Interactive Learning of Microtiming in an Expressive Drum Machine

Grigore Burloiu

CINETic UNATC
Bucharest, Romania
grigore.burloiu@unatc.ro

Abstract. The micro (milliseconds) scale is central in coding expressive musical interaction. We introduce `rolypoly~`, a drum machine for live performance that adapts its microtiming, or groove, in relation to a human musician. We leverage state-of-the-art work in expressive performance modelling with LSTM and Seq2Seq architectures, towards real-time application on the micro scale. Our models are pretrained on the Groove MIDI Dataset from Magenta, and then fine-tuned iteratively over several duet performances of a new piece. We propose a method for defining training targets based on previous performances, rather than a prior ground truth. The agent is shown to adapt to human timing nuances, and can achieve effects such as morphing a rhythm from straight to swing.

Keywords: real-time performance, microtiming, expressive accompaniment, interactive machine learning, drumming

1 Introduction

Since the 1980s, musicians and producers have used drum machines as creative partners, imbuing them with a "live" touch (Brett, 2020) or, conversely, leveraging their "mechanical" nature as a postmodern aesthetic device (Bennett, 2017).

With this work we explore the coupling between drum machine and human player, with a view to extending their mutual dynamics computationally (d'Inverno & McCormack, 2015). We centre on the microtiming scale as a locus of expressive music interaction (Leman, 2016). By separating timing from the higher levels in a timescale hierarchy, we model the role of drummers as timekeepers in a group, whose inner-beat groove might react to other players, while holding down a steady tempo.

We delineate a set of design principles for a new musical agent. `rolypoly~` is a score-driven[1] drum machine with the following characteristics:

[1] Rowe (1992) distinguishes between *score-driven* and *performance-driven* systems. The latter generally relate to situations of improvisation or "jamming", which are outside our current scope. We deal with the standard case of an ensemble playing a composed piece, with the machine-drummer not necessarily being exposed to the parts of its partners beforehand.

- *lightweight*: inference must be fast enough to run in real-time on an average, accessible computer. Our evaluations were conducted on a laptop with an i5 CPU and 8GB RAM, without a dedicated GPU.
- *audio-interactive*: must not only adapt to an incoming sound stream (hereafter called *target* audio), but also "listen" to a coupling of the former with its own output.
- *progressive/scalable*: must not simply rely on a static dataset; rather, it builds up a performance corpus, learning a piece or a style from the ground up.
- *score-agnostic*: does not require (but may incorporate) symbolic specification of the parts played by other musicians.

These specifications follow the mechanics of ensembles in various genres, where the timekeeper builds up a "feel" for the music without memorising a complete score of the piece. Moreover, this design aligns with the workflow of modern digital music production, where an artist may program a rhythm track and then play a figure which remains unspecified in symbolic notation.

The purpose of `rolypoly~` is to couple the groove of the drum track with the timing inflexions of the target audio, in real time, while building up a schema of the structure and style of the music over repeated performances, analogously to the way human players gradually assimilate music and bond together.

Eventually our goal is to develop a multi-agent system with the following functions: expressive interpretation (of a symbolic part or program), coordination (with live musical signal(s)), self-reflection and proactivity (via a cognitive architecture that also allows user feedback), and time-awareness from the micro (milliseconds) scale to the supra (across performances) (Roads, 2004).

The present work describes the agent active on the micro scale. The source code, trained models[2] and a demo video[3] are available online.

2 Related Work

In studies on tempo and time-shift representation, Honing (2001, 2005) posits that global tempo curves alone cannot account for the alterations observed in performances of the same material at different speeds. Nevertheless, score-driven automatic accompaniment has traditionally worked by computing such a curve to drive the warping of a backing track (Raphael, 2010; Arzt & Widmer, 2010; Cont, 2011). Increasingly however, attention is also being paid to the *interpretation* of real-time accompaniment on the micro scale (Xia et al., 2015; Cancino-Chacón et al., 2017; Maezawa, 2019).

Expressive machine drumming has mostly been approached as an offline process. Timing and dynamics are the most frequently studied expressive parameters, which is also reflected in "groove" and "humanisation" functions implemented in commercial software like Reason or Ableton Live, where timing and loudness profiles can be captured from one clip and applied to another, or slightly

[2] See `https://github.com/RVirmoors/rolypoly/tree/master/py`.
[3] See `https://youtu.be/UHBIzfc5DCI`.

randomised to add a non-deterministic feel. However, microtiming adjustments can have counterintuitive effects: for instance, systematic deviations to basic rhythms have been found to damage groove and naturalness perception (Davies et al., 2013). By leveraging advances in natural language processing, recurrent neural network (RNN)-based machine learning architectures have been shown to produce state-of-the-art expressive outputs (Jeong et al., 2018; Gillick et al., 2019; Oore et al., 2020).

RNNs have also been conditioned on human partner data to compose (Makris et al., 2019) and perform (Castro, 2019) rhythm patterns. However, to our knowledge no real-time system exists that extends this coupling to the microtime scale.

3 The `rolypoly~` Microtiming Agent

We propose a recurrent neural agent that generates rhythm microtiming in real time, following the design requirements laid out in the introduction.

3.1 Data Representation

Input. The piece to be interpreted is represented as a sequence of feature vector rows, each representing the t-th set of drum hits, with the following components:

- drums to hit: one-hot encoded over a set of 9 categories, covering kick, snare, hi-hats, toms and cymbals, following (Gillick et al., 2019);
- timestep t duration, in milliseconds (how long before the next hit);
- local tempo, as specified in the score in beats per minute;
- local time signature, as one value (e.g. 3/4 and 6/8 both resolve to 0.75);
- beat phase, a fractional value (the relative position in a bar);
- target audio descriptor[4] (previous hit)
- $\widehat{diff_{t-1}}$, drum-target attack distance (previous hit; as fraction of bar)

The online, causal nature of the system means that, at the moment of inference, it lacks access to the target audio input at or close to the time of the current hit. In fact, since the audio corresponding to hit $t-1$ helps determine the microtiming offset of hit t, it makes sense to parse it into the feature vector for timestep t.

The drum-target offset, $\widehat{diff_t}$, is negative if the target audio hits before the drum[5], and positive otherwise. The closest target attack is picked, within the

[4] We tested several spectral descriptors but none was found significantly superior. In Section 5 we discuss prospective strategies for this machine listening task.

[5] We used a very fast onset detection algorithm developed by R Constanzo and PA Tremblay, available at `https://discourse.flucoma.org/t/real-time-onset-detection-example/163/68`.

window $(t - \frac{dur_t}{3}, t + \frac{dur_t}{3})$, for dur_t the duration of the hit at timestep t.[6] If no target attack is recorded in this interval, $\widehat{diff_{t-1}}$ is carried over.

Output. Each feature row corresponds to a drum microtiming offset, y_t, with its corresponding estimation $\hat{y}_t$ determining when in relation to the absolute notated time the drum hit will actually be triggered. In a conventional supervised learning pipeline, the y labels are given at the outset, then used to train the model, which can finally perform inference on new data. Our online, adaptive approach, means that y is not given: it can only be obtained after and as a function of a performance.

We define a variable d_t at timestep t as the cumulated realised offsets of score-to-drum and (variance-adjusted) drum-to-target:

$$d_t = \hat{y}_t + A \cdot \frac{\widehat{diff_t}}{\sigma_{\widehat{diff}}/\sigma_{\hat{y}}} \ , \tag{1}$$

where $\widehat{diff_t}$ is the drum-target offset observed at timestep t.[7] Since we found empirically that target deviations are up to an order of magnitude wider than drum groove timings, we scale them to match the latter's standard deviation.

We can then use d_t to determine the ground truth drum offsets for training the next iteration of the model, by subtracting its mean (to keep outputs centred around zero) and again applying deviation normalisation:

$$y_t = B \cdot \frac{d_t - \mu_d}{\sigma_d/\sigma_{\hat{y}}} \ . \tag{2}$$

A and B above are hyperparameters, controlling the weighting of the target offsets and the cumulated offsets, respectively. Their default setting is 1, allowing the timing output to adapt gradually, without major fluctuations. For rhythm morphing (see Section 4.2) they can be set to cancel out the effect of variance scaling. Finally they, as well as the learning rate and number of train epochs, may be optimised through gradient descent as part of a *meta learning* pipeline, by using them as labels and querying a deep net with historical performance features, plus a desired future $\sigma_{\widehat{diff}}$.[8]

[6] Attacks that fall within the middle third of a drum hit duration are considered "off-beat" notes and added to the dataset as no-hit rows with the position quantised to 24 steps per bar. For the first iteration they are flagged as tentative and skipped during training, but a subsequent performance can confirm them as permanent. Only one such note, with a length > 10ms, may be added per window, per performance.

[7] Since training occurs after inference, we now have access to all $\widehat{diff}$ values recorded in a performance. Therefore, linking $\widehat{diff_t}$ to timestep t is simply a matter of shifting its column one step into the future.

[8] This component is currently in an experimental stage. Eventually a meta model might maintain a database of model states, and choose among them mid-performance depending on running $\sigma_{\widehat{diff_t}}$ values and other metrics.

3.2 Model Design

We propose two RNN-derived architectures that predict each following drum timing based on the input features received in real time. Both are defined and trained using the PyTorch library (Paszke et al., 2019), and communicate via OSC to Max, which handles the audio playback and analysis.

The first model is built on a 2-layer unidirectional LSTM network (Hochreiter & Schmidhuber, 1997) with 256 hidden units fed into *tanh* nonlinear activations. The result is passed through a linear layer with a single output, $\hat{y}$.

The second model is a simplification of the Seq2Seq architecture described in (Gillick et al., 2019). Sequence to sequence models (Sutskever et al., 2014) comprise an encoder network, producing a latent vector z as a representation of a source sequence, and a decoder network which, primed with z, outputs a corresponding target sequence. In our case the source sequence is the complete pre-performance input dataset (*sans* the unrealised audio-related features), and the target is the performance dataset up to the current timestep. The encoder is a bidirectional LSTM, whose hidden units can learn both past and future dependencies at every timestep. The decoder is a 2-layer unidirectional LSTM with 256 hidden units. As with the first model, a *tanh* nonlinearity and final linear layer project the decoder output to a one-dimensional activation, $\hat{y}$.

To illustrate the difference between the proposed models with an analogy, the unidirectional LSTM is equivalent to sightreading a drum part, while the Seq2Seq architecture enables memorising the material "backwards and forwards."

3.3 Training and Deployment

A dataset comprising several performances of the same piece with precise annotations of the notes played by the drummer in relation to the symbolic score, which would be needed to train our proposed models, does not exist.[9]

Fortunately, we are able to use transfer learning to train initial configurations for our models, by processing a drums-only performance dataset. We used the Groove MIDI Dataset (GMD)[10] from Magenta (Gillick et al., 2019), the largest existing dataset of expressive drumming.

All MIDI files are parsed with the `pretty midi` library (Raffel & Ellis, 2014), with short takes ($\leq$ 1s or containing a single bar) being pruned out. Feature vectors are then extracted according to Section 3.1 and labelled with the residual drum offsets,[11] since no ensemble performance data exists. Train, validation and test labels are retained from the GMD specification. The models are trained using the Adam optimiser (Kingma & Ba, 2014) with a 0.001 initial learning rate and 0.3 dropout for the 2-layer networks.

[9] A list of research datasets related to music information retrieval is maintained at `https://www.audiocontentanalysis.org/data-sets/`.

[10] available at `https://magenta.tensorflow.org/datasets/groove`.

[11] Thus, y measures the distance to the drum hit from its quantised position. While (Gillick et al., 2019; Makris et al., 2019) used 16 steps per bar, we chose a quantisation step of 24, to better account for triplets and swing.

These pretrained models are ready to use as offline, audio-agnostic expressive drum part interpreters (similarly to (Gillick et al., 2019)) but can also be fine-tuned through subsequent performances. Fine-tuning takes place in an offline phase between live human-machine duet takes, where we also allow the use of (a subset of) the GMD as a validation set to avoid overfitting.

Both models use a mean squared error loss. For training, only hits with a corresponding target onset are fed into the loss function, to avoid onset detection errors or outliers propagating over sustained target notes. In the validation phase, all realised hits count towards computing the loss.

4 Evaluation

Evaluating interactive machine learning systems is a difficult undertaking (Boukhelifa et al., 2018). In addition, adaptive musical agents pose specific challenges (Pasquier et al., 2016). By definition, the execution results are in flux, which impacts the assessment methodology. For the pretrained models we provide error measurements. We then conduct two experiments that highlight our adaptive training pipeline's viability for different use cases.

4.1 Baseline

Table 1 shows error metrics for our proposed models, pretrained as described in Section 3.3. We provide a notebook[12] allowing for reproduction of the results.

Model	MSE [/bar]	MSE [16th note]
No μtiming	0.000072	0.018679
Basic LSTM	0.000060	0.015505
Seq2Seq	**0.000055**	**0.014260**

Table 1. Mean Squared Error rates of the proposed models over the GMD test set.

Note that, while our error metrics are significantly lower than those reported by Gillick et al. (2019), we also start from a tighter quantised, zero-offset baseline. This is due to the different objectives of the two model classes: GrooVAE *generates* patterns on a 16th note grid, while `rolypoly~` *performs* particular piece timings[13]—hence the finer quantisation of our GMD parsing.

4.2 Experimental

Starting from the pretrained models, our first experiment tests the ability to morph rhythms away from a notated pattern, by playing a different timing sequence on top. As target audio we use direct input electric guitar.

[12] See `https://github.com/RVirmoors/rolypoly/blob/master/py/rolypoly.ipynb`.

[13] Also for this reason, we predict single overall timing offset values, rather than individual offsets for each drum, like GrooVAE. Spreading concurrent hits apart constitutes *flamming*; generating such drumming flams is outside our current scope.

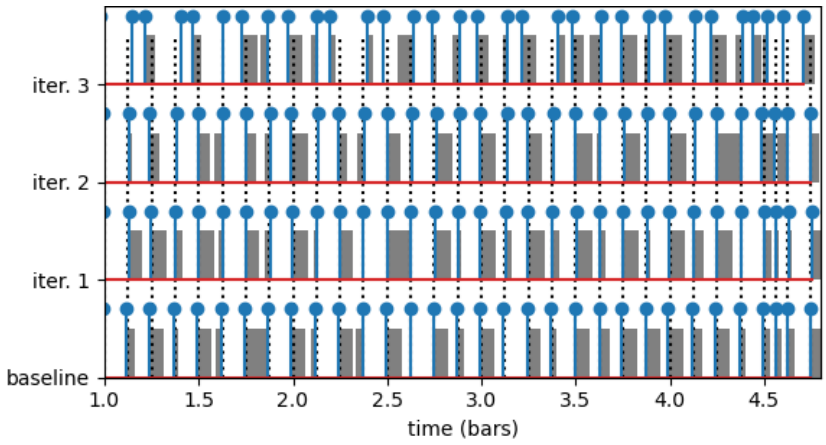

Fig. 1. Rhythm morphing over three training iterations. Score: dotted lines. $\hat{y}$: blue stems. $\widehat{diff}$: grey bars. Transition from straight (bottom) to swinging (top).

Figure 1 pictures the Seq2Seq model transitioning from a straight 4/4 beat to a "swing" shuffle, where offbeats are pushed slightly later. To obtain the effect, we expanded the guitar offset detection window to $(t - \frac{dur_t}{2.5}, t + \frac{dur_t}{2.5})$. Similarly we were able to morph the pattern `x--x--x-` to three equally-distanced triplets.[14]

The second experiment simulates the typical use case, where a song is performed multiple times and the basic LSTM model learns incrementally after each take. In Figure 2 we plot the evolution of drum-target offsets over several iterations. The agent is able to "tame" the variance of the guitar, and visibly adapts to structural patterns in the piece.

Inference on the Seq2Seq model takes around 40ms on an i5 machine, which is too slow for the shortest hits in the song; a possible remedy would be to predict several hits ahead. Moreover, we found a head-to-head comparison of the two proposed models using the same recorded target audio not only theoretically untenable (frozen guitars don't react to realised drums), but also practically unfeasible, due to the microtime jitter inherent in our Python-Max setup, which causes stochastic drift over time. In this context, while the Seq2Seq model performs marginally better in the GMD pretraining phase, its overall superiority over the basic LSTM model remains an open question.

5 Discussion and Future Work

This paper introduced `rolypoly~`, an adaptive drum machine for real-time performance. The project is open source, and to our knowledge unique in coupling human-computer microtiming and learning iteratively without prior training on duet data. Beyond this initial stage, there is ample room for further research.

[14] See also `https://github.com/RVirmoors/rolypoly/tree/master/_csmc-mume2020`.

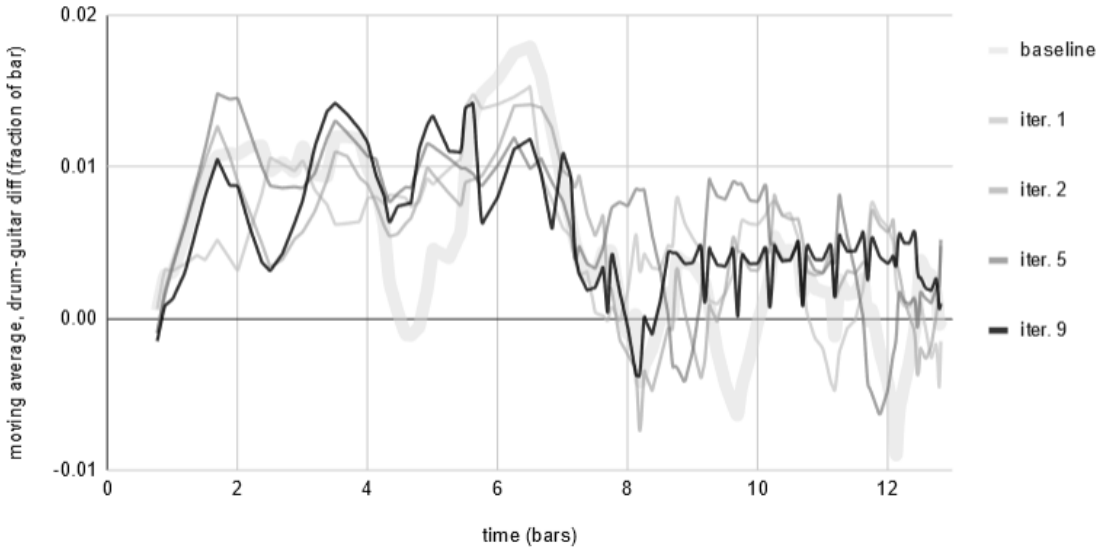

Fig. 2. Moving average (period: 6 hits) of the timing difference between drum hit and guitar onset. The baseline (thick, light gray) is the basic LSTM model pretrained on GMD. Subsequent training runs for 5 epochs (Adam w/ l.r. 0.001) after each take.

Our process of defining y seeks to minimise drum-target offset jitter. While our evaluation supports this intuition, it remains an open question whether this formula is optimal or another strategy might prove more musically useful.

Structurally, while the models must remain lightweight, they may benefit from a richer data representation of agent and human actions. One possible route is the learning of a latent feature space of descriptors (Maezawa, 2019). Moreover, a partial cause of the large variance seen in drum-guitar offsets might be the detection step; we are considering other just-in-time algorithms (e.g. audio alignment) to supplement or replace onset detection.

The rhythm morphing exercises raise the question of altering the notated score: once a song is being played differently in a systematic way, it makes sense to modify the underlying symbolic representation. Such a decision could be offered interactively to the user or might be partially automated.

All of the above are to be approached in the context of building a multi-timescale hierarchical system. Rather than treating tasks independently, we plan to design agents that inform each other and interact. The first step in this direction will be to add a tempo modelling (Burloiu, 2016) component.

Presently, we are working on modelling drum hit velocity, and porting the inference code into a Max external using LibTorch, making the system more reliable and accessible, while enabling more musical, human-centred, performer and listener assessments (Boukhelifa et al., 2018; McCormack et al., 2019). Ultimately, as with the drum machines of the past 40+ years, it is user adoption that will drive the viability of such a music performance tool.

References

Arzt, A., & Widmer, G. (2010). Simple tempo models for real-time music tracking. In *Sound and music computing conference (smc).*

Bennett, S. (2017). Songs about fucking: John Loder's southern studios and the construction of a subversive sonic signature. *Journal of Popular Music Studies*, *29*(2), e12209.

Boukhelifa, N., Bezerianos, A., & Lutton, E. (2018, July). Evaluation of Interactive Machine Learning Systems. In J. Zhou & F. Chen (Eds.), *Human and Machine LearningVisible, Explainable, Trustworthy and Transparent* (p. 341-360). Springer. Retrieved from `https://hal.inria.fr/hal-01845018`

Brett, T. (2020, May). Prince's rhythm programming: 1980s music production and the esthetics of the LM-1 drum machine. *Popular Music and Society*, *43*(3), 244–261. doi: 10.1080/03007766.2020.1757813

Burloiu, G. (2016). Online score-agnostic tempo models for automatic accompaniment. In *International workshop on machine learning and music (mml).*

Cancino-Chacón, C., Bonev, M., Durand, A., Grachten, M., Arzt, A., Bishop, L., ... Widmer, G. (2017). The accompanion v0. 1: an expressive accompaniment system. *arXiv preprint arXiv:1711.02427*.

Castro, P. S. (2019, April). Performing Structured Improvisations with pretrained Deep Learning Models. *arXiv:1904.13285 [cs, eess]*.

Cont, A. (2011). On the creative use of score following and its impact on research. In *Smc.*

Davies, M., Madison, G., Silva, P., & Gouyon, F. (2013). The effect of microtiming deviations on the perception of groove in short rhythms. *Music Perception: An Interdisciplinary Journal*, *30*(5), 497–510.

d'Inverno, M., & McCormack, J. (2015). Heroic versus collaborative AI for the arts.

Gillick, J., Roberts, A., Engel, J., Eck, D., & Bamman, D. (2019). Learning to groove with inverse sequence transformations. *Proceedings of the 36th International Conference on Machine Learning, PMLR*, 2269–2279.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Honing, H. (2001). From time to time: The representation of timing and tempo. *Computer Music Journal*, *25*(3), 50–61.

Honing, H. (2005). Timing is tempo-specific. In *Proceedings of the international computer music conference* (pp. 359–362).

Jeong, D., Kwon, T., & Nam, J. (2018). Virtuosonet: A hierarchical attention rnn for generating expressive piano performance from music score. In *Neurips 2018 workshop on machine learning for creativity and design.*

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leman, M. (2016). *The expressive moment: How interaction (with music) shapes human empowerment.* MIT press.

Maezawa, A. (2019). Deep linear autoregressive model for interpretable prediction of expressive tempo. *Proc. SMC*, 364–371.

Makris, D., Kaliakatsos-Papakostas, M., Karydis, I., & Kermanidis, K. L. (2019). Conditional neural sequence learners for generating drums' rhythms. *Neural Computing and Applications*, *31*(6), 1793–1804.

McCormack, J., Gifford, T., Hutchings, P., Llano Rodriguez, M. T., Yee-King, M., & d'Inverno, M. (2019). In a silent way: Communication between ai and improvising musicians beyond sound. In *Proceedings of the 2019 chi conference on human factors in computing systems* (pp. 1–11).

Oore, S., Simon, I., Dieleman, S., Eck, D., & Simonyan, K. (2020). This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, *32*(4), 955–967.

Pasquier, P., Eigenfeldt, A., Bown, O., & Dubnov, S. (2016). An introduction to musical metacreation. *Computers in Entertainment (CIE)*, *14*(2), 2.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (pp. 8026–8037).

Raffel, C., & Ellis, D. P. (2014). Intuitive analysis, creation and manipulation of midi data with pretty midi. In *15th international society for music information retrieval conference late breaking and demo papers* (pp. 84–93).

Raphael, C. (2010). Music plus one and machine learning. In *International conference on machine learning (icml)*.

Roads, C. (2004). *Microsound*. MIT press.

Rowe, R. (1992). *Interactive music systems: machine listening and composing*. MIT press.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Xia, G., Wang, Y., Dannenberg, R. B., & Gordon, G. (2015). Spectral learning for expressive interactive ensemble music performance. In *Ismir* (pp. 816–822).