SUPPLEMENTARY MATERIALS: An agent-based approach to the gravity model of linguistic diffusion

Xiaoxing Yu[†]

Project advisor: Christina J. Edholm[‡]

Appendix A. Sociolinguistic and sociophonetic theory.

A.1. Vowels and chain shifts. Phoneticians characterize vowels by their formant frequencies—the resonant frequencies which carry the most acoustic energy. Most of the important acoustic information in a vowel is conveyed in its two lowest formants (F1 and F2), and it is conventional to imagine vowels as points in a two-dimensional feature space defined by their F1 and F2 values. It is also conventional to refer to an increase in a vowel's F1 as "lowering" and an decrease in a vowel's F1 as "raising". Similarly, an increase in F2 is called "fronting" while a decrease in F2 is called "backing". Although other acoustic cues are also involved in the production and perception of vowels, F1 and F2 are sufficient to categorize most of the commonest vowel sounds found in the languages of the world. Figure SM1 on the following page shows the conventional depiction of F1-F2 vowel space used in linguistics.

Along with other acoustic cues, formant frequencies organize vowel sounds into categories by similarity (see § A.3). A category of vowel sound which distinguishes words from another category is called a *phoneme*. For example, the phoneme /i/ distinguishes the word *heat* from the word *hit*, which has the phoneme /ɪ/. Because phonemes distinguish words, they are said to bear *functional load*. In order to be able to communicate effectively, it is hypothesized that speakers will attempt to maintain phonetic distinctions between phonemes [SM1]. In

[†]Pomona College, Claremont, CA (xyaa2021@mymail.pomona.edu)

[‡]Scripps College, Claremont, CA (cedholm@scrippscollege.edu)

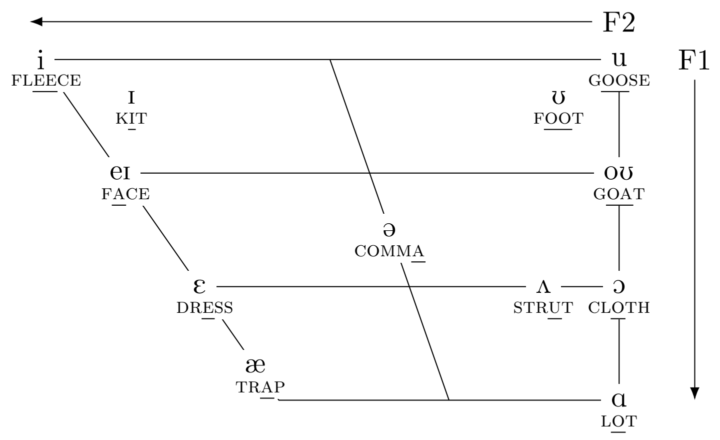

Figure SM1. *Conventional schematic depiction of English F1-F2 vowel space, heavily adapted from [SM7, ch. 3]. Each phonetic symbol follows International Phonetic Alphabet (IPA) conventions and is accompanied below by an English word that contains the vowel. The letter(s) which corresponds to the vowel shown is underlined. The arrow sat the side of the diagram show directions of increasing F1 and F2; note that the terminology of "raising", "lowering", "fronting", and "backing" derive from this conventional representation of vowel space.*

other words, it is hypothesized that speakers *speak to be understood*, and will generally avoid removing information from the speech signal by maintaining distinctive productions of the two contrasting phonemes.[1]

This "repelling force" that functional load exerts can, in combination with a single vowel's shift, lead to global changes in a speaker or language community's vowel system. As a single vowel shifts its position in F1-F2 space, it may "push" another vowel away, which may in turn "push" a third vowel, and so on. In this way a *push-chain* of vowel shifts occurs. It is also possible for a *pull-chain* shift to occur, wherein, under the functionalist hypothesis, in the wake of a single vowel shift a second vowel expands into the gap in F1-F2 space left behind to maximize distinctiveness between productions [SM4]. Note that the functionalist-teleological view of chain shifts (the idea that chain shifts occur to achieve some end goal) explained above is not without criticism, and alternative perspectives exist (see [SM4]). It is quite possible that the apparent teleological or "objective-seeking" nature of chain shifts is also an emergent

[1]This hypothesis does not preclude the possibility that two phonemes merge (a contrast is lost); phonemic merger is by no means a rare or unattested process [SM9, ch. 4].

phenomenon of speaker interactions, which would be interesting to investigate in a separate agent-based model. Nevertheless, for the purposes of the present paper we will defer to the functionalist-teleological view for simplicity.

A.2. The Northern Cities Shift. The Northern Cities Shift (NCS) or Northern Cities Vowel Shift is a feature of vowel systems of speakers in cities of the U.S. Inland North (that is, the region around the Great Lakes) [SM7]. Map 14.4 of [SM7] shows the approximate geographic extent of the NCS. As mentioned previously, the NCS is an example of a chain shift, in which not just one, but a whole series of vowels shifts in vowel space. In particular, the NCS is a complex system of two coupled chains whose trigger is the general raising of the vowel /æ/ (as in *bat*) [SM7, ch. 11]. The vowel /ɑ/ (as in *bot*) then moves into the gap /æ/ leaves, after which /ɔ/ (as in *bought*)[2] moves into the gap /ɑ/ leaves. This is the red pull-chain shown schematically in Figure 11.1 of [SM7]. The second chain consists of the backing of the vowel /ɛ/ (as in *bet*) towards the vowel /ʌ/ (as in *but*), which in turn is pushed towards the former position of /ɔ/. Finally, the vowel /i/ (as in *beet*) backs into the gap left by the shifting of /ɛ/ [SM7, ch. 11]. This is the blue chain shown schematically in Figure 11.1 of [SM7].

As the previous description shows, the NCS is a fairly complicated process that consists of both pull- and push-chains. It should also be noted that not every feature of the NCS manifests in every speaker which has the shift, and not to the same extent. The dimensions along and the extents to which the NCS changes take place are modulated not only by geography and social demographics, but also by speaker identity and agency [SM2].[3] For the purposes of constructing a simple and generalizable model, we will make no more mention of these additional factors in the remainder of this paper, but it is important to keep in mind the complex realities that can and often do underlie language change.

[2]For many speakers of U.S. English *bought* rhymes with *bot* due to a phenomenon known as the *cot-caught* merger. This merger has not taken place for speakers with the NCS.

[3]This was demonstrated famously in Penelope Eckert's 1988 study on high school students in a suburb of Detroit [SM2], who manifested the NCS more or less depending on their alignment with one of two social groups, "jocks" and "burnouts".

A.3. Exemplar theory. Up to this point, we have been treating vowels as points in a two-dimensional acoustic space. Leaving aside issues of psychoacoustic processing, it is clear that this way of thinking about vowels is by itself insufficient. No two spoken instances of a given vowel will be exactly the same as an acoustic signal. Despite this, language users seem to organize wide classes of acoustic signals into psychological categories [SM5]—even if two different people say the word *cat*, the vast majority of English speakers will be able to recognize the word as referring to a furry feline friend and furthermore identify that the vowel in the word is in some sense "the same" as the vowels in *hatch*, *strap*, and *pal*. Hence linguistics must have some way of explaining how listeners perform categorization.

One approach consistent with the general theme of emergence is that of *exemplar theory*, which posits that speakers store small snippets of episodic memory (*exemplars*) connected in a network [SM3]. Each exemplar has a different activation, and the patterns of activation patterns between exemplars bring out emergent categories. Taken to the logical extreme, eliminativist connectionist theories posit that there is no psychological reality to linguistic categories and that only the activation networks are psychologically real. However, empirical evidence challenges this view and many modern exemplar theories are hybrid theories that make use of both connections and categories [SM3].

For the present model, we make the simplifying assumption that exemplars contain one vowel each, leaving aside the question of how vowels are isolated from the speech signal in the first place. This assumption allows each exemplar to be modeled as a point in F1-F2 space and a vowel category to be modeled as a fuzzy "cloud" of points in acoustic space (Figure SM2). We also follow [SM8] in assuming that the network of connections can be modeled by summary statistics of the "cloud"—properties such as its mean and covariance matrix. A separate agent-based model of exemplar connections may be interesting to pursue in its own right. Finally, we assume that speakers have a limited memory size—that they can only hold a finite number of exemplars at a time. We also assume that all exemplars are

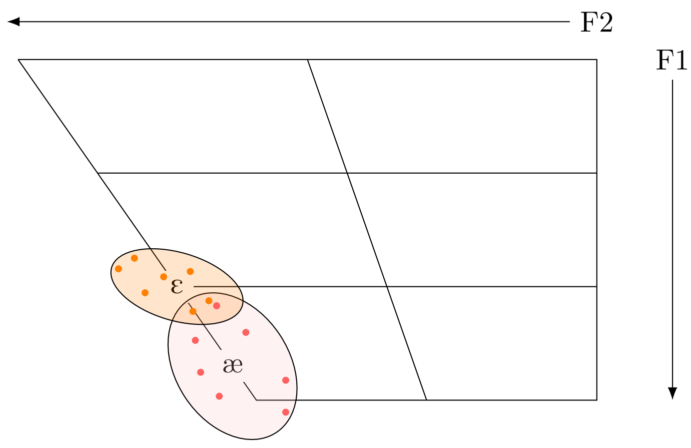

Figure SM2. *Schematic representation of exemplar-categories as interpreted for the present model. Vowels are represented as "clouds" (ellipses) which emerge from a set of exemplars (points in F2-F1 space). The boundaries of the ellipses may be interpreted as an arbitrarily chosen cutoff on the probability that a given token is an exemplar of the corresponding vowel. These "clouds" may overlap, as is depicted for the categories /æ/ and /ɛ/ given in this example.*

equally influential in processing speech—we do not assign them weights for the purposes of calculating the aforementioned summary statistics.

Having established the basic tenets of within-speaker and within-community vowel systems, we now turn to theoretical considerations about between-speaker between-community interactions.

A.4. Gravity model. The gravity model of linguistic diffusion proposes that linguistic features (such as sounds, words, and the structures they fit in) spreads from one language community to another at a rate proportional to the interactions between individuals between the two communities [SM10]. The name "gravity model" arises from a physics metaphor: just as the gravitational attraction between two bodies increases with increasing mass of either body and decreasing distance between their centers of gravity, the gravity model predicts that the average number of linguistic interactions (ex., conversations) between two communities increases with increasing population densities of either language community and decreasing distances between them. Furthermore, it predicts that increased numbers of linguistic interactions leads to an increased rate of spread for a given change. The surprising prediction that the gravity model makes, then, is that a feature can spread from a high-density settlement to a medium-density settlement skipping past a geographically intervening sparsely settled region. The implicit assumptions behind positing that population density and distance are the only factors mediating interaction rates include the homogeneity of social networks (every individual in one population has a roughly equal probability of interacting with every individual in another) and no intervening geographic barriers (such as a mountain or a river).

Appendix B. ODD (Overview, Design concepts, Details).

Where appropriate, we will note similarities and differences with [SM8]'s original design (henceforth the *base model*).

We illustrate the model with flowcharts throughout this section. In each flowchart, the submodel begins at the red node START, then runs through the given steps until it reaches the red node STOP. Rounded boxes represent start/stop nodes, rhombic boxes decision points, double boxes subsubmodels, and simple boxes single actions. Where a decision box contains a yes-no question, the edges proceeding from the box are labeled with YES and NO, respectively; where a decision box contains a state variable or "breed", the edges proceeding from the box are labeled with the values of that state variable or the breed of the turtle.

B.1. Purpose and patterns. This model is designed to explore the theoretical underpinnings of the gravity model of linguistic feature diffusion. In particular, it seeks to investigate whether the rate of interaction between speaker communities alone can explain the surprising phenomenon of a feature spreading from a high-density population center to a medium-density population center while skipping over an intervening sparsely populated region and to check whether the predictions that a feature diffuses faster with increased population density and slower with increased distance are borne out purely from speaker-to-speaker interactions. This investigation promises to lend insight into the mechanics of language diffusion patterns and provide support for or challenge existing perspectives on this type of change.

The key outcomes of the model suggest that patterns of travel between settlements is also an important determinant in which communities a shift spreads to first and that additional parameters such as speakers' acceptance of novel acoustic signals as vowel tokens are critical in determining the patterns linguistic diffusion follows.

The present model differs in both purpose and key outcomes from the base model. The base model sought to model the empirical dialect phenomena of (intergenerational) transmission, (intercommunity) diffusion, and incrementation (of sound change) as laid out in [SM6]. The present model seeks to verify the theoretical model that is the gravity model, and does not concern itself with intergenerational transmission. While the base model finds that an interaction and exemplar model are together sufficient to reproduce the three phenomena, the key outcome of the present model is that other parameters such as acceptance threshold are important in determining whether the model outcome qualitatively fits that of the gravity model.

B.2. Entities, state variables, and scales. The entities in the model consist of turtles, patches, and NetLogo's Observer. The turtles come in three "breeds": `cityfolk`, `townsfolk`, and `villagefolk` (singular: `cityperson`, `townsperson`, and `villageperson`). These breeds correspond to the three different settlements in the model, the `city`, the `town`, and the `village` (see § B.4.6). The patches lie in a 450 $\times$ 450 grid with no wrapping at the edges. The Observer is a static entity through which the user controls the simulation.

The choice of entities follows from the study's purpose to study speakers, language communities, and their interactions. Speakers and language communities are directly represented by entities, the former by turtles and the latter by the `city`, `town`, and `village` collectives. The patches represent space, which mediates between the communities by determining the distances between them. Recall that verifying whether increased population density and decreased distance (both spatially-formulated variables) leads to higher rates of shift is one of the aims of the present model. The choice to implement three collectives reflects the model's purpose in reproducing the gravity model's prediction that a change may "skip" from one

Variable	Type	Units	Range	Dynamic?	Meaning
`pxcor`	integer	—	$\{0, 1, 2, \ldots, 449\}$	no	east-west location of patch
`pycor`	integer	—	$\{0, 1, 2, \ldots, 449\}$	no	north-south location of patch

Table SM1

Summary of patch state variables. Here and elsewhere in this paper, reference to cardinal directions are used as a convenient way to describe the model world. Cardinal directions are not to be interpreted as real directions in the real world.

community to another over a geographically intervening community—three is the minimum number of communities at which this phenomenon could potentially be observed.

The choice to implement three abstract collectives is one of the most important ways in which the present model diverges from the base model, which incorporates only two collectives, and which explicitly models them after the cities of Chicago and St. Louis.

Patches have only two state variables each: an x- and a y-coordinate (`pxcor` and `pycor`, respectively). Each variable takes integer values between 0 and 449, inclusive, to form the aforementioned 450×450 non-wrapping grid of the world. This size was chosen to strike a balance between a too-small world in which turtles would be highly likely to wander near the edges of the world and induce unwanted edge effects, and a too-large world in which turtles would not be able to reach communities other than their own in a reasonably short time or talk to enough other speakers to trigger a shift in a reasonably short time. Neither the size of the world nor its wrapping are explicitly given in [SM8]. Table SM1 summarizes the patches' state variables.

There are seven state variables whose values may differ for each turtle: `xcor` and `ycor`, `mode`, `time-to-return`, `home-patch`, `interlocutor`, and `exemplars`. `xcor` and `ycor` are integers that taken values from 0 to 449, inclusive, and specify a turtle's location in space. `mode` is a string that determines how a turtle moves through space and can take the values `"traveling"`, `"wandering"`, and `"returning"`. `time-to-return` determines how many time periods a turtle will pass before switching to `returning` mode, and can take on integer values from 1 to 4, inclusive. A turtle also remembers its `home-patch`, the patch on which it spawned. When speaking, a turtle also sets its `interlocutor`, a single other turtle it speaks to.

By far the most complicated state variable turtles have is `exemplars`, which consists of a list of three lists of floats. Each float represents a vowel exemplar; each list of floats, a vowel category. The justification for using single floats to represent vowel exemplars is as follows: recall that the trajectory of vowels in the NCS is roughly circular; hence the two-dimensional F1-F2 vowel space may be simplified to a single parameter to be interpreted as the "advancement" of a given vowel along the shift's track. Assuming the shift does not advance as far as to rotate every vowel in vowel space, the topology of this track is equivalent to that of a line segment, and may be represented simply as a float. The processes each of the turtles' state variables are used in and their meaning within the model are explained in detail in § B.3. Table SM2 summarizes the turtles' state variables.

The base model's state variables are not explicitly listed in [SM8], but many of the same mechanisms the present model's state variables are used for find a parallel in the base model. At the very least, the base model includes for each turtle a state variable which is a list of numbers functioning as a category, (that is, a cloud of exemplars).

The model has three abstract scales: time, space, and population. Each tick in the model corresponds to the amount of time in which a speaker can move a distance corresponding to one patch. Depending on the application, "move" may mean walking, driving, or some other method of transportation. In terms of space, one patch corresponds to the distance from which two users can have a conversation. For the purposes of the model, these scales provide an uncomplicated correspondence between space and time. Finally, if the city is taken to be of a similar size to Chicago and the town a similar size to St. Louis (following the scale given in [SM8]), the population scale is changed from the base model's scale of 1500 humans to 1 turtle, so that about 5000 humans correspond to 1 turtle. This population scale was chosen to give enough detail about interactions without undue computational burden.

Variable	Type	Units	Range	Dynamic?	Meaning
`xcor`	integer	—	$\{0, 1, \ldots, 449\}$	yes	east-west location of turtle
`ycor`	integer	—	$\{0, 1, \ldots, 449\}$	yes	north-south location of turtle
`mode`	string	—	$\left\{\begin{array}{l}\texttt{traveling}\\ \texttt{wandering}\\ \texttt{returning}\end{array}\right\}$	yes	how a turtle moves through space
`time-to-return`	integer	time periods	$\{1, 2, 3, 4\}$	yes	time until a turtle switches to `returning` mode
`home-patch`	patch	—	—	no	the patch a given turtle spawned on
`interlocutor`	turtle	—	—	yes	who the turtle is currently speaking to
`exemplars`	list of float lists	—	$\mathbb{R}$	yes	vowel exemplars organized by category
`destination`	patch	—	—	yes	the patch a villageperson is headed to

Table SM2

Summary of turtle state variables. Here and elsewhere in this paper, reference to cardinal directions are used as a convenient way to describe the model world. Cardinal directions are not to be interpreted as real directions in the real world.

B.3. Process overview and scheduling. Each tick, the following actions occur in the following order. For more detail on the submodels described here, see the sections referenced below.

Certain submodels only occur each *time period*. In the present model, every 500 ticks constitutes a *time period*. Hence the first thing the model does is to check whether the current tick count divides 500 (that is, whether the model has entered a new time period). If so, each turtle executes the special submodel `check-return` (§ B.7.1), which checks whether it is time for the turtle to return to its home patch and updates the turtle's `time-to-return` state variable. Then, the Observer executes the submodel `make-travelers` (§ B.7.2), which selects 5% of each breed of turtle to travel, updating their `mode`. If the turtle is a villageperson, `make-travelers` also updates their `destination`.

Next, whether or not the model has entered a new time period, every turtle with their `mode` state variable set to `traveling` execute the submodel `travel` (§ B.7.3), which sends the turtle to a different community from its own, updating its `xcor` and `ycor`. Every turtle with their `mode` state variable set to `wandering` and also execute the submodels `move` (§ B.7.4), which lets the turtle roam around the world by updating its `xcor` and `ycor`, and `speak` (§ B.7.5), which lets turtles talk to each other and updates their `interlocutor` and `exemplars` (see also § A.3). Every turtle with their `mode` state variable set to `returning` execute the submodel `go-home` (§ B.7.6), which sends a turtle to its home patch by updating its `xcor` and `ycor`. Finally, each turtle executes `update-vowels` (§ B.7.7), which cuts each of the three lists in their `exemplars` down to the maximum memory size and lets vowel categories (lists in `exemplar`) "push" one another if they are too close. After each of these submodels has been executed, the Observer increments the model's `ticks` count and starts the process from the beginning once again. Note that each time a set of turtles executes the same submodel, the order of execution is randomly chosen. Figure SM3 on the next page presents a flowchart that summarizes the overall schedule.

The precise ordering of processes is crucial in only two respects: the precedence of `make-travelers` before `travel` and the precedence of `speak` before `update-vowels`. The first of these ensures that newly created travelers begin traveling to their destination in the

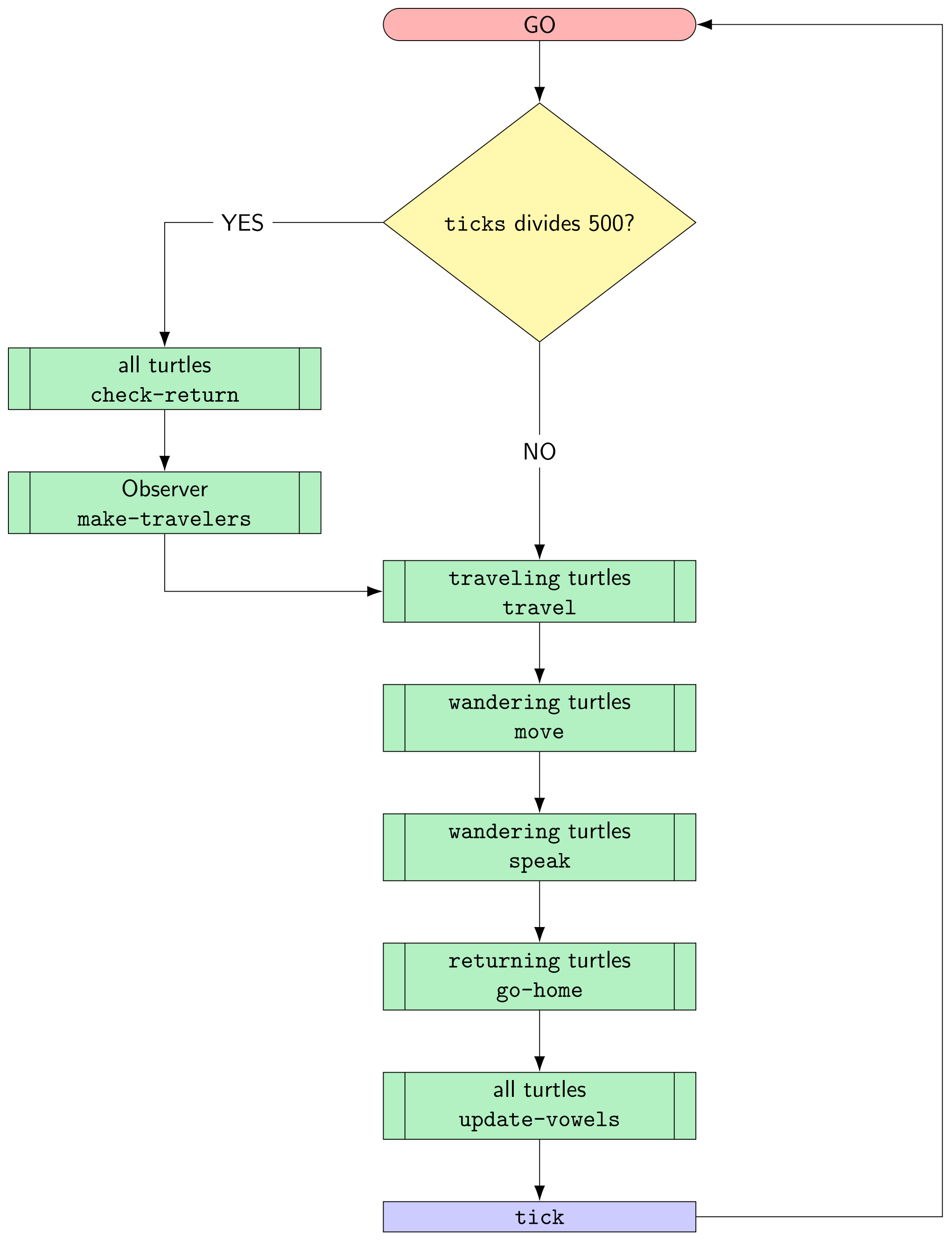

Figure SM3. *Flowchart showing the overall process of the model. Each tick, the model begins at the red node GO, then runs through the given steps until it reaches the blue action `tick`, at which it increments the tick count and begins the process anew. Rounded boxes represent start/stop nodes, rhombic boxes represent decision points, double boxes represent submodels, and simple boxes represent single actions.*

same tick, so that they do not spend a tick doing nothing while `wandering` and `returning` turtles are moving. Hence this order guarantees that all turtles execute their submodels each tick. The second of these ensures that by the end of each tick all turtles have appropriate memory sizes. If the model did not proceed in this order, the turtles' vowel lists would continually expand. See § B.7.7 for a more thorough investigation of this point.

The present model's schedule critically differs from the base model in that it dispenses with the base model's submodels associated with generational transmission (ex., birth, death, coupling, maturing). As generational transmission is not a focus of the current project, we have chosen to exclude these factors to focus on the gravity model as stated in [SM10], which makes no explicit mention of generational transmission. Moreover, the particulars of the present schedule also differ slightly from that of the base model in the following ways: the populations of the communities has been reduced by a factor of 0.3 and the number of ticks per time period has been reduced by a factor of 0.5 compared to the base model to alleviate unnecessary computational burdens on the model. The specific implementation of each of the base model's submodels and their order of execution is not specified in [SM8], so we have implemented and arranged them in a convenient manner according to the reasoning in the paragraph above. However, the analysis in the main article shows that these modifications continue to reproduce the essential qualitative results of the base model.

B.4. Design concepts.

B.4.1. Basic principles and emergence. The gravity model, as presented in [SM10], is the basic principle and guiding theoretical framework of the present model. In short, the gravity model predicts that the rate of diffusion of a linguistic feature between two communities increases with increasing population density of each community and decreasing distance between the two communities (but for more detail see § A.4). A surprising consequence of this prediction is that a feature can spread from one community to another more quickly than a third community physically between the two, if the third community is sparsely populated.

The central goal of the present model is to analyze whether the gravity model, a theory about the vowel systems of language communities, is an *emergent* phenomenon that results from theories about the vowel systems of individual language users—namely, exemplar theory (§ A.3) and the functionalist-teleological view of chain shifts (§ A.1). The results in the main article show that the gravity model does emerge, under certain conditions mediated by vowel production and perception and patterns of interaction.

The present model is able to evaluate the emergence of the gravity model because it is not explicitly represented in the model's structure. In contrast, the exemplar model and the functionalist-teleological view of chain shifts are directly implemented as the `speak` and `update-vowels` submodels (see § B.7.5, B.7.7) and are hence not taken to be emergent for the present purposes.

Again, the present model differs in these guiding principles and the expected emergent phenomena from the base model. The base model aims to validate the three empirically observed phenomena of transmission, diffusion, and incrementation, and finds that they emerge from the same submodels as in the present model, together with an additional submodel for birth, death, and maturation.

B.4.2. Adaptation and objectives. A critical feature of the present model is that turtles *adapt* their `exemplars` (and hence their vowel categories) on the basis of recent experience (that is, conversations with others) and cumulative knowledge (the general closeness of their vowel categories). In particular, a turtle speaking with another accepts the tokens its interlocutor produces as instances of one of its own vowel categories if the token's value is within a certain `tolerance` parameter of the range of its own category. Moreover, if two of its vowel categories are less than the parameter `stable-distance` apart, each turtle will add a new exemplar to the higher category. These processes are described in further detail in § B.7.5 and B.7.7.

These adaptations can also be understood in terms of objectives. By accepting new vowel tokens as exemplars agents indirectly maximize their adherence to community speech norms, or intelligibility between their own speech and others'. On the other hand, by pushing their vowels apart from one another, agents indirectly maximize the distinctiveness between their vowel categories and hence the the amount of information a given vowel signal carries.

Although these adaptations are directly represented in the model through the `speak` and `update-vowels` submodels (§ B.7.5, B.7.7), the objectives are not. Furthermore, although the turtles change their `exemplars` in response to other turtles' vowel tokens, the process of change is neither a decision nor a choice between multiple possible alternatives.

The same interpretation of adaptation and objectives can be applied to the base model.

B.4.3. Learning and prediction. Neither the present nor the base model incorporate learning or prediction.

B.4.4. Sensing and interaction. The turtles in the present model do not sense state variables of other turtles directly. However, they do learn information about other turtles' state variables indirectly through direct interaction (to be interpreted as a conversation).

Each tick, every `wandering` turtle selects another turtle one patch away from them (if such a turtle exists) to be their `interlocutor`, and the two turtles have a conversation. The decision to only allow `wandering` turtles to select conversational partners follows from the decision to preclude two turtles both traveling communities from speaking to each other. Therefore all conversations take place in, but not between the communities. This reflects the assumption that people are more likely to speak to each other while within a community than on the road between communities.

The base model differs from the present model mostly in that it does not explicitly distinguish `wandering` turtles from other kinds of turtles, so that any two turtles may have a conversation.

B.4.5. Stochasticity. The initialization of the model (§ B.5) and the `move` (B.7.4), `speak` (B.7.5), `check-return` (B.7.1), and `make-travelers` (B.7.2) submodels all incorporate stochasticity. For more details on how each incorporates stochasticity, see their respective sections.

Stochasticity is used in initialization and each of the aforementioned submodels to introduce variability without modeling its causes. For instance, in the `speak` submodel, turtles produce vowel tokens by drawing from a normal distribution with the same mean and standard deviation as their list of exemplars. This introduces variability in vowel production without modeling the complex pattern of connections and activations between exemplars. Variability in the model is desirable in general to bring conditions closer to that of the real world without having to model the complex and unpredictable mechanisms that underlie processes the present model is not interested in explaining. For details, see the respective sections referenced above.

It is generally not made clear how stochasticity is used in the base model in [SM8], though in a few places it is noted that it is to introduce variability and reduce unnaturalness. In particular, the base model points out that it asks turtles to randomly choose from 1 to 4 time periods before returning to their home community to avoid the unnatural situation of all turtles returning at the same time.

B.4.6. Collectives. The present model incorporates three explicit collectives: the `city`, the `town`, and the `village`. They are implemented in the model as the three breeds to which turtles belong and roughly correspond to the spatial distribution of turtles over the course of a single run of the model. For instance, the `cityfolk` begin distributed around the `city`'s center, and at any given tick in the simulation, it remains the case that all but a relatively small proportion of `cityfolk` are distributed around the center.

The present model is interested in the progression of vowel shifts throughout language communities. Explicitly representing the collectives allows the model to keep track of aggregate statistics for each community, which reveals observations about community-wide vowel norms and cuts the number of model outputs down to a manageable size.

The base model differs from the present model in that it does not include the `village` collective, focusing instead on only two communities. Since the present model aims to verify whether a change can spread from one community to another skipping an intervening sparse community, at least three communities are required. The base model also includes explicit "adult" and "children" collectives, but since the present model does not incorporate generational transmission, it dispenses with these collectives.

B.4.7. Observation. The main outputs of the model are vowel value means for each vowel category for each community (3 vowel categories $\times$ 3 communities $=$ 9 vowel value means). These outputs are calculated as follows. Let m represent the `memory-size` of the model, P_i be the population of the ith community, and $\mathbf{x}_{ij}^{(N)}$ be an $m \times 1$ vector containing all the vowel exemplars for vowel category N of the jth turtle in the ith community, where N stands for one of the three letters A, B, and C, i takes values $1, 2, 3$, and j takes integer values from 1 to P_i, inclusive. Furthermore, let $\bar{x}_{ij}^{(N)}$ be the arithmetic mean of the entries of $\mathbf{x}_{ij}^{(N)}$. Then the outputs are the 9 numbers

$$\mu_i^{(N)} \triangleq \frac{1}{P_i} \sum_{j=1}^{P_i} \bar{x}_{ij}^{(N)} \tag{B.1}$$

where again $i = 1, 2, 3$, N stands for A, B, C. In other words, the output values are means of mean vowel category values taken over each community. This is essentially a summary statistic. Although occasionally the $\bar{X}_i^{(N)}$ are taken at every tick or every ten ticks of the simulation, in many of the resulting analyses the $\bar{X}_i^{(N)}$ are measured only at the 10,000th tick of the simulation. The 10,000th tick is the first tick of the 21st time period and, from exploration of the model, allows enough time for interesting behavior to emerge and stabilize. No virtual scientist method is employed in collecting these observations.

In [SM8], the vowel category means of each turtle in the base model are often plotted as scatterplots, though numbers are also averaged over other collectives such as adults and children.

Parameter	Value	Parameter	Value
city-x	300	town-x	100
city-y	300	town-y	100
city-population	570	town-population	75
city-radius	60	town-radius	30

Table SM3
Summary of the initial settings of the city and town parameters

B.5. Initialization. The model begins by creating three breeds of turtles (`cityfolk`, `townsfolk`, `villagefolk`) in three collectives (`city`, `town`, `village`). Each collective has the four parameters `-x`, `-y`, `-population`, and `-radius`, which are named by prefixing the collective name (ex., `city-x`, `town-y`, `village-population`). The `-x` and `-y` parameters determine the *center patch* of each collective (the "`city` center", "`town` center", and "`village` center"); the `population` parameters determine how many turtles of the breed corresponding to the community are created.

The three communities are set to be close enough to interact with one another fairly frequently (that is, turtles can move far enough in a relatively small number of ticks to have many conversations) but faraway enough to remain distinct (the random motion of turtles doesn't cause two communities to speak to each other so much they effectively act as one community). Table SM3 summarizes the initial values the `city` and `town` parameters are set to.

The `village` has an additional parameter, `village-distance`, which determines how far southeast along the perpendicular bisector of the axis connecting the `city` and `town` centers the `village` center lies. Figure SM4 on the next page illustrates the configuration of the three communities in space; equation B.2 shows how `village-x` and `village-y` are calculated from `village-distance` and the `city` and `town` centers.

$$\begin{bmatrix} \texttt{village-x} \\ \texttt{village-y} \end{bmatrix} = \frac{1}{2}\left(\begin{bmatrix} \texttt{city-x} \\ \texttt{city-y} \end{bmatrix} + \begin{bmatrix} \texttt{town-x} \\ \texttt{town-y} \end{bmatrix}\right) + \frac{\texttt{village-distance}}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{B.2}$$

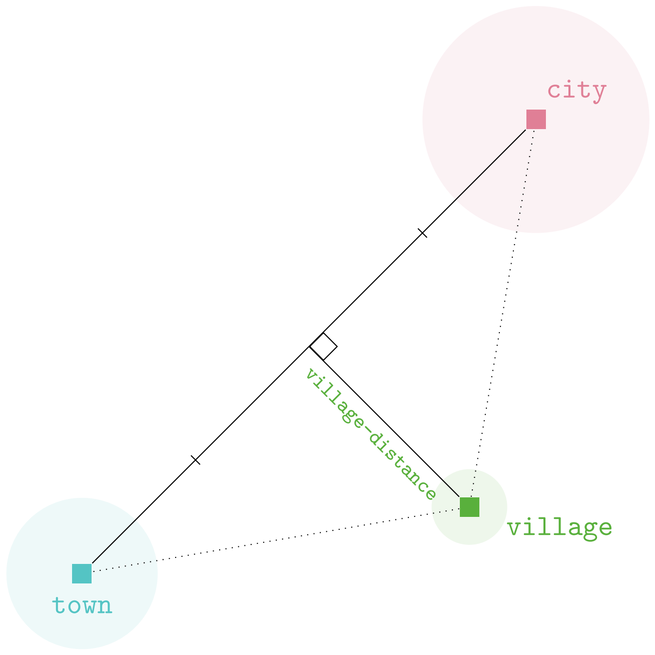

Figure SM4. *Configuration of the three communities in space. Each square labeled with the community name represents the center patch of that community, and the light circle around represents patches falling within the radius of that community. The `village` sits `village-distance` patches along the perpendicular bisector of the axis connecting the `city` and `town` centers to the southeast. The dotted segments connecting the village center to the `city` and `town` centers are the same length.*

In the main article, `village-population`, `village-radius`, and `village-distance` are the main parameters varied in experiments.

The turtles are created around the center of their community according to the community's population. Their initial x- and y-coordinates are set by normal distributions with mean the x- and y-coordinates of the community center and standard deviation half the community radius. This number was chosen so that around 90% of each turtle breed begin within one community-radius of their community center. The turtles are also set to the color corresponding their community for easy visual identification.

Finally, each turtle is given an initial list of exemplars the size filled up to their `memory-size` for each vowel category, namely A, B, and C. The mean values of these three categories are 5, 15, and 25 for the `townsfolk` and `villagefolk`; the `cityfolk` have the mean value of their vowel A set to 10. The choice of these values is so that the `townsfolk` and `villagefolk` have stable vowel systems, but the `cityfolk` have an unstable vowel system that will eventually lead to shift. In particular, these values have been set up so that a *chain shift* occurs: vowel A will push vowel B up in value, which should in turn push vowel C up in value. Moreover, the shift always spreads *from* the `city` *to* the `town` and `village`. The exemplar values also have a small amount of normally distributed noise added to them. This noise has mean zero and standard deviation equal to the parameter `initial-dev`. This noise introduces some variability in the initial state of the community and represents a more realistic situation in which every member of the community does not have exactly the same exemplars.

The base model appears to initialize its turtles in a similar way, but does not include the village collective and does not specify the distance between communities or how turtles are created within each community. Furthermore, [SM8] does not specify whether in the base model turtles begin with randomness in their list of exemplars or not.

B.6. Input data. Like the base model, the present model uses no input data.

B.7. Submodels. Each submodel in the present model makes use of a different set of parameters and updates a different set of state variables. Table SM4 provides an overview of the parameters and state variables involved with each submodel; Table SM5 summarizes the parameters and their values used in all analyses in the main article aside from the LHS/PRCC analysis. For how the submodels fit into the larger process, review § B.3 and Figure SM3. Note that not all submodels are explicitly represented as NetLogo procedures in the code.

B.7.1. `check-return`. At the start of each time period, each turtle performs `check-return` to see if it should begin returning to its home patch. If the turtle's state variable `time-to-return` is 0, the turtle runs the `go-home` submodel, then draws a random integer from the set $\{1, 2, ..., \texttt{max-stay}\}$ and sets it as its new `time-to-return`. Otherwise, it decrements its `time-to-return`. In other words, `time-to-return` ticks down if it is not 0, but initiates the `go-home` submodel and resets if it is. This submodel represents traveling speakers ending their stay at their destination and returning home. A flow diagram representing the submodel is

Submodel	Parameters	State variables updated
check-return	▶ max-stay	▶ time-to-return ▶ mode
make-travelers	▶ city-probability ▶ max-stay	▶ destination (villagefolk only) ▶ time-to-return ▶ mode
travel	—	▶ xcor ▶ ycor
move	—	▶ xcor ▶ ycor
speak	▶ tolerance	▶ exemplars ▶ interlocutor
go-home	—	▶ mode ▶ xcor ▶ ycor
update-vowels		

Table SM4
Overview of parameters and state variables appearing in each submodel.

shown in Figure SM5 on page SM23.

B.7.2. `make-travelers`**.** At the start of each time period, the Observer executes the submodel `make-travelers` to select turtles to travel to destinations outside their home communities. The Observer then *asks* different sets of turtles to perform different subprocesses. In particular, the observer asks an amount of `cityfolk` equal to `percent-travelers`·`city-population` to set their `mode` to `"traveling"` and to draw their `time-to-return` from a random integer between 1 and `max-stay`, inclusive. The observer then asks an amount of `townsfolk` equal to `percent-travelers` · `town-population` to do the same. In effect, this selects a fixed percentage of `cityfolk` and `townsfolk` to become travelers and assigns them a random integer number of time periods to stay at the community that is their destination.

The Observer finally asks an amount of `villagefolk` equal to `percent-travelers` · `village-population` to set their `mode` to `"traveling"` and perform a random draw of an in-

Name	Type	Default value	Meaning
`memory-size`	integer	50	maximum number of exemplars a turtle can store at the beginning of a tick
`tolerance`	float	1	maximum difference a vowel token's value can have from the range of a vowel category that can still be accepted as an instance of the same category
`drift-rate`	float	0.1	how quickly a vowel category too close to another drifts away from the latter
`max-stay`	integer	4	the maximum number of time periods a turtle can stay in a community that is not its own
`percent-travelers`	float	0.05	the proportion of each breed of turtle whose mode is changed to `traveling` at the start of every time period
`initial-dev`	float	0.1	the amount of variation in the turtles' initial exemplar lists; the standard deviation of the normal distribution the initial exemplars are drawn from

Table SM5

Summary of parameters appearing in the submodels. The default values are the values used throughout the analysis unless otherwise noted.

teger between 0 and 99. If this integer is strictly less than the parameter `city-probability`, which itself takes integer values between 0 and 100 inclusive, then the `villageperson` sets their `destination` to `"city"`. Otherwise, the `villageperson` sets their `destination` to `"town"`. Hence the parameter `city-probability` controls what proportion of `traveling` `villagefolk` are headed towards which destination each time period.

This submodel represents speakers deciding to travel to a different community each time period. Note that because the Observer looks at all turtles of each breed, it is possible for a turtle already in a different community to decide to "prolong its stay", changing its `mode` from `returning` to `traveling` again and randomly redrawing its `time-to-return` state variable. Figure SM6 on page SM24 illustrates this submodel as a flowchart.

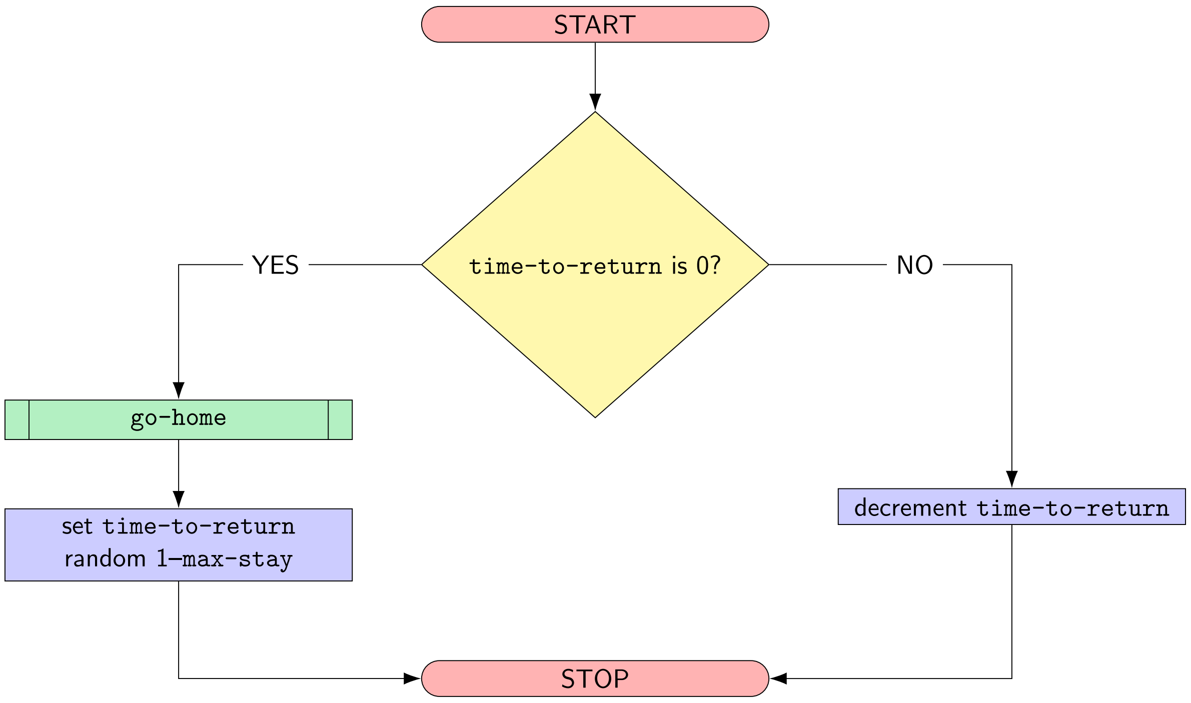

Figure SM5. *Flowchart showing the process* `check-return`*.*

B.7.3. `travel`**.** Each tick every turtle in the `traveling mode` executes the submodel `travel`. In this submodel, `traveling` `cityfolk` face the `town` center and move forward 1 patch while `traveling` `townfolk` face the `city` center and move forward 1 patch. As for `traveling` `villagefolk`, they face the `city` center or the `town` center if their `destination` state variable is `"city"` or `"town"`, respectively. This process can be interpreted as language users traveling a distance of a patch directly towards the center of the community that is their destination. Figure SM7 illustrates this submodel as a flowchart. Figure SM8 shows a capture of the turtles traveling in the model. Both figures can be found on page SM25.

B.7.4. `move`**.** Each tick every turtle in the `wandering mode` executes the submodel `move`. In this submodel, each `wandering` turtle draws a random integer from 0 to 8, inclusive. If the result is strictly less than 8 (an 8-in-9 chance), the turtle selects a patch neighboring the one

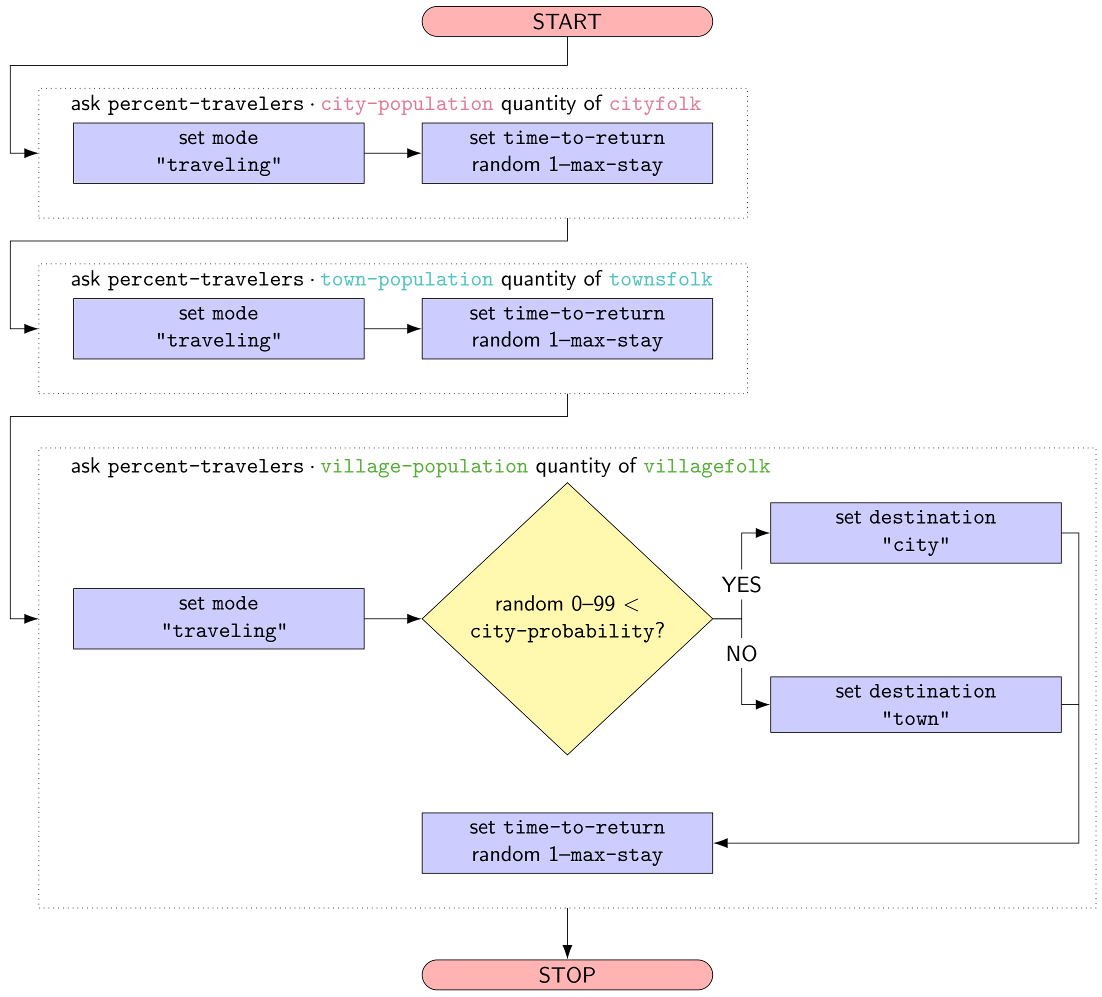
START
ask percent-travelers · city-population quantity of cityfolk
set mode "traveling"
set time-to-return random 1–max-stay
ask percent-travelers · town-population quantity of townsfolk
set mode "traveling"
set time-to-return random 1–max-stay
ask percent-travelers · village-population quantity of villagefolk
set mode "traveling"
random 0–99 < city-probability?
YES
NO
set destination "city"
set destination "town"
set time-to-return random 1–max-stay
STOP

Figure SM6. *Flowchart showing the process* `make-travelers`*. Subprocesses asked of sets of turtles are indicated in dotted boxes, and the sets of turtles asked to perform each subprocess are indicated in the top left corner of each box.*

it is on at random and moves there. A neighboring patch is defined as one of the eight patches that shares an edge or a vertex to the patch the turtle is currently on. Otherwise, the turtle does not do anything. In other words, the turtle has an equal chance of moving to any of its neighbors and not moving. This submodel produces random motion for the wandering turtles and allows nontraveling turtles to mix and speak with each other. Similarly, language users in the real world move about in their daily lives and converse with one another. Figure SM9 on the page after the next illustrates this submodel as a flowchart.

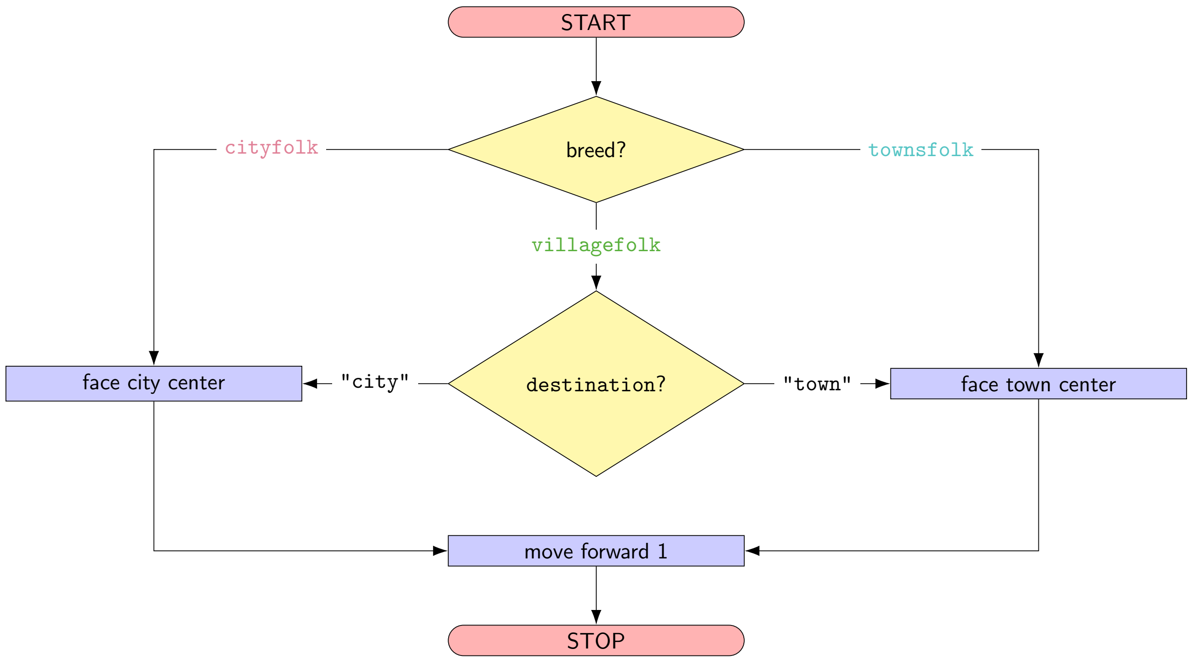

Figure SM7. *Flowchart showing the process* `travel`.

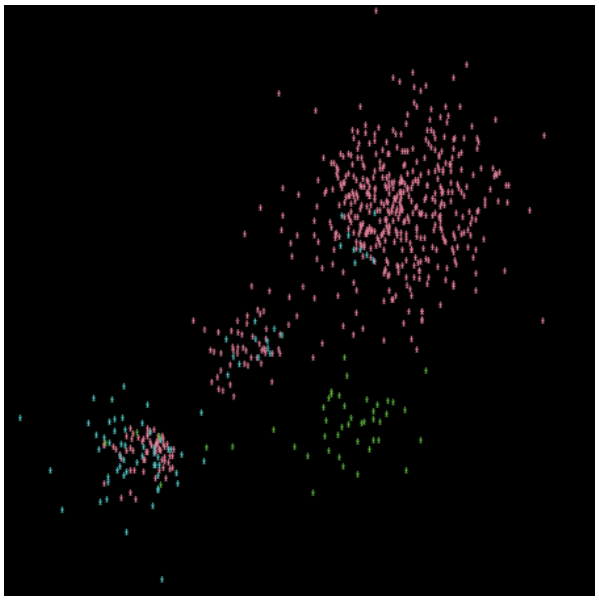

Figure SM8. *A snapshot of the model in action at tick 2613. The* `city` *is the large aggregate of pink turtles in the top right, the* `town` *is the aggregate of cyan turtles in the lower left, and the* `village` *is the smattering of green turtles in the lower right. The traveling agents appear between the* `city` *and the* `town`.

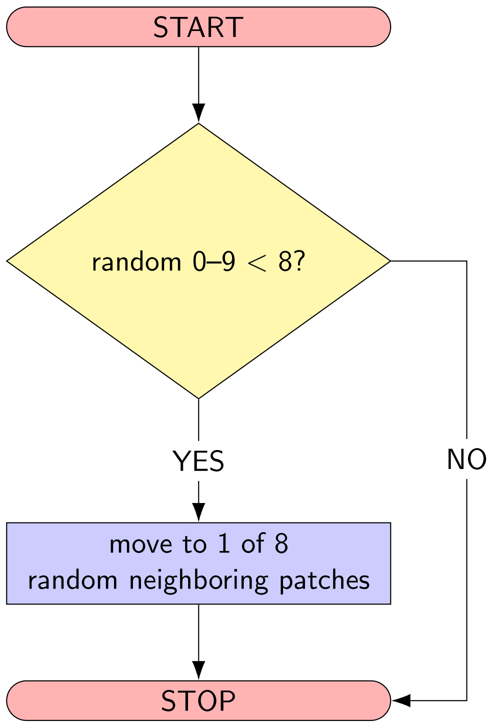

Figure SM9. *Flowchart showing the process* `move`*. Labels on edges represent the value of the variable in the decision point the edge starts from.*

B.7.5. `speak`**.** Each tick, a turtle in the `wandering` `mode` executes the submodel `speak` after executing the submodel `move`. In this submodel, each `wandering` turtle sets its `interlocutor` for another turtle within one patch of itself (if no such turtle exists, the state variable `interlocutor` remains empty). If the `interlocutor` is a second turtle, the first turtle first asks the second turtle to set the second turtle's `interlocutor` state variable to the first turtle, then executes the subsubmodel `exchange-vowels`.

In `exchange-vowels`, the first turtle communicates three vowel tokens (numbers) to the second turtle, one for each vowel category. These tokens are drawn from a normal distribution with the same mean and standard deviation as the list of exemplars in the relevant category. For example, suppose that the first turtle's list of vowel A exemplars has mean 5 and standard deviation 0.1, its list of vowel B exemplars has mean 10 and standard deviation 0.2, and its list of vowel C exemplars has mean 20 and standard deviation 0.15. Then the first turtle will produce a token of vowel A drawn from $\mathcal{N}(5, 0.1)$, a token of vowel B from $\mathcal{N}(10, 0.2)$, and a token of vowel C from $\mathcal{N}(20, 0.15)$. The use of normal distributions here models the variability in production introduced by the activation patterns of exemplars.

Next, the second turtle hears all three tokens and compares them to the minimum and maximum exemplar values in its own categories. Denote the minimum and maximum exemplar values of category N (N standing for A, B, or C) by m_N and M_N, respectively, and the value of `tolerance` by τ. Then the second turtle accepts a vowel token as an instance of category N if and only if the token falls within the range $(m_N - \tau, M_N + \tau)$. For instance, suppose the second turtle's vowel B has a maximum exemplar value of 22 and a minimum exemplar value of 17, and that its `tolerance` is 3. Then the second turtle would accept vowel tokens with values between 14 and 25, inclusive, as instances of vowel B. Once the second turtle accepts a vowel token as an instance of the vowel category, it *appends* that exemplar to the category as the most recent entry. If this happens, the second turtle will temporarily have a category of exemplars one more than `memory-size`.

When `exchange-vowels` finishes, the second turtle then executes `exchange-vowels` (recall that the second turtle's `interlocutor` is the first turtle). The second turtle now speaks three tokens and the first turtle accepts or rejects them as instances of its own vowel categories. This can be interpreted as the assumption that two language users contribute equally (in terms of speaking and hearing vowels) to a conversation.

Figure SM10 on the next page illustrates this submodel as a flowchart.

B.7.6. `go-home`. Each tick every turtle in the `returning mode` executes the submodel `go-home`. In this submodel the turtle first sets their `mode` to `"returning"`, then checks if it is not already on its `home-patch` (the patch where it spawned). If so, it sets its `mode` to `"wandering"`. If not, it faces its `home-patch` and moves forward 1 patch towards it. This submodel models language users returning home from their travel destination. Figure SM11 on the page after the next illustrates this submodel.

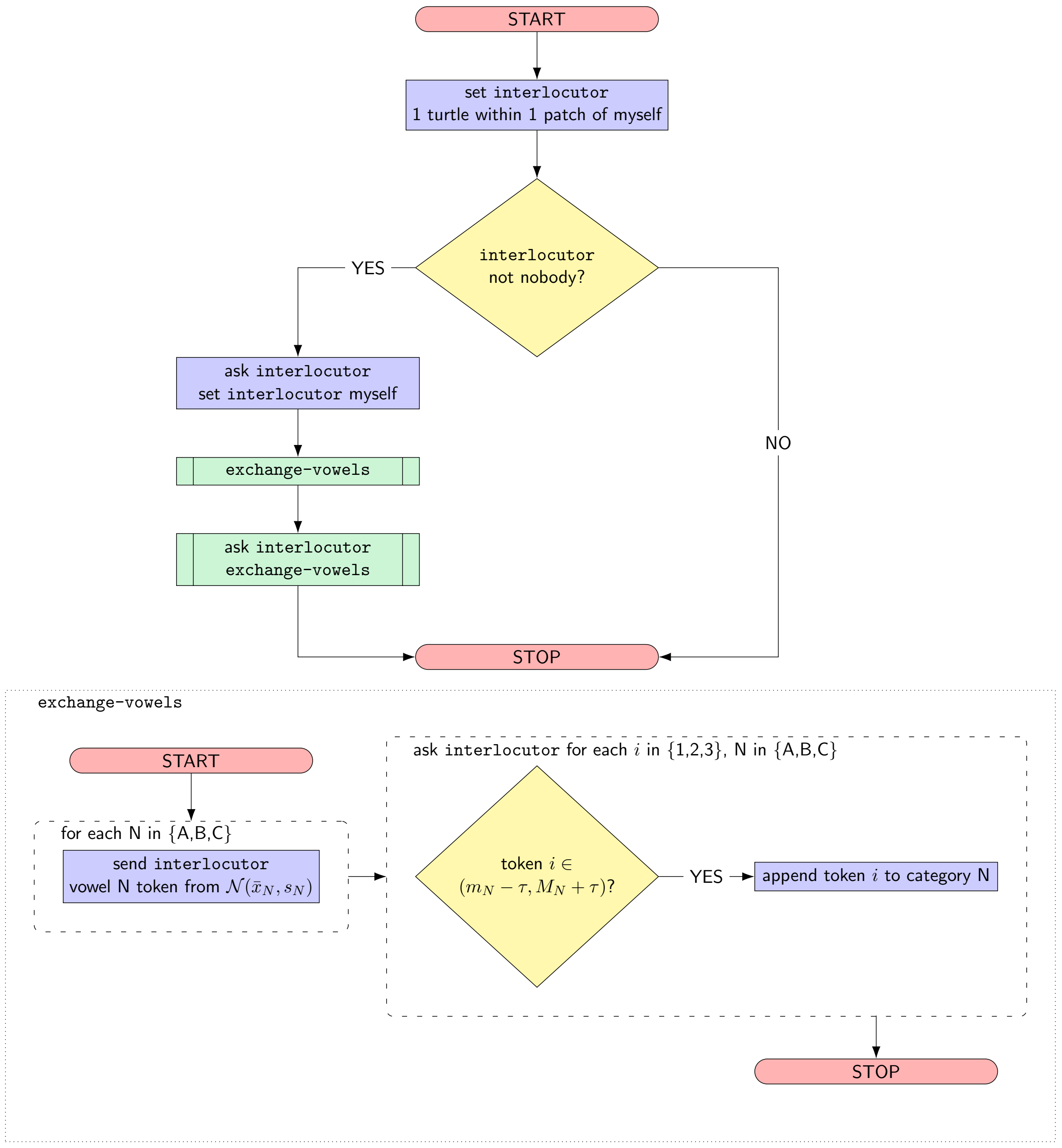

Figure SM10. *Flowchart showing the process* `speak`*. The flowchart for the subsubmodel* `exchange-vowels` *is indicated in a dotted box below. For-each loops are indicated in rounded dashed boxes with the loop variable in the top left corner. $\mathcal{N}(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. $\bar{x}_N$ denotes the mean of the exemplars in category N and s_N the (sample) standard deviation of the exemplars in category N. Finally, m_N, M_N, τ stand for the minimum and maximum exemplar values in category N and the value of* `tolerance`*, respectively.*

]

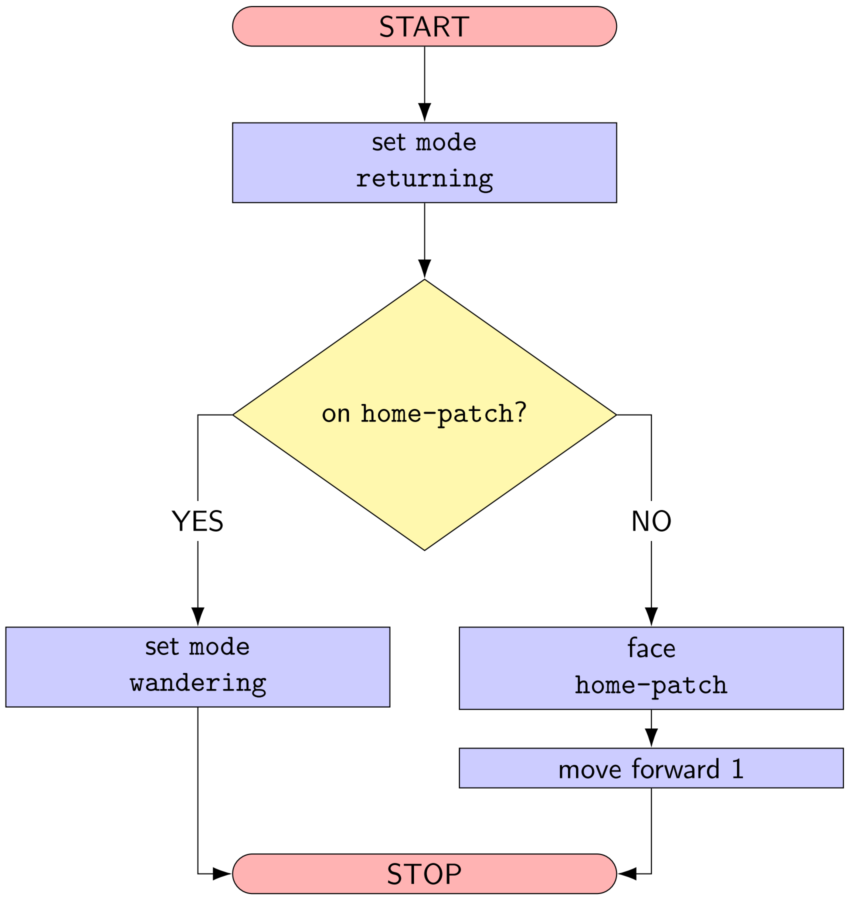

Figure SM11. *Flowchart showing the process* `go-home`*.*

B.7.7. `update-vowels`**.** Each tick every turtle executes the submodel `update-vowels`. This submodel consists of two subsubmodels: `clean-vowels` and `drift-vowels`. First, each turtle executes `clean-vowels`. In this subsubmodel, the turtle removes the oldest exemplar from each of its vowel categories A, B, and C if the length of the category exceeds `memory-size`. Note that this assumes the turtles will only every have categories at most one longer than `memory-size`—this follows from the behavior of the `speak` submodel (§ B.7.5).

Next, each turtle executes the subsubmodel `drift-vowels`. In this subsubmodel, lower vowel categories push higher vowel categories up when they get too close. Note that higher vowel categories cannot push down lower vowel categories. This asymmetry is based on the base model [SM8], in which only shifts in one direction are allowed in order for any shifts to happen. Since two vowel categories being "too close" is a symmetric relation, we must introduce some asymmetry into the system to observe a change. Explaining this asymmetry is a challenge for linguistic theory but is empirically observed as chain shifts.

The mechanism by which vowels drift is as follows. When a turtle has two too-close vowel categories, it adds a higher exemplar to the higher category to gently push it away from the lower one. Suppose for example the higher category is vowel B and the lower category is vowel A. Then the value of the exemplar newly added to category B is given by the following equation:

$$x'_B = \bar{x}_B - \eta(\bar{x}_B - \bar{x}_A - \delta), \tag{B.3}$$

where $\bar{x}_A$ and $\bar{x}_B$ are the mean vowel values of categories A and B, respectively, δ is the `stable-distance` parameter, η is a `drift-rate` parameter, and x'_B is the value of the exemplar newly added to $\bar{x}_B$. An analogous equation holds for exemplars added to category C:

$$x'_C = \bar{x}_C - \eta(\bar{x}_C - \bar{x}_B - \delta), \tag{B.4}$$

but note that no exemplars are added to category A in this submodel. The `stable-distance` parameter represents the smallest distance between the means of two vowel categories a turtles is willing to tolerate, and the `drift-rate` parametrizes how quickly a turtle adjusts its vowel category in response to two intolerably close vowel categories. The right-hand side of equation B.3 approaches $\bar{x}_B$ from above as $\bar{x}_B - \bar{x}_A$ approaches δ from below, so that the pushing effect is weaker the closer together the actual distance between the categories $\bar{x}_B - \bar{x}_A$ is to the stable distance between the two categories and vice versa.

Over time, the accumulation of slightly higher exemplars causes the turtle's higher vowel category to shift upwards, until the distance between it and the category it was too close to matches the `stable-distance`. This represents the vowel system change occuring for an individual language user. A flowchart of this submodel is provided in Figure SM12 on the next page.

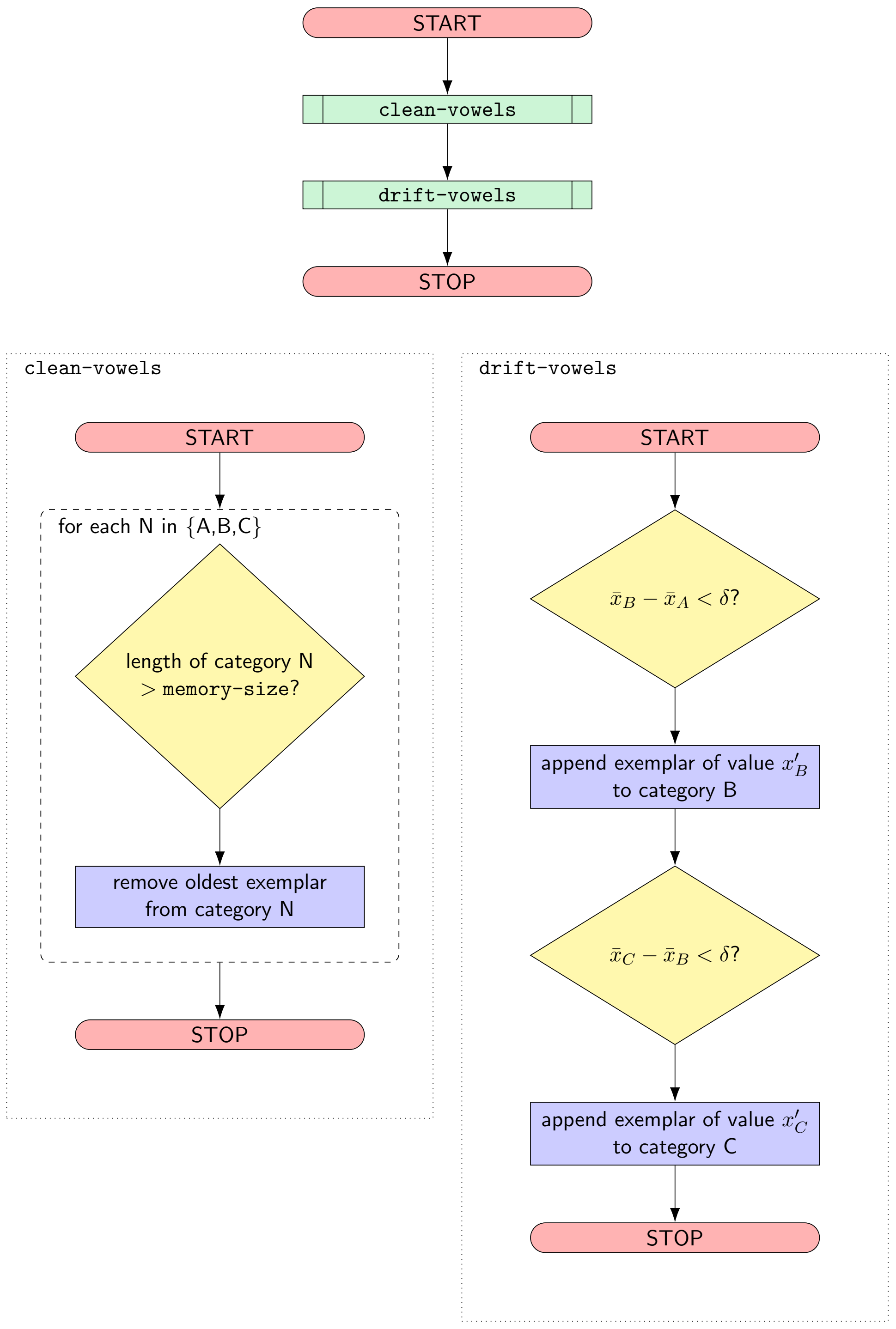
START
clean-vowels
drift-vowels
STOP
clean-vowels
START
for each N in {A,B,C}
length of category N > memory-size?
remove oldest exemplar from category N
STOP
drift-vowels
START
$\bar{x}_B - \bar{x}_A < \delta$?
append exemplar of value x'_B to category B
$\bar{x}_C - \bar{x}_B < \delta$?
append exemplar of value x'_C to category C
STOP

Figure SM12. *Flowchart showing the process* `update-vowels`*. Flowcharts for the subsubmodels* `clean-vowels` *and* `drift-vowels` *are indicated in a dotted box below. For-each loops are indicated in rounded dashed boxes with the loop variable in the top left corner.*

REFERENCES

[1] P. Boersma, Sound change in functional phonology, (1997).

[2] P. Eckert, Adolescent social structure and the spread of linguistic change, Language in society, 17 (1988), pp. 183–207.

[3] M. Goldrick and J. Cole, Advancement of phonetics in the 21st century: Exemplar models of speech production, Journal of Phonetics, 99 (2023), p. 101254.

[4] M. J. Gordon, Methodological and theoretical issues in the study of chain shifting, Language and Linguistics Compass, 5 (2011), pp. 784–794.

[5] L. L. Holt and A. J. Lotto, Speech perception as categorization, Attention, Perception, & Psychophysics, 72 (2010), pp. 1218–1227.

[6] W. Labov, Transmission and diffusion, Language, 83 (2007), pp. 344–387.

[7] W. Labov, The atlas of North American English: Phonetics, phonology, and sound change, Walter de Gruyter, Berlin, 2008.

[8] J. N. Stanford and L. A. Kenny, Revisiting transmission and diffusion: An agent-based model of vowel chain shifts across large communities, Language variation and change, 25 (2013), pp. 119–153.

[9] S. G. Thomason and T. Kaufman, Language contact, vol. 22, Edinburgh University Press Edinburgh, 2001.

[10] W. Wolfram and N. Schilling-Estes, Dialectology and linguistic diffusion, The handbook of historical linguistics, (2017), pp. 713–735.