Turkana superficially resembles Sour Grapes. \([+ATR]\) is dominant, and spreads bidirectionally (13a); the ‘VOI’ suffix (bolded) is exceptional and triggers \([-ATR]\) harmony (13b) (Dimmendaal, 1983). \([+ATR]\) has the appearance of spreading all the way unless blocked, but this is better analyzed as two different processes (Baković, 2000, 2005). The language is Tier-Based Strictly Local (TSL) (Heinz et al., 2011); it can be defined by banning certain substrings over a tier that contains vowels and a symbol marking the left edge of the ‘VOI’ suffix # (14). The first three conjuncts ban tier-adjacent disharmonic vowels in general and across the ‘VOI’ suffix boundary, and the fourth bans the ‘VOI’ suffix from surfacing with a \([+ATR]\) vowel.  

\[
(13) \begin{align*}
& \text{a.} \quad /e-ib-us-a-km/ \rightarrow [eibusok´in] \quad \text{‘it has fallen down’} \\
& \text{b.} \quad /e-ib-us-a-km-a/ \rightarrow [eibusok´ina] \quad \text{‘it has throw itself down’} \\
\end{align*}
\]

\[
(14) \quad \neg [-ATR][+ATR] \land \neg [+ATR][-ATR] \land \neg [+ATR][\neg AT]\ell \land \neg \#[+ATR] \\
\]

---

Not only does Sour Grapes define a more complex language in terms of stringsets, it cannot be defined as an Autosegmental Strictly Local language, which bans a finite set of connected \( k \)-subgraphs (Jardine, 2016b, 2017, 2018). Because the only feature shared between targets and blockers is \([-\text{nasal}]\), there is no way to distinguish between spreading contexts (15) and non-spreading contexts (16).\(^7\) Both contain arbitrarily many targets, and the latter contains a blocker. Because the contexts are unbounded, there is not a way to pick these out with a finite graph. Strings in Copperbelt Bemba, on the other hand, are easily expressible in these terms (17).\(^7\)  

\(^7\)Blum (2018) shows this is not the case when targets and blockers have another feature in common.
Acknowledgments

This work has greatly benefited from discussions with Eileen Blum, Adam Jardine, Charlie O’Hara, Joe Pater, Brandon Prickett, Caitlin Smith, participants in the UMass Sound Workshop, and anonymous reviewers for LSA 2019. All remaining errors are of course my own.

References


