Article

Not peer-reviewed version

Beyond Tokens: Introducing Large Semiosis Models (LSMs) forGrounded Meaning in Artificial Intelligence

Luciano Silva *

Posted Date: 22 April 2025

doi: 10.20944/preprints202504.1830.v1

Keywords: Large Semiosis Models (LSMs); symbol grounding; semiotics; Large Language Models (LLMs); grounded AI; artificial intelligence (AI)

 doi:10.20944/preprints202504.1830.v1

Article

Beyond Tokens: Introducing Large Semiosis Models (LSMs) for Grounded Meaning in Artificial Intelligence

Luciano Silva

Bioquaintum Research and Development; luciano.silva@bioquaintum.io

Abstract: Large Language Models (LLMs) represent a significant leap in artificial intelligence, demonstrating remarkable capabilities in processing and generating human-like text. However, their operational paradigm, primarily based on statistical correlations between symbolic tokens (signifiers/representamens), reveals fundamental limitations concerning genuine understanding and semantic grounding. This paper posits that semiotic theory, drawing upon the foundational frameworks of Ferdinand de Saussure and Charles Sanders Peirce, offers essential analytical tools for diagnosing these deficiencies and proposing advancements. We argue that LLMs predominantly model the Saussurean *signifier* or the Peircean *representamen*, remaining largely disconnected from the conceptual *signified* or the referential *object* and meaning-effect *interpretant*. To address this critical semantic gap, we introduce the concept of **Large Semiosis Models (LSMs)**. LSMs are conceived as next-generation AI systems architected to explicitly model the triadic or dyadic relationships inherent in sign processes, thereby integrating representations of meaning and reference with symbolic manipulation. This paper outlines the theoretical rationale for LSMs, delineates their potential capabilities—including enhanced reasoning, robust grounding, and meaningful interaction—and proposes distinct implementation strategies inspired by Saussurean and Peircean semiotics. Conceptual Python implementations using the LangChain framework are sketched to illustrate pathways for adapting current technologies towards LSM development. We conclude that the pursuit of LSMs constitutes a vital research trajectory for fostering AI systems exhibiting greater robustness, reliability, and semantic intelligence.

Keywords: Large Semiosis Models (LSMs); symbol grounding; semiotics; Large Language Models (LLMs); grounded AI; artificial intelligence (AI)

1. Introduction

The contemporary landscape of Artificial Intelligence (AI) has been profoundly reshaped by the emergence and rapid development of Large Language Models (LLMs). Architectures such as OpenAI's GPT-4 [1], Google's Gemini family [2], Anthropic's Claude 3 [3], and Meta's Llama series [4] exemplify this paradigm shift. These models, trained on vast datasets of text and code, exhibit unprecedented fluency in natural language processing tasks, demonstrating proficiency in generation, translation, summarization, and complex question answering. Their capacity to discern and replicate intricate statistical patterns within linguistic data has enabled significant advancements across numerous applications and research domains, fueling speculation about pathways towards more general forms of artificial intelligence.

Despite these remarkable achievements, a growing body of research and critical analysis highlights inherent limitations that challenge the depth of understanding possessed by LLMs [5,6]. A central concern revolves around the persistent **symbol grounding problem** [7], which questions how symbolic representations manipulated by computational systems can acquire intrinsic meaning connected to the external world or perceptual experience. In the context of LLMs, tokens—the basic units of processing—are primarily defined by their distributional relationships with other tokens within the

 doi:10.20944/preprints202504.1830.v1

training corpus, lacking robust mechanisms to link them to non-linguistic referents or embodied sensations [8,9]. This fundamental disconnect is believed to underlie several observed shortcomings.

One significant issue is the propensity of LLMs to generate "hallucinations" , outputs that are linguistically plausible and coherent but factually incorrect, nonsensical, or untethered from verifiable reality [10,11]. Such unreliability poses significant risks for applications requiring high factual accuracy and trustworthiness, with ongoing research exploring both detection and mitigation strategies (e.g., [12]; [13]). Furthermore, the reasoning capabilities of LLMs, particularly concerning causality, common sense, and complex logical inference, often prove fragile [6,14,15]. While they can retrieve and recombine patterns resembling reasoning steps found in their training data, they frequently struggle with novel problems or scenarios requiring a deeper, model-based understanding of underlying mechanisms or principles [16,17]. These limitations have led some researchers to characterize LLMs as sophisticated mimics, or "stochastic parrots" [18], adept at manipulating linguistic form but lacking genuine semantic comprehension or intentionality [19].

To gain deeper theoretical insight into these challenges and to inform the development of more capable AI systems, this paper turns to **semiotics**, the interdisciplinary study of signs, symbols, and processes of signification. Semiotics provides structured frameworks for analyzing how meaning is constructed and interpreted within systems of representation. The foundational theories of Ferdinand de Saussure [20] and Charles Sanders Peirce [21] are particularly pertinent. Saussure's dyadic model emphasizes the arbitrary relationship between the *signifier* (the sign's physical form) and the *signified* (the concept it evokes) within a defined linguistic system (*langue*). Peirce's triadic model describes semiosis as a dynamic process involving a *representamen* (the sign vehicle), an *object* (that which the sign stands for, including its real-world counterpart, the dynamic object), and an *interpretant* (the meaning-effect or understanding produced in an interpreter, often another sign).

Applying these semiotic frameworks reveals that current LLMs operate predominantly at the level of the Saussurean *signifier* or the Peircean *representamen*. They excel at learning the statistical structure and combinatorial rules governing these formal elements within massive datasets. However, they lack robust internal representations corresponding to the *signified* (stable concepts) or mechanisms to reliably connect representamens to dynamic *objects* in the external world [9]. Consequently, the "meaning" captured by LLMs remains largely distributional and correlational, rather than conceptual or referential. The internal states and generated outputs of LLMs might be viewed as computational correlates of Peircean *interpretants*, but they are effects within a closed symbolic system, not necessarily grounded interpretations linked to external reality or internal conceptual understanding.

Based on this semiotic diagnosis, we propose the concept of **Large Semiosis Models (LSMs)** as a necessary evolution beyond the current LLM paradigm. LSMs are envisioned as AI systems architected to explicitly model and integrate the core components of semiotic processes. This entails moving beyond token manipulation to incorporate: (a) representations of linguistic or other sign forms (signifiers/representamens), (b) structured representations of concepts or meanings (signifieds/interpretants), and critically, (c) mechanisms for linking these symbols to representations of, or interactions with, an external environment or world model (objects). The fundamental goal of LSMs is to bridge the semantic gap by embedding grounding mechanisms directly into their architecture, thereby fostering systems capable of more robust reasoning, reliable performance, and semantically meaningful interaction.

This paper aims to delineate the theoretical foundations for LSMs, drawing explicitly on Saussurean and Peircean insights to critique current LLMs and motivate the LSM concept. We will explore the potential capabilities afforded by such models and discuss distinct architectural and implementation strategies corresponding to the dyadic and triadic semiotic frameworks. To provide concrete illustrations, conceptual Python code sketches leveraging the popular LangChain framework [22] will be presented, demonstrating potential pathways for adapting existing tools towards LSM development. Finally, the paper concludes by summarizing the arguments, acknowledging the significant research

 doi:10.20944/preprints202504.1830.v1

challenges, and advocating for the pursuit of LSMs as a crucial direction for advancing artificial intelligence towards greater semantic depth and capability.

2. State of the Art: Large Language Models (LLMs) and the Semantic Gap

To fully appreciate the motivation for proposing Large Semiosis Models, a detailed examination of the current state of the art, embodied by LLMs, is necessary. This involves understanding their underlying architectural principles and training methodologies, acknowledging their significant achievements, and critically analyzing the pervasive limitations related to semantic understanding that define the "semantic gap."

The dominant architectural paradigm underpinning contemporary LLMs remains the Transformer model [23], although recent variations and potential successors are actively researched (e.g., state space models; [24]). The Transformer's core self-attention mechanism, enabling dynamic weighting of input token relevance, facilitates the capture of complex, long-range dependencies crucial for language understanding. Continued analysis seeks to better understand the internal workings of these attention mechanisms and their contribution to model capabilities [25]. LLMs are pre-trained on increasingly vast and diverse corpora, often combining web text, books, code, and potentially multimodal data, scaling to trillions of tokens [1,2,4]. Self-supervised objectives, primarily autoregressive prediction or masked language modeling, drive this pre-training phase, allowing models to implicitly learn linguistic structures, factual associations, and stylistic nuances from the data distribution [26,27]. This foundation is typically augmented through instruction fine-tuning and alignment techniques, prominently including Reinforcement Learning from Human Feedback (RLHF) or newer variants like Direct Preference Optimization (DPO), aiming to enhance controllability, helpfulness, and adherence to safety guidelines [28,29]. The relentless pursuit of scale, guided by empirical scaling laws [30,31], continues to push the boundaries of model size and computational requirements, although recent work also explores efficiency and the potential of smaller, highly optimized models [32].

The empirical success derived from this paradigm remains impressive. LLMs consistently set state-of-the-art benchmarks across a wide array of NLP tasks, including nuanced generation, complex question answering, translation, and summarization, often exhibiting performance comparable to or exceeding human capabilities on specific benchmarks [1,3]. Their ability to synthesize code across multiple programming languages has also become increasingly sophisticated [4]. Furthermore, the phenomenon of in-context learning, where models adapt to new tasks based solely on examples provided within the prompt, continues to be a subject of intense study, suggesting powerful pattern-matching and generalization capabilities emerging from scale [27,33]. These achievements underscore the efficacy of large-scale, data-driven approaches in capturing the statistical essence of human language as represented in massive textual datasets.

However, this proficiency in manipulating linguistic form operates alongside persistent and well-documented deficiencies concerning semantic depth and connection to the non-linguistic world, creating a significant "semantic gap." The symbol grounding problem [7] remains a central theoretical challenge. The tokens processed by LLMs derive their functional meaning entirely from the statistical context established during training; they lack inherent, stable connections to perceptual inputs, embodied actions, or abstract concepts independent of their linguistic distribution [8,9]. While multimodal LLMs (e.g., [2]) attempt to bridge this gap by incorporating visual or auditory data, the extent to which this achieves true conceptual grounding versus sophisticated cross-modal correlation remains an open question [34]. The fundamental learning process, primarily based on pattern matching in vast but disembodied datasets, differs markedly from human cognition, which is scaffolded by continuous interaction with a rich physical and social environment [35,36].

This lack of grounding contributes directly to the models' tenuous grasp on referential intent and structured world knowledge. An LLM's use of a term is governed by sequence probability, not by accessing a stable conceptual representation tied to defining properties or causal relationships

 doi:10.20944/preprints202504.1830.v1

[37]. While extensive factual information is implicitly encoded, LLMs generally lack an explicit, dynamic, and queryable world model, hindering their ability to maintain consistency, track entity states accurately over long interactions, or integrate new information reliably without extensive retraining [38,39]. Their internal "knowledge" often reflects the statistical biases and factual inconsistencies present in the training data, rather than a curated, coherent representation of reality derived through interaction or explicit modeling. Efforts to imbue LLMs with more structured knowledge, perhaps through retrieval augmentation (RAG) or integration with knowledge graphs, represent attempts to mitigate this but often operate as external add-ons rather than core architectural changes [40,41].

The generation of "hallucinations" remains a critical practical challenge stemming from these underlying issues [10]. LLMs frequently produce outputs that are fluent and contextually plausible but factually incorrect or nonsensical, as their generation objective prioritizes statistical likelihood over truthfulness or factual verification [11]. Significant research effort in 2024 focuses on detecting and mitigating these hallucinations, exploring techniques like uncertainty estimation, internal consistency checks, retrieval-augmented generation, and fine-tuning for factuality, yet completely eliminating them remains elusive [12,13,42]. This inherent unreliability continues to impede the deployment of LLMs in domains demanding high factual integrity.

Furthermore, complex reasoning capabilities, particularly those involving multi-step logic, causality, planning, and robust commonsense understanding, remain a significant area of weakness [6,14]. While techniques like chain-of-thought prompting can elicit improved performance on certain reasoning tasks by encouraging explicit intermediate steps [43], LLMs often struggle with compositional generalization, systematicity, and counterfactual reasoning [15,17,44]. Evaluating causal reasoning remains particularly challenging, with evidence suggesting that LLMs often rely on superficial correlations rather than understanding underlying causal structures [45–47]. Their reasoning failures highlight a potential reliance on pattern matching over algorithmic or model-based deduction [16].

The current LLM paradigm, while achieving remarkable feats in linguistic pattern replication, exhibits a fundamental gap concerning semantic depth, grounding, and robust reasoning. They master the statistical correlates of language form but struggle with the underlying meaning, reference, and logical structure in a way that aligns with human comprehension [5,19]. This persistent semantic gap, underscored by the ongoing challenges of grounding, hallucination, and reasoning, provides the central motivation for exploring new architectural paradigms. The Large Semiosis Models proposed herein aim directly at addressing this gap by incorporating principles from semiotics to build systems where signs, meanings, and their connection to the world are explicitly modeled and integrated, paving the way for more truly intelligent and reliable AI.

3. Semiotic Foundations I: The Saussurean Sign and LLM Analysis

To systematically dissect the semantic limitations inherent in current LLM architectures, as outlined in the previous section, we turn to the foundational principles of structuralist semiotics articulated by Ferdinand de Saussure. His framework, primarily concerned with the internal structure of language as a system of signs, provides a powerful lens for analyzing precisely what LLMs capture and what they fundamentally miss. Saussure conceptualized the linguistic sign not as a simple link between a name and a thing, but as a two-sided psychological entity, an inseparable dyad composed of the *signifier* and the *signified* [20]. Understanding these components and their relationship within the broader linguistic system (*langue*) is crucial for evaluating the operational paradigm of LLMs.

The **signifier** (*signifiant*), according to de Saussure [20], is the 'sound-image' or psychological imprint of the sign's physical form—the sensory aspect, be it auditory (the pattern of sounds in speech) or visual (the sequence of letters in writing). It is not the physical sound wave or ink marks themselves, but rather their mental representation. In the context of LLMs, the operational units—tokens, typically representing words, subwords, or characters—function analogously to Saussurean signifiers. These tokens are the discrete, formal elements that the models process and manipulate. LLMs operate

 doi:10.20944/preprints202504.1830.v1

directly on these textual representations, learning patterns based on their sequential arrangement and co-occurrence. Furthermore, Saussure identified the **linearity of the signifier** as a fundamental principle: signifiers unfold sequentially in time (speech) or space (writing), one element after another. LLM architectures, particularly the sequential processing inherent in Transformers [23], directly mirror this linearity. They process input tokens and generate output tokens in a sequential manner, where the position and order of signifiers are paramount to the patterns learned and the outputs generated. The models' ability to handle complex syntax and dependencies relies heavily on capturing this linear structure.

The other face of the Saussurean sign is the **signified** (*signifié*), defined as the concept or idea evoked by the signifier [20]. It is a mental construct, an abstract notion shared, to varying degrees, by users of the same language system. For instance, the signifier "tree" evokes the signified concept of a perennial plant with a trunk and branches. Importantly, Saussure deliberately distinguished the signified (the concept) from the *referent* (the actual physical tree in the world), focusing his analysis primarily on the internal structure of the language system itself, bracketting the complexities of direct world reference. The signified is thus an element within the realm of mental representations, a unit of meaning defined within the linguistic system.

A cornerstone of Saussurean theory is the **arbitrariness of the sign** [20]. This principle asserts that the connection between the signifier and the signified is fundamentally conventional and unmotivated; there is no inherent reason why a particular sequence of sounds or letters (e.g., /dog/ or d-o-g) should be linked to the concept of a canine companion. Different languages employ entirely different signifiers for the same signified (cf. French 'chien', German 'Hund'), demonstrating the lack of any natural or necessary bond. This arbitrariness necessitates the existence of a shared social contract, a collective system of conventions—what Saussure termed ***langue***—for signs to function communicatively. Meaning is not inherent in the sign itself but is established and maintained through collective agreement and usage within a language community.

Crucially, within the system of *langue*, the meaning or **value** (*valeur*) of a sign arises not from a direct, positive correspondence to a concept, but differentially and oppositionally [20]. The signified associated with a signifier is defined primarily by what it is *not*—by its difference from other related signifieds within the same conceptual field. For example, the value of the English word "sheep" is partly defined by its opposition to "mutton" (the meat vs. the animal), a distinction not necessarily made by a single signifier in other languages (e.g., French 'mouton'). Similarly, the value of "red" arises from its contrast with "blue," "green," "yellow," etc. Meaning, therefore, is relational; it resides in the systematic network of differences and similarities that structure the entire linguistic system. This relational concept of value resonates strongly with the operational principles of LLMs. The vector embeddings assigned to tokens (signifiers) capture precisely this kind of relational meaning [48,49]. The position of a token's vector in the high-dimensional embedding space—its proximity to some vectors and distance from others—is determined by its distributional properties, i.e., the contexts in which it appears relative to all other tokens in the training data. LLMs thus implicitly learn the Saussurean 'value' of signifiers by modeling the intricate web of their distributional relationships, effectively capturing a statistical representation of the *langue* as manifested in the corpus [50]. The ability of embeddings to capture semantic similarities (e.g., 'king' is to 'queen' as 'man' is to 'woman') is a direct consequence of modeling these systemic, differential relationships between signifiers. Semiotic analyses emphasize how such systems construct meaning through internal relations [51].

Synthesizing this analysis, we can posit that LLMs represent remarkably sophisticated computational models of the **Saussurean system of signifiers**. They demonstrate an unparalleled ability to learn the complex statistical regularities governing the linear arrangement (syntax) and relational values (distributional semantics) of tokens within the vast textual corpora representing approximations of *langue*. Their success in generating fluent, coherent text stems directly from their mastery over these formal, systemic properties of the signifier layer. They effectively map the intricate network of

 doi:10.20944/preprints202504.1830.v1

differences and oppositions that, according to Saussure, constitutes the primary source of linguistic value.

To illustrate this focus on the signifier level and the potential disregard for the distinct signifieds, consider the following practical example using the LangChain library to interact with an LLM. We will use a scenario involving homonyms – words with the same signifier (spelling/sound) but different signifieds (concepts).

```
import os
from langchain_openai import ChatOpenAI
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from dotenv import load_dotenv

# Ensure API key is loaded (replace with your method if not using .env)
load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")

# Initialize a capable LLM (e.g., GPT-3.5 or GPT-4)
# Using a slightly less powerful model might make the potential failure mode clearer,
# but even powerful models operate based on signifier patterns.
llm = ChatOpenAI(temperature=0, model_name="gpt-3.5-turbo", openai_api_key=
    openai_api_key)
# or use "gpt-4"

# Define the prompt template with a scenario using the homonym "bat"
prompt_text = """
Consider the following short story:
A baseball player carefully selected his favorite wooden bat from the rack. He needed a
     home run. As he stepped towards the plate, a sudden fluttering sound came from the
     dugout shadows, and a small, brown bat flew out erratically into the evening sky.
    The crowd gasped. The player gripped his bat tightly, focusing on the pitcher.

Question: What object did the baseball player grip tightly?

Answer:
"""
prompt = PromptTemplate(template=prompt_text, input_variables=[])

# Create an LLMChain
chain = LLMChain(llm=llm, prompt=prompt)

# Run the chain and print the result
try:
    response = chain.invoke({}) # No input variables needed for this template
    print("LLM Response:")
    print(response['text'])
except Exception as e:
    print(f"An error occurred: {e}")
    print("Please ensure your OPENAI_API_KEY is set correctly in your environment.")
```

Commentary on the Example and Results:

The Python code sets up a simple scenario using LangChain to query an LLM. The prompt describes a scene involving two distinct uses of the signifier "bat". The first instance clearly refers to the signified 'a piece of sporting equipment used in baseball' (*conceptual signified 1*). The second instance refers to the signified 'a nocturnal flying mammal' (*conceptual signified 2*). The question asks specifically what object the player gripped tightly. A human reader, accessing the distinct conceptual signifieds triggered by the context, immediately understands that the player gripped the baseball bat (signified 1).

 doi:10.20944/preprints202504.1830.v1

When this prompt is presented to a capable LLM like GPT-3.5-turbo or GPT-4, the typical response will correctly identify the object gripped as the baseball bat. For instance, a likely output would be:

```
LLM Response: The baseball player gripped his wooden bat tightly.
```

At first glance, this correct answer might suggest the LLM possesses a nuanced understanding equivalent to human conceptual differentiation. However, viewed through the Saussurean lens, the interpretation mechanism is crucially different. The LLM arrives at the correct answer not by accessing two distinct mental concepts associated with "bat", but by processing the statistical patterns of the *signifiers* in the provided context.

1. **Signifier Processing:** The model processes the entire sequence of tokens, including "baseball player," "wooden bat," "gripped his bat tightly," "flew out erratically," and "brown bat." 2. **Relational Value (Distributional Semantics):** The model's internal representations (embeddings) capture the strong statistical association (Saussurean *value*) between signifiers like "baseball player," "gripped," and "bat" (in the sense of equipment). This association is learned from trillions of tokens where these words co-occur. The context "wooden bat" further strengthens this association. 3. **Contextual Disambiguation via Signifiers:** The other signifiers, "flew out erratically," "dugout shadows," "evening sky," and "brown bat," are statistically associated with the animal meaning. The LLM uses the surrounding signifiers to determine the most probable referent for each instance of the token "bat" *within this specific linear sequence*. 4. **Pattern Matching for the Answer:** The question asks what was "gripped tightly." The model identifies the textual segment "The player gripped his bat tightly." This segment is situated immediately after the description of the baseball bat and is strongly associated with the player's actions. The model identifies this statistically dominant pattern and extracts the relevant object associated with "gripped" in that specific textual micro-context, which is the wooden (baseball) bat.

Therefore, even when the LLM provides the correct answer, it does so by expertly navigating the **system of signifiers** and their learned distributional relationships (values). It identifies the most statistically coherent interpretation based on the co-occurrence patterns of tokens in the input sequence. It does not possess two distinct, stable mental concepts (signifieds) for "bat" that it consciously selects between. Its success relies on the linguistic context providing sufficiently strong statistical cues (other signifiers) to differentiate the usage. A slightly different phrasing, a less common context, or a less capable model might fail if the statistical association of the more frequent signified (e.g., "bat" as equipment in a sports context) overrides the contextual cues pointing to the less frequent one. The potential for failure, and the mechanism of success (statistical pattern matching on signifiers), both underscore the LLM's operation at the level of Saussure's signifier, highlighting the absence of a robust, independent layer corresponding to the signified.

The Saussurean dyadic model thus provides critical insights into the nature of LLM capabilities and limitations. These models function as powerful computational engines for learning and manipulating the system (*langue*) of linguistic signifiers, capturing their linear structure and relational values with extraordinary fidelity. Their success demonstrates the power of modeling the formal, differential aspect of language structure. Yet, this analysis also starkly reveals their fundamental deficit: the missing signified. Operating solely on the plane of the signifier, LLMs remain detached from the conceptual realm that constitutes meaning for human communicators, thus perpetuating the semantic gap. This diagnosis sets the stage for considering Peirce's triadic semiotics, which introduces the concept of the object (referent) and offers a different, potentially richer perspective on the challenges of grounding and meaning in AI.

4. Semiotic Foundations II: The Peircean Sign and LLM Analysis

While Saussure's dyadic model effectively highlights the LLM's proficiency with the signifier system and its detachment from the conceptual signified, the semiotic framework developed by Charles Sanders Peirce offers a complementary and arguably more encompassing perspective for analyzing the relationship between signs, meaning, and reality, particularly concerning the challenges

 doi:10.20944/preprints202504.1830.v1

of grounding AI. Peirce's semiotics is fundamentally triadic, dynamic, and pragmatically oriented, focusing on the process of interpretation (**semiosis**) and the sign's connection to its object in the world [21].

Peirce defined a sign, or **Representamen**, as "something which stands to somebody for something in some respect or capacity" ([21] [CP 2.228]). This definition immediately establishes a three-part relationship. The Representamen is the sign vehicle itself—the perceptible form, such as a word, an image, a sound, smoke, or a symptom. It corresponds closely to Saussure's signifier but is conceived more broadly to include non-linguistic signs. In the context of LLMs, the textual tokens serve as the primary Representamens processed by the system.

Crucially, the Representamen stands for something else: its **Object**. Peirce made a vital distinction regarding the object:

- The **Immediate Object** is the Object as the sign itself represents it; the idea or schema of the object evoked directly by the sign. It is internal to the sign system in a sense, representing the object's characteristics as portrayed by the sign.
- The **Dynamic Object** is the actual object or state of affairs in reality that grounds the sign and the process of semiosis. It is the "really efficient but not immediately present object" ([21] [CP 8.343]) that determines the sign's representation. This Dynamic Object exists independently of any single interpretation, providing the ultimate referential anchor.

The third element of the Peircean triad is the **Interpretant**. This is not the human interpreter, but rather the effect, meaning, or understanding produced by the sign in the "mind or quasi-mind" of an interpreter ([21] [CP 4.536]). The Interpretant is itself another sign, often more developed than the initial Representamen, which clarifies or translates the original sign. Peirce further elaborated different types of interpretants:

- The **Immediate Interpretant** is the potential meaning or effect inherent in the sign itself, its 'interpretability' before any specific act of interpretation.
- The **Dynamic Interpretant** is the actual effect produced by the sign on a specific interpreter in a specific instance—a thought, feeling, or action.
- The **Final Interpretant** represents the ultimate, converged-upon meaning or habit of interpretation that would be reached after sufficient investigation and consideration by an ideal community of inquirers. It is the ideal outcome of the semiotic process.

This triadic structure is inherently dynamic. The process of **semiosis** involves a continuous chain where a Representamen refers to an Object, generating an Interpretant, which then often serves as a new Representamen referring to the same (or a slightly modified aspect of the) Object, generating a further Interpretant, and so on [51]. This potentially infinite process allows for the refinement of meaning and understanding through ongoing interpretation grounded, ultimately, in the Dynamic Object.

Peirce also classified signs based on the nature of the relationship between the Representamen and its Object:

- **Icons** function through similarity or resemblance (e.g., a portrait resembles its subject, a diagram reflects structural relations).
- **Indices** function through a direct physical, causal, or existential connection (e.g., smoke is an index of fire, a pointing finger is an index of location, pronouns like 'this' or 'here' are indices pointing to elements in the context).
- **Symbols** function through convention, habit, or learned rule (e.g., most words in a language, mathematical symbols, traffic lights). The connection is arbitrary, similar to Saussure's concept.

Analyzing LLMs through this Peircean lens reveals both capabilities and profound limitations. LLMs demonstrate exceptional ability in processing **Representamens**, particularly those functioning as

 doi:10.20944/preprints202504.1830.v1

Symbols. Their training on vast corpora allows them to internalize the complex conventions, rules, and statistical associations governing symbolic language use. They effectively learn the habits connecting symbolic Representamens within the textual domain, mirroring the conventional nature of symbolic signs.

However, the most significant deficit identified through the Peircean framework is the LLM's fundamental disconnection from the **Dynamic Object**. As discussed in Section 2, LLMs lack direct perceptual access to or interaction with the external world. Their "reality" is constituted solely by the patterns of Representamens (tokens) in their training data [8,9]. Consequently, the signs they manipulate are ungrounded; they lack the crucial link to the Dynamic Object that, in Peircean semiotics, determines the sign and anchors the entire process of semiosis. The symbols float freely, connected only to other symbols through statistical correlation, rather than being tied to the real-world entities, properties, or states of affairs they are meant to represent. This lack of grounding in the Dynamic Object is arguably the root cause of their inability to achieve genuine understanding, their susceptibility to factual errors, and their struggles with reasoning that requires knowledge beyond textual patterns [16].

The LLM's internal processing and output generation can be viewed as producing a form of computational **Dynamic Interpretant**. When presented with an input Representamen (a prompt), the model undergoes internal state changes (activations within the neural network) and produces an output Representamen (the generated text). This output is the actual effect produced by the input sign *on the LLM system*. However, this computational interpretant differs fundamentally from the Peircean concept in several ways. It is generated algorithmically based on learned statistical patterns among Representamens, not through a process grounded in the Dynamic Object. It lacks the potential richness of human interpretants (cognitive, emotional, dispositional effects) and, crucially, does not typically lead towards a Final Interpretant representing converged-upon truth or understanding about the Dynamic Object. Instead, it represents the statistically most plausible continuation of the symbolic sequence given the model's training.

Furthermore, the Peircean classification highlights LLMs' inherent difficulties with **Icons** and **Indices**, especially for text-only models. Icons rely on resemblance, a relationship difficult to capture purely through textual descriptions without access to the sensory modalities involved. While multimodal LLMs are improving Icon processing by correlating text with images [2,34], establishing deep conceptual similarity remains challenging. Indices pose an even greater problem, as they depend on a direct, contextual link to the Dynamic Object. Resolving an indexical like "put that there" requires knowing what "that" refers to and where "there" is in the current, specific situation (the Dynamic Object context). LLMs can often *mimic* the correct use of indices by learning common patterns (e.g., pronouns often refer to recently mentioned nouns), but they lack the direct connection to the specific situational context needed for robust, unambiguous resolution [6]. Their semiosis remains largely confined to the symbolic realm, detached from the indexical connections that ground human language in immediate reality.

To illustrate the LLM's disconnect from the Dynamic Object and its struggles with tracking changes in its state, consider this example involving simple object manipulation described textually:

```
import os
from langchain_openai import ChatOpenAI
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from dotenv import load_dotenv
import json # For structured tool output/input

# Ensure API key is loaded
load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")

# Initialize LLM
```

 doi:10.20944/preprints202504.1830.v1


```
llm = ChatOpenAI(temperature=0, model_name="gpt-4", openai_api_key=openai_api_key)
# Using a more capable model like GPT-4

# Define the prompt template describing a state change
prompt_text = """
Imagine a scene with a red box and a blue ball.
Initially, the red box is empty.
Then, someone puts the blue ball inside the red box.
After that, someone closes the lid of the red box.

Question: What color is the object currently inside the closed red box?

Answer:
"""
prompt = PromptTemplate(template=prompt_text, input_variables=[])

# Create and run the chain
chain = LLMChain(llm=llm, prompt=prompt)

try:
    response = chain.invoke({})
    print("LLM Response:")
    print(response['text'])
except Exception as e:
    print(f"An error occurred: {e}")
    print("Please ensure your OPENAI_API_KEY is set correctly.")

# --- Second prompt, slightly trickier reference ---
prompt_text_v2 = """
Imagine a scene with a red box and a blue ball.
Initially, the red box contains the blue ball and the lid is closed.
Then, someone opens the red box and takes the blue ball out.
They place the blue ball on a nearby table.
Finally, they close the empty red box again.

Question: Referring to the red box, is the object that was previously inside it now on
    the table?

Answer:
"""
prompt_v2 = PromptTemplate(template=prompt_text_v2, input_variables=[])
chain_v2 = LLMChain(llm=llm, prompt=prompt_v2)

try:
    response_v2 = chain_v2.invoke({})
    print("\nLLM Response V2:")
    print(response_v2['text'])
except Exception as e:
    print(f"An error occurred: {e}")
```

Commentary on the Example and Results:

The first prompt describes a simple sequence of actions affecting the state of two objects (Representamens: "red box", "blue ball"). The Dynamic Object, in this case, is the implied physical state of these items as described. Humans reading this easily track the state: the ball starts outside, goes inside, the box closes. The question asks about the color of the object inside the closed box. A capable LLM like GPT-4 typically answers correctly:

`LLM Response: The object currently inside the closed red box is blue (the blue ball).`

The LLM likely arrives at this by tracking the co-occurrence and sequential relationship of Representamens: "blue ball" + "inside" + "red box," and noting the final state description "closes the

 doi:10.20944/preprints202504.1830.v1

lid." The model correctly associates the property "blue" with the object described as being placed inside. This demonstrates competence in processing the symbolic description of the state change.

The second prompt (v2) introduces a slightly more complex scenario involving removal and relocation, and crucially, asks a question requiring tracking the identity of an object ("the object that was previously inside it") across state changes and locations. A human understands that "the object that was previously inside" refers specifically to the blue ball, which is now on the table. A likely correct answer is:

```
LLM Response V2: Yes, the object that was previously inside the red box (the blue
ball) is now on the table.
```

Again, a capable LLM might answer correctly. However, the Peircean analysis emphasizes *how* it likely achieves this. The LLM processes the sequence of Representamens: "red box contains the blue ball," "takes the blue ball out," "place the blue ball on a nearby table," "close the empty red box." It then processes the question containing the Representamen phrase "the object that was previously inside it" (where "it" refers indexically back to "red box"). The model correlates this phrase with the "blue ball" based on the initial state description and the "takes...out" action. It also correlates "blue ball" with "on a nearby table" from the subsequent description. It then synthesizes these correlations into a "Yes" answer, potentially adding the clarifying parenthesis based on patterns of helpful explanation.

The crucial point is that the LLM succeeds by manipulating the *Representamens* and their learned statistical relationships, effectively resolving the co-reference and state descriptions present *in the text*. It does not possess an internal model of the *Dynamic Object* (the actual implied box and ball whose state changes over time) that is independent of the textual description. Its understanding is mediated entirely through the sequence of signifiers. If the text describing the state changes were ambiguous, contradictory, or relied on unstated physical assumptions, the LLM's ability to track the state would likely degrade rapidly, as it lacks recourse to an underlying model of the physical reality (Dynamic Object) to resolve inconsistencies or fill in gaps. Its "interpretant" (the generated answer) is a function of the input Representamens, not an interpretation grounded in an independent representation of the situation the Representamens describe.

The Peircean framework thus reveals that while LLMs are powerful processors of symbolic Representamens, their operations remain largely ungrounded due to the missing link to the Dynamic Object. This disconnection limits their ability to handle indexicality robustly, prevents genuine situation awareness, and contributes to their fragility when dealing with information not explicitly encoded in familiar textual patterns. Building systems capable of true understanding, from this perspective, necessitates architectures that can explicitly model or interact with the Dynamic Object, allowing for semiosis to be grounded in something beyond the interrelations of symbols.

5. Synthesis: LLMs as Masters of Form, Disconnected from Meaning

Consolidating the analyses derived from both Saussurean structuralism and Peircean pragmatism provides a unified and compelling semiotic critique of Large Language Models. Despite their differing terminologies, focal points (internal system vs. interpretative process), and ontological commitments (dyadic vs. triadic), both frameworks converge on a central diagnosis: **current LLMs exhibit exceptional mastery over the formal, structural aspects of linguistic signs but remain fundamentally disconnected from the dimensions of conceptual meaning and real-world reference that constitute genuine understanding.** They operate predominantly on the plane of the signifier (Saussure) or the representamen (Peirce), manipulating symbols based on learned statistical patterns, while lacking robust internal correlates for the signified (Saussure) or the crucial links to the object and grounded interpretant (Peirce).

The Saussurean perspective (Section 3) illuminates how LLMs function as powerful computational instantiations of *langue*, the abstract system of linguistic conventions. They learn the intricate

 doi:10.20944/preprints202504.1830.v1

web of differential relationships between signifiers, capturing their sequential constraints (linearity) and relative values within the system through mechanisms like distributional embeddings [48,50]. Complementing this, the Peircean framework (Section 4) broadens the critique by emphasizing the dynamic process of semiosis and the crucial role of the object, showing LLMs' proficiency with symbolic Representamens but their critical detachment from the Dynamic Object that anchors meaning in reality [21].

Therefore, the synthesis of these perspectives paints a clear picture: LLMs are masters of linguistic form. They excel at learning and replicating the complex statistical patterns, syntactic structures, collocational frequencies, and distributional properties inherent in the vast sequences of signifiers/representamens they are trained on. Their ability to generate fluent, coherent, and often contextually appropriate text is a direct consequence of this mastery over the surface structure and statistical regularities of language use. They effectively model the conventional, rule-governed aspect of symbolic systems.

However, this mastery of form exists in parallel with a profound **core semiotic deficit**: the absence of robust connections to semantic content and referential grounding. Specifically, this deficit manifests in several key, interconnected ways:

- **Lack of Conceptual Representation (The Missing Signified):** LLMs do not possess internal structures that function equivalently to stable, abstract human concepts or the Saussurean signified. While their embedding vectors capture rich contextual and relational information about how tokens (signifiers) are used relative to each other, these representations remain fundamentally distributional rather than conceptual [19]. They do not encode abstract definitions, necessary and sufficient conditions, prototypical structures, or the rich network of causal and ontological relations that characterize human conceptual knowledge [36]. For instance, an LLM's representation of "bird" is derived from how the token "bird" co-occurs with tokens like "fly," "nest," "wings," etc., but it lacks an underlying abstract concept of 'birdness' that includes core properties (e.g., vertebrate, feathers, lays eggs) and allows for reasoning about atypical cases (e.g., penguins don't fly but are still birds) based on conceptual structure rather than just statistical association. This lack of true conceptual representation severely hinders capabilities requiring deep semantic understanding, such as resolving complex ambiguities that rely on conceptual distinctions rather than surface context, performing nuanced analogical reasoning that involves mapping conceptual structures, interpreting metaphors beyond common usage patterns, and achieving robust compositional generalization where meanings of complex expressions are built systematically from the meanings of their parts [44]. Much of human commonsense reasoning depends heavily on this rich substrate of conceptual knowledge, explaining LLMs' persistent fragility in this area [16].
- **Lack of Referential Grounding (The Missing Dynamic Object):** The symbols manipulated by LLMs are fundamentally ungrounded, lacking a systematic connection to the external world – the Peircean Dynamic Object – or even to perceptual experiences of it [7,8]. LLMs learn from vast datasets of text *describing* the world, but they do not learn *from* direct interaction with or perception of that world in the way humans do [35]. Their internal representations are thus correlations between signifiers, not representations anchored in non-linguistic reality. Even multimodal LLMs that process images or audio alongside text primarily learn cross-modal correlations rather than achieving deep conceptual grounding; the extent to which associating the signifier "cat" with images of cats imbues the model with the actual concept of a cat remains debatable [9,34]. This critical disconnection from the Dynamic Object is arguably the primary source of LLM unreliability. It prevents them from independently verifying factual claims against external reality, making them prone to generating confident falsehoods or "hallucinations" when statistical patterns deviate from factual states [10,11]. It limits their ability to adapt to novel real-world situations or contexts not adequately represented in their training data, as they lack a causal

 doi:10.20944/preprints202504.1830.v1

or physical model of the world to reason from. It also fundamentally restricts their capacity to resolve indexical expressions (like 'this', 'that', 'here', 'now') reliably, as these depend entirely on the specific, shared spatio-temporal context (the Dynamic Object) which the LLM cannot directly access [6]. Ultimately, the lack of referential grounding undermines the trustworthiness and applicability of LLMs in any scenario where connection to the actual state of the world is critical.

- **Ungrounded Interpretation (The Superficial Interpretant):** In the Peircean framework, the Interpretant represents the meaning-effect of a sign, ideally leading towards a more refined understanding or habit related to the Object. The outputs generated by LLMs, while superficially resembling interpretations or responses based on understanding, function differently. They are the result of an algorithmic process optimizing for the statistically most probable sequence of signifiers/representamens given the preceding sequence and the model's training data [5]. This computational effect lacks the cognitive depth and referential grounding of a Peircean interpretant. For instance, when an LLM generates a step-by-step explanation for a reasoning problem (e.g., via chain-of-thought prompting; [43]), it is often replicating common explanatory patterns found in its training data rather than performing genuine logical deduction based on an understanding of the concepts (signifieds) or the situation (object) involved. The resulting "interpretant" is a plausible linguistic form, not necessarily the outcome of a valid meaning-making process grounded in the problem itself. This superficiality limits the potential for LLMs to engage in truly meaningful dialogue, where participants build shared understanding by generating interpretants grounded in shared concepts and references. It also contributes to the perception of LLMs as "stochastic parrots" [18], systems capable of intricate linguistic mimicry but devoid of the underlying intentional and interpretative processes that characterize genuine communication and thought. Their generated interpretants do not reliably refine understanding or converge towards truth about the Dynamic Object because the entire process remains within the closed system of symbolic correlations.

The combined insights from Saussurean and Peircean semiotics therefore reveal that Large Language Models, despite their computational power and impressive linguistic fluency, operate primarily at a syntactic and statistical level, manipulating the formal aspects of signs with remarkable proficiency. They lack, however, the crucial semantic and referential dimensions that underpin genuine meaning and understanding in human communication. They function as masters of the signifier/representamen system but remain disconnected from the signified, the object, and the grounded interpretant. This fundamental limitation necessitates a paradigm shift in AI development, moving beyond models that merely process linguistic form towards architectures explicitly designed to integrate the missing components of the semiotic triangle or dyad. It is precisely this need that motivates the proposal for Large Semiosis Models (LSMs), which will be formally introduced and conceptualized in the following section as a pathway towards more meaningful and grounded artificial intelligence.

6. Introducing Large Semiosis Models (LSMs): Concept and Vision

The semiotic analysis presented in the preceding sections culminates in a clear imperative: transcending the limitations of current Large Language Models requires moving beyond architectures primarily focused on the statistical modeling of signifiers or representamens. The demonstrated mastery of linguistic form, while impressive, is insufficient for achieving robust understanding, reliable reasoning, and meaningful interaction, precisely because it operates detached from the conceptual and referential dimensions of meaning highlighted by both Saussurean and Peircean frameworks. To bridge this fundamental semantic gap, we introduce and formally conceptualize **Large Semiosis Models (LSMs)** as a next-generation paradigm for artificial intelligence.

 doi:10.20944/preprints202504.1830.v1

We define a Large Semiosis Model (LSM) as **an AI system architecturally designed to explicitly represent and integrate the core components and relational dynamics of semiotic processes, encompassing not only the sign vehicle (signifier/representamen) but also its connection to conceptual meaning (signified/interpretant) and its referential grounding in a representation of reality (object/world state).** Unlike LLMs, where semantic properties are at best emergent statistical correlates of manipulating ungrounded tokens, LSMs aim to incorporate structures and mechanisms specifically dedicated to modeling meaning, reference, and interpretation as integral parts of their operation. The ambition is to build systems where the processing of signs is intrinsically interwoven with representations of what those signs signify and refer to, thereby fostering a more holistic and grounded form of artificial intelligence.

The development and realization of LSMs should be guided by a set of core principles that directly address the semiotic deficits identified in LLMs. These principles, when realized, promise to unlock significantly more complex inferential capabilities than currently achievable with standard LLMs:

- **Explicit Semiotic Modeling:** At its heart, an LSM must possess an architecture that deliberately instantiates the relationships central to semiotic theory, rather than relying on these relationships to emerge implicitly from undifferentiated sequence modeling. This involves designing distinct computational components or representational layers corresponding to the Signifier/Representamen (e.g., token processing modules, possibly adapted LLM cores), the Signified/Interpretant (e.g., concept networks, dynamic meaning states), and the Object (e.g., world models, perceptual interfaces), along with explicit mechanisms governing their interaction. For example, instead of merely associating the token "cup" with "coffee" statistically, an LSM architecture might explicitly link the "cup" Representamen to a "Cup" Signified node in a conceptual graph, which in turn has defined relations like `contains(Liquid)`, `madeOf(Material)`, `affords(Grasping, Drinking)`, and crucially, links this concept to potential instances in a world model or perceptual input (the Object).
  - *Inferential Potential:* This explicit modeling allows for inferences impossible or unreliable for standard LLMs. Consider the statement: "The porcelain cup fell off the shelf." An LLM might predict the next word is "broke" based on statistical co-occurrence. An LSM, however, could access the "Cup" Signified, retrieve the `madeOf(Porcelain)` property, access knowledge associated with "Porcelain" (e.g., `isBrittle`), consult a basic physics model linked to the "Object" component (gravity causes falling, impact transfers energy), and *infer* that breaking is a highly probable outcome based on this chain of explicitly modeled semiotic and physical relationships, not just word statistics. This enables reasoning about novel situations (e.g., a cup made of a new, fictional material described as flexible) where statistical patterns are absent but conceptual properties and physical principles still apply.
- **Integrated Grounding Mechanisms:** LSMs must incorporate mechanisms that actively and continuously link symbolic representations to non-linguistic sources, thereby addressing the symbol grounding problem [7,8]. This grounding provides the necessary semantic anchor missing in LLMs. Key strategies include:
  - *Perceptual Grounding:* Direct integration with sensory data streams (vision, audio, etc.). Linguistic tokens like "red," "heavy," or "loud" would be dynamically associated with specific patterns of activation in perceptual processing modules analyzing real-time input. This moves beyond static text-image correlations. *Inferential Potential:* This enables understanding and reasoning about perceptually rich descriptions or instructions. An LLM struggles with "Pick up the slightly heavier of the two identical-looking cubes." An LSM with grounded perception (perhaps via simulated haptic feedback or visual analysis of micro-features correlated with weight) could potentially execute or reason about this task by linking the

 doi:10.20944/preprints202504.1830.v1

linguistic comparator "heavier" to actual perceived/simulated sensory data associated with the object referents.

- *Interactional/Embodied Grounding:* Learning through active engagement with a physical or simulated environment. Associating linguistic commands ("push the button") with motor actions and their perceived consequences (light turns on) creates strong bidirectional grounding [35]. *Inferential Potential:* This is crucial for robotics and tasks requiring understanding of affordances and causal effects of actions. An LLM might generate plausible instructions for assembling furniture but cannot adapt if the user makes a mistake or encounters an unexpected physical issue. An interactionally grounded LSM could potentially observe the deviation (via perception), update its world model (Object state), understand the physical implications, and provide corrective, contextually relevant instructions based on the actual, unfolding physical situation. It could learn intuitive physics concepts (e.g., stability, friction) through interaction, enabling predictions about physical scenarios not explicitly described in text.
- *Knowledge-Based Grounding:* Systematically linking linguistic entities to structured representations in knowledge graphs or ontologies [41]. This provides explicit conceptual structure and factual constraints. *Inferential Potential:* Enables disambiguation and reasoning based on established world knowledge. An LLM might confuse two historical figures named "John Smith" if mentioned closely. An LSM grounded in a knowledge graph could distinguish them by linking each mention to distinct nodes representing individuals with different properties (birth dates, professions, known associates), resolving the ambiguity using explicitly represented knowledge about the distinct "Objects" (the individuals). It could also perform complex multi-hop reasoning over the knowledge graph, guided by linguistic queries, that goes beyond simple fact retrieval.
- *Internal World Modeling:* Maintaining a dynamic, internal representation of the relevant environment, situation, or domain (the Object state), updated through perception, interaction, or inference [38]. *Inferential Potential:* This allows for tracking state changes over time and reasoning about hypothetical or future states. An LLM struggles to maintain consistency in long dialogues involving changing object properties or locations. An LSM with an internal world model could interpret statements like "The cat is now on the mat" by updating the `location` attribute of the `cat` entity in its internal model. It could then answer questions like "Where was the cat five minutes ago?" by potentially accessing historical states or reasoning backward, enabling much more coherent and stateful interaction and planning capabilities.

- **Rich Meaning Representation (Signified/Interpretant):** LSMs require internal representations that go beyond the distributional vectors of LLMs to capture the structure and nuances of concepts (Signified) and enable the generation of grounded, meaningful effects (Interpretants). Exploration areas include:
  - *Neuro-Symbolic Integration:* Combining neural networks' pattern learning with symbolic logic's explicit reasoning provides a pathway to represent both fuzzy statistical knowledge and crisp conceptual rules [52]. *Inferential Potential:* An LSM could use neural components to process perceptual input identifying objects and symbolic components to reason about their properties using formal logic (e.g., "If object X is fragile and force Y exceeds threshold Z, then object X breaks"). This allows for combining data-driven learning with verifiable, rule-based inference, crucial for safety-critical applications or tasks requiring explainable reasoning.
  - *Structured Concept Spaces:* Employing graph structures (learned or predefined) where nodes represent concepts and edges represent semantic relations (e.g., ISA, HASA, CAUSES). *Inferential Potential:* Facilitates deeper analogical reasoning (mapping relational structures between domains, e.g., understanding "an atom is like a solar system" by mapping the 'orbits'

 doi:10.20944/preprints202504.1830.v1

relation) and compositional semantics (deriving the meaning of "transparent aluminum cup" from the properties of its components and modification rules) in a more systematic way than LLM embeddings typically allow [44].

- *Causal and Physical Representations:* Incorporating modules that explicitly model causal relationships [45] or simplified physical dynamics. *Inferential Potential:* Enables genuine causal inference ("Why did the bridge collapse?") by consulting the causal model rather than just correlations, and allows for counterfactual reasoning ("What if the beam had been thicker?") by simulating alternatives based on the physical/causal model. This is critical for tasks like diagnosis, planning under uncertainty, and scientific hypothesis generation.
- *Dynamic Interpretant States:* Designing the model's core processing state to represent not just sequence context but a dynamic, grounded interpretation of the current situation, integrating linguistic input, world model state, and conceptual knowledge. *Inferential Potential:* This state would serve as the basis for more nuanced and contextually aware decision-making or generation. For example, the interpretation of the command "Be careful!" would depend heavily on this dynamic state – involving perception of hazards, understanding of potential consequences based on the world model, and accessing relevant concepts of danger – leading to contextually appropriate actions or responses far beyond simple textual association.

- **Integrated Processing and Semiosis Loop:** LSM architectures should facilitate cyclical information flow between components, enabling perception to inform interpretation, interpretation to guide action, and action to update the world state, which then influences subsequent perception and interpretation, mimicking Peircean semiosis. This contrasts with the typically linear, feed-forward processing of autoregressive LLMs. *Inferential Potential:* This continuous loop enables adaptive behavior, error correction, and learning from interaction in dynamic environments. Consider an LSM controlling a robot exploring a room. It perceives an object (Representamen: visual data), interprets it as a potential obstacle (Interpretant based on world model and concept of 'obstacle'), decides to navigate around it (Action), updates its internal map (Object state), and then perceives the new view (next Representamen), continuing the cycle. This tight integration allows for complex, goal-directed behavior and learning that adapts to unforeseen circumstances, far exceeding the capabilities of LLMs generating static plans based only on initial text prompts. It enables truly interactive learning and collaborative problem-solving [53].
- **Reference, Intentionality, and Transparency (Aspirational Goals):** While full realization is distant, the LSM principles aim towards these crucial aspects of intelligence. *Explicit reference tracking* becomes more feasible when symbols are grounded in world models or knowledge graphs, allowing the system (or an external observer) to verify what a symbol refers to. *Proto-intentionality* could emerge from goal-directed behavior driven by the interaction between grounded interpretations (Interpretants) and the world model (Object state), where actions are selected purposefully to achieve desired changes in the represented world. *Transparency* is enhanced because the modular nature, with explicit components for meaning, grounding, and world state, potentially allows for clearer debugging and explanation of the system's reasoning process [54] compared to the monolithic opacity of end-to-end LLMs. Achieving even partial progress towards verifiable reference and interpretable, goal-directed processing based on grounded representations would constitute a major advance in AI trustworthiness and capability.

The vision for LSMs is not merely an incremental improvement upon LLMs but a fundamental architectural and conceptual shift. By embracing principles derived from semiotics—explicit modeling of sign components, integrated grounding, rich meaning representation, and dynamic processing loops—LSMs aim to overcome the core limitations stemming from the semantic gap. The enhanced inferential capabilities described above illustrate the profound potential of this approach to move AI

 doi:10.20944/preprints202504.1830.v1

from manipulating form towards genuinely understanding and interacting with the world based on meaning and reference.

7. Potentialities and Applications of LSMs

The architectural principles and core design philosophy underpinning Large Semiosis Models—explicitly modeling semiotic components, integrating grounding mechanisms, employing rich meaning representations, and enabling dynamic processing loops—translate directly into a suite of profound potential capabilities that significantly exceed those of current LLMs. These potentialities promise not merely incremental improvements but qualitative leaps in AI performance, reliability, and applicability across diverse domains. By addressing the fundamental semantic gap identified through semiotic analysis, LSMs pave the way for AI systems that understand and interact with the world in a substantially more meaningful and robust manner.

- **Enhanced Semantic Understanding and Robust Reasoning:** Perhaps the most significant potentiality lies in achieving a deeper level of semantic comprehension. By linking signifiers/representamens to explicit conceptual representations (Signifieds) and grounding them in world models or perception (Objects), LSMs can move beyond surface-level pattern matching. This enables superior **ambiguity resolution**; for instance, the meaning of "bank" (river vs. financial institution) could be resolved not just by surrounding words but by consulting the LSM's internal world model or knowledge base to determine the relevant context (e.g., is the discussion about geography or finance?). Similarly, understanding **nuance, irony, and metaphor** often requires accessing conceptual knowledge and world understanding that LLMs lack. An LSM could potentially interpret "My lawyer is a shark" not just as a common idiom but by mapping relevant conceptual properties (predatory, aggressive) from the 'shark' concept onto the 'lawyer' concept within a specific interactional context. Crucially, LSMs promise far more **robust reasoning**. Equipped with internal world models, causal representations, and potentially integrated symbolic logic modules, they could perform reliable **commonsense reasoning** (e.g., inferring that putting water in a sieve results in it flowing through, based on object properties and physical principles), **causal inference** (e.g., diagnosing a system failure by tracing causal chains within its model, not just correlating symptoms described in text; [45,46]), and **counterfactual reasoning** ("What would have happened if the temperature had been lower?"). This contrasts sharply with LLMs, whose reasoning is often brittle, susceptible to superficial changes in phrasing, and struggles with novel scenarios not explicitly covered by training data patterns [6,14,15]. LSMs could reason *about* the world, not just *from* textual descriptions of it.
- **Improved Reliability, Trustworthiness, and Factuality:** The pervasive issue of hallucination in LLMs [10,11] stems directly from their lack of grounding. LSMs offer potential solutions by design. **Grounding mechanisms provide pathways for verification.** An LSM asked about the capital of France could potentially verify the answer ("Paris") by cross-referencing its internal knowledge base (knowledge-based grounding) or even, in principle, accessing external, trusted data sources linked to its Object representation layer. Perceptual grounding allows verification against observed reality; an embodied LSM describing a room could ground its statements in its actual visual input. Furthermore, the **internal world model** enables consistency checking. If an LSM asserts "Object A is inside Box B" and later asserts "Object A is on Table C" without describing a transfer action, its internal world model would register an inconsistency, potentially triggering clarification requests or self-correction, unlike LLMs which can easily contradict themselves across turns. This inherent capacity for verification and consistency maintenance promises significantly **higher factual reliability**. This is critical for deploying AI in high-stakes domains such as medicine (diagnostics, treatment information), finance (market analysis, advice), legal research, journalism, and education, where the cost of inaccurate or fabricated information generated by

 doi:10.20944/preprints202504.1830.v1

current LLMs is unacceptably high [55]. LSMs could become trustworthy sources of information and analysis precisely because their knowledge is structured, grounded, and internally consistent.

- **Meaningful Human-AI Interaction and Collaboration:** LSMs have the potential to revolutionize human-AI interaction, moving beyond the often stilted and error-prone exchanges with current chatbots. By maintaining a **grounded understanding of the shared context** (e.g., the state of a collaborative task, the physical environment in robotics, the concepts discussed), LSMs can engage in far more **natural and fluid dialogue**. The ability to robustly resolve **indexical references** ("put *this* block on *that* one," "referring to *her* earlier point") becomes feasible when symbols are linked to specific entities or locations within the shared Object representation (internal world model or perceived reality). This allows for seamless integration of language with non-linguistic cues like pointing gestures in embodied systems. LSMs could become true **collaborative partners**, capable of understanding user intentions not just from literal text but inferred from the shared context and task goals represented internally. They could proactively ask clarifying questions based on their world model ("Do you mean the large red block or the small one?"), offer relevant suggestions grounded in the current situation, and maintain coherent memory of the interaction history tied to evolving states of the world or task. This contrasts with the limitations of LLMs, which often struggle with long context windows, frequently misunderstand pragmatic intent, and lack the shared situational awareness necessary for deep collaboration [5]. LSMs could enable interfaces where language is used naturally to interact with complex systems or engage in joint problem-solving grounded in a shared reality.
- **Advanced Robotics and Embodied AI:** The realization of truly capable and adaptable robots operating in unstructured human environments hinges on abilities that LSMs are designed to provide. **Embodied grounding** through perception and interaction is non-negotiable for robotics. An LSM controlling a robot could connect linguistic commands like "Carefully pick up the fragile glass" directly to perceptual input (recognizing the object, assessing its likely fragility based on visual cues or prior knowledge linked to the 'glass' concept) and appropriate motor control strategies (adjusting grip force). It could understand complex spatial relations ("behind," "underneath," "between") by grounding these terms in its internal spatial representation derived from sensors. The **semiosis loop** (perception -> interpretation -> action -> updated world state -> perception) becomes the core operational cycle. This enables robots to **learn from interaction**, adapting their understanding of objects and their own capabilities based on experience (e.g., learning that a specific surface is slippery after detecting slippage). They could potentially handle novel objects by inferring affordances based on perceived shape, material (linked to grounded concepts), and basic physical reasoning. This vision far surpasses the capabilities of robots controlled by LLMs generating static plans from text; LSM-powered robots could dynamically adapt plans based on real-time perception and a continuously updated understanding of the physical world (the Dynamic Object), leading to more robust, flexible, and useful robotic assistants in homes, hospitals, and industries.
- **Accelerated Scientific Discovery and Complex Problem Solving:** The ability of LSMs to integrate diverse data types and reason based on grounded models holds immense potential for scientific research and complex problem-solving. LSMs could be designed to ingest and **fuse information from heterogeneous sources** – scientific literature (Representamens), experimental data (grounding in measurement Objects), simulation results (grounding in model Objects), and scientific imagery (perceptual grounding). By building internal models that represent not just statistical correlations but underlying mechanisms and causal relationships (grounded concepts and Object models), LSMs could assist researchers in **formulating novel hypotheses**. They could potentially design *in silico* experiments by manipulating their internal models or even suggest real-world experiments. Critically, they could **interpret complex experimental results** in the context of existing knowledge and their internal models, potentially identifying subtle patterns or

 doi:10.20944/preprints202504.1830.v1

inconsistencies missed by human researchers. Fields like drug discovery (modeling molecular interactions and predicting effects), materials science (designing novel materials with desired properties based on physical principles), climate modeling (integrating observational data and complex simulations), and systems biology (understanding intricate biological pathways) could benefit enormously from AI systems capable of this level of integrated, grounded reasoning [45]. Current LLMs can retrieve information from papers but struggle to synthesize it meaningfully or perform novel reasoning grounded in the underlying scientific principles or data.

- **Safer and More Interpretable AI:** While safety and interpretability remain profound challenges for any advanced AI, the architectural principles of LSMs offer potential advantages over monolithic LLMs. **Grounding provides constraints:** an AI whose actions are tied to a verifiable internal world model or knowledge base might be less likely to engage in undesirable behavior stemming from purely statistical extrapolation. Rules and constraints could potentially be encoded within the conceptual (Signified) or world model (Object) layers. Furthermore, the envisioned **modularity of LSMs**—with distinct components for processing signs, representing concepts, modeling the world, and handling grounding—could enhance **interpretability**. It might become possible to probe the system's internal world state, inspect the conceptual knowledge being activated, or trace how a particular interpretation (Interpretant) was derived from specific inputs and model components [54]. This contrasts with the often opaque, end-to-end nature of LLMs, where understanding the "reasoning" behind an output is notoriously difficult (the 'black box' problem). While not a complete solution, this potential for greater transparency could facilitate debugging, identifying failure modes, building trust, and ultimately contribute to developing AI systems that are more aligned with human values and intentions [16].

The potentialities unlocked by embracing the LSM paradigm are transformative. They span from achieving more human-like understanding and reasoning, enhancing reliability and trustworthiness, enabling seamless human-AI collaboration and advanced robotics, accelerating scientific progress, to potentially fostering safer and more interpretable AI systems. These prospects underscore the significance of pursuing LSMs, viewing the challenges not merely as technical hurdles but as necessary steps towards realizing a future generation of artificial intelligence capable of engaging with the world based on grounded meaning rather than just sophisticated pattern matching on symbolic forms.

8. Implementation Pathway I: A Saussurean-Inspired LSM

Having established the conceptual foundation and potential benefits of Large Semiosis Models, we now explore concrete implementation pathways. The first pathway draws inspiration directly from Ferdinand de Saussure's dyadic model of the sign, focusing specifically on addressing the critical limitation identified in Section 3: the LLM's proficiency with the Signifier coupled with its fundamental lack of the Signified. The primary goal of this Saussurean-inspired LSM is, therefore, to architecturally bridge the gap between the linguistic form (Signifier) and its associated concept (Signified), thereby enriching the model's processing with a layer of explicit conceptual meaning.

Conceptual Architecture

A Saussurean-inspired LSM would necessitate a modular architecture integrating distinct components responsible for handling the two faces of the Saussurean sign and the link between them:

1. **Signifier Processing Component (SPC):** This component would be responsible for processing the raw linguistic input and generating linguistic output, leveraging the proven strengths of existing LLM technology. It could be implemented using a pre-trained Transformer-based LLM (e.g., variants of GPT, Llama, Gemini) as its core engine. The SPC's primary function remains the modeling of sequential dependencies and distributional patterns among tokens (Signifiers), capturing the Saussurean *langue* as represented in text. However, unlike a standard LLM, its

 doi:10.20944/preprints202504.1830.v1

internal representations (e.g., token embeddings, hidden states) would be designed to interface directly with the Signified component.

2. **Signified Representation Component (SRC):** This component represents the core innovation of this pathway, designed to explicitly store and structure conceptual knowledge, embodying the Saussurean Signifieds. Several implementation strategies could be explored for the SRC, each with distinct advantages and disadvantages:
    - *Structured Knowledge Bases (KBs):* Utilizing large-scale, curated knowledge graphs like Wikidata, ConceptNet, Cyc, or domain-specific ontologies (e.g., SNOMED CT in medicine). In this approach, Signifieds are represented as nodes (entities, concepts) interconnected by labeled edges representing semantic relations (ISA, HASA, RelatedTo, AntonymOf, etc.). This offers high interpretability, explicit relational structure, and allows leveraging vast amounts of curated human knowledge. Challenges include coverage limitations (KBs are never complete), integration complexity with neural models, potential rigidity, and maintaining consistency.
    - *Learned Conceptual Embedding Spaces:* Creating a dedicated vector space where embeddings represent concepts (Signifieds) rather than just token distributions (Signifiers). These conceptual embeddings could be learned jointly during training or derived from structured KBs using graph embedding techniques (e.g., TransE, GraphSAGE). This approach offers the flexibility and scalability of neural representations but potentially sacrifices the explicit structure and interpretability of symbolic KBs. Defining the appropriate structure and learning objectives for such a space is a key research challenge.
    - *Hybrid Neuro-Symbolic Representations:* Combining elements of both approaches, perhaps using neural embeddings for concepts but constraining their relationships based on an ontological structure or logical rules [52]. This aims to balance flexibility with structure and explicit knowledge.
3. **Signifier-Signified Linking Mechanism (SSLM):** This is arguably the most crucial and technically challenging component. The SSLM must establish a dynamic, bidirectional bridge between the representations generated by the SPC (contextualized token embeddings/hidden states) and the representations within the SRC (concept nodes/embeddings). Its function is to map instances of Signifiers encountered in the input text to their corresponding Signified representations in the SRC, and conversely, to allow activated Signifieds in the SRC to influence the generation of Signifiers by the SPC. Potential mechanisms include:
    - *Cross-Attention Mechanisms:* Employing attention layers (similar to those within Transformers) that allow the SPC to "attend" to relevant concepts in the SRC when processing input tokens, and conversely, allow the SRC to attend to relevant textual context when activating concepts. This enables context-dependent mapping between signifiers and signifieds.
    - *Projection Layers and Shared Spaces:* Training projection layers to map SPC hidden states into the SRC's conceptual space (or vice-versa), potentially enforcing alignment through contrastive learning objectives that pull representations of related signifiers and signifieds closer together.
    - *Multi-Task Learning Frameworks:* Training the entire system jointly on multiple objectives, including standard language modeling (predicting the next Signifier) and auxiliary tasks like predicting the correct Signified concept(s) associated with input tokens/phrases or generating text conditioned on specific input Signifieds from the SRC.
    - *Explicit Entity Linking / Concept Mapping Modules:* Incorporating dedicated modules, potentially based on traditional NLP techniques or specialized neural networks, designed specifically to perform entity linking or concept mapping between mentions in the text (Signifiers) and entries in the SRC (Signifieds).

 doi:10.20944/preprints202504.1830.v1

Training Methodologies

Training a Saussurean-inspired LSM would require going beyond standard self-supervised pre-training on raw text. It would likely involve a multi-stage process or multi-task learning objectives incorporating data that explicitly links linguistic forms to concepts:

- **Joint Training:** Training the SPC and SRC components simultaneously, possibly with a shared objective function that includes both language modeling loss (predicting Signifiers) and a concept alignment loss (e.g., maximizing similarity between SPC representations of a word and SRC representation of its linked concept, using datasets annotated with word-concept links like WordNet or KB entity links).
- **Pre-training followed by Alignment:** Pre-training the SPC (LLM core) and potentially pre-training the SRC (e.g., graph embeddings on a KB) separately, followed by a dedicated alignment phase focusing on learning the SSLM parameters using aligned text-concept data.
- **Data Requirements:** Training would benefit significantly from large datasets where textual mentions are explicitly linked to concepts in a target knowledge base or ontology. Resources like Wikipedia (with its internal links and connections to Wikidata), annotated corpora, or large-scale dictionaries/thesauri could be leveraged. Generating pseudo-aligned data might also be explored.
- **Concept-Conditioned Generation:** Training the model not only to predict text but also to generate text that accurately reflects or elaborates on specific input concepts provided from the SRC. This forces the model to learn the mapping from Signified back to appropriate Signifiers.

Operational Flow

During inference, the interaction between components would enable richer processing:

1. **Input Processing:** Input text (sequence of Signifiers) is processed by the SPC (LLM core).
2. **Conceptual Activation:** As the SPC processes the input, the SSLM dynamically identifies key Signifiers and maps them to corresponding Signified representations within the SRC. This might involve activating specific nodes in a KB or producing context-dependent conceptual embeddings. The strength or relevance of activated concepts could be modulated by the textual context processed by the SPC.
3. **Conceptual Reasoning/Enrichment (Optional):** Depending on the sophistication of the SRC, limited reasoning might occur within the conceptual layer itself (e.g., traversing relations in a KB to infer related concepts or properties not explicitly mentioned in the text). Activated concepts might also retrieve associated factual knowledge.
4. **Informed Generation:** The activated conceptual representations in the SRC, along with the processed textual context from the SPC, jointly inform the generation of the output text. The SPC's generation process would be modulated or constrained by the active Signifieds, ensuring the output is not only linguistically plausible but also conceptually coherent and potentially factually consistent with the activated knowledge. For instance, when generating text about a "penguin," the activated "Penguin" concept in the SRC (linked to properties like `ISA(Bird)`, `Habitat(Antarctic)`, `Ability(Swim)`, `Cannot(Fly)`) could guide the SPC to produce relevant and accurate statements, potentially overriding purely statistical textual patterns that might incorrectly associate penguins with flying due to the general association between "bird" and "fly."

Limitations of This Pathway

While this Saussurean-inspired pathway directly addresses the "missing Signified" problem, significantly enhancing conceptual understanding and potentially reducing certain types of hallucinations related to conceptual confusion or lack of factual knowledge accessible via the SRC, it inherently inherits the limitations of Saussure's focus on the internal linguistic system.

 doi:10.20944/preprints202504.1830.v1

- **Indirect Grounding:** The grounding achieved is primarily *conceptual* rather than *referential* or *perceptual*. The Signifieds within the SRC, especially if based on KBs or learned purely from text-aligned data, might themselves lack direct connection to the Peircean Dynamic Object (real-world referents or sensory experience). The system understands the *concept* of "red" as defined within its SRC (e.g., related to 'color,' opposed to 'blue'), but it doesn't *perceive* red in the world.
- **Potential Brittleness of KBs:** If relying heavily on curated KBs for the SRC, the system's performance can be limited by the coverage, accuracy, and consistency of the KB itself. Integrating and updating massive KBs also poses significant engineering challenges.
- **Abstractness:** This approach is better suited for tasks involving abstract knowledge, factual recall, and conceptual reasoning than for tasks requiring direct interaction with a dynamic, physical environment or understanding based on real-time perception.

The Saussurean-inspired LSM pathway thus offers a concrete strategy for augmenting LLMs with an explicit layer of conceptual meaning (Signifieds), leveraging structured knowledge or learned concept representations. Architecturally bridging the Signifier-Signified gap promises enhanced conceptual coherence, improved factual consistency (if the SRC is reliable), and more nuanced semantic understanding compared to standard LLMs operating solely on Signifiers. This approach represents only a partial step towards fully grounded AI, however, as it primarily addresses conceptual meaning within a largely symbolic system, leaving the challenge of direct grounding in the external world (the Dynamic Object) largely unaddressed. This limitation motivates the exploration of the Peircean-inspired pathway discussed next.

9. Implementation Pathway II: A Peircean-Inspired LSM

While the Saussurean-inspired pathway offers a route to enriching LLMs with explicit conceptual meaning (Signifieds), it falls short of addressing the fundamental grounding problem exposed most clearly by Charles Sanders Peirce's semiotics. The second implementation pathway, therefore, takes direct inspiration from Peirce's triadic model (Representamen-Object-Interpretant) and his emphasis on the dynamic process of semiosis. This approach is inherently more ambitious, aiming to build systems that connect signs not just to internal concepts but to representations of, or direct interactions with, the external reality (the Dynamic Object), thereby enabling a truly grounded form of interpretation and action.

Conceptual Architecture

A Peircean-inspired LSM necessitates a significantly more complex and integrated architecture, explicitly designed to handle the triad and the flow of semiosis, often implying multimodality and embodiment (virtual or physical):

1. **Representamen Processing Component (RPC):** Analogous to the SPC in the Saussurean pathway, this component handles the processing of sign vehicles. Given the Peircean emphasis on diverse sign types, the RPC must be inherently **multimodal**. It would likely incorporate:
    - A powerful language processing core (e.g., an LLM architecture) for handling symbolic Representamens (text, speech).
    - Sophisticated perception modules (e.g., Computer Vision networks like CNNs or Vision Transformers, audio processing networks) capable of extracting features and representations from sensory data, which also function as Representamens (specifically Icons and Indices derived from perception).
    - Mechanisms for fusing information from different modalities, potentially using cross-attention or dedicated fusion layers. The RPC's output would be rich, contextualized representations of the incoming signs, irrespective of their modality.

 doi:10.20944/preprints202504.1830.v1

2. **Object Representation Component (ORC):** This component is the cornerstone of the Peircean pathway, tasked with representing the Dynamic Object – the external reality or situation the signs refer to. This is the primary locus of grounding. Implementing the ORC is a major research challenge, likely involving a combination of approaches:
    - *Direct Perceptual Interface:* Direct connections to sensors (cameras, microphones, LIDAR, tactile sensors) providing real-time data about the immediate environment. This data stream *is* the most direct representation of the Dynamic Object available to the system at any given moment.
    - *Simulation Engine Integration:* For virtual agents, tight integration with a high-fidelity physics or environment simulator (e.g., Unity, Unreal Engine, Isaac Sim, MuJoCo). The state of the simulation serves as the Dynamic Object representation, allowing the LSM to interact with and observe the consequences of actions within a controlled, dynamic world.
    - *Internal World Model:* A dynamically maintained internal representation of the environment's state, entities, properties, and relations, constructed and updated based on perceptual input, interaction history, and inferential processes. This model might employ various formalisms:
        - *Scene Graphs:* Representing objects, their properties (color, shape, location, pose), and spatial relationships in a structured graph format.
        - *Factor Graphs or Probabilistic Models:* Representing uncertain knowledge about the world state and enabling probabilistic inference.
        - *State Vector Representations:* Using embedding vectors to capture the state of objects or the entire environment, potentially learned end-to-end.
        - *Knowledge Graphs (Dynamic):* Employing knowledge graphs where nodes represent specific object instances and properties, with mechanisms for updating these based on new information or actions.

    The ORC must be dynamic, capable of reflecting changes in the external world or simulation over time, either through direct updates from perception/simulation or through internal predictive modeling based on actions and physical laws.
3. **Interpretant Generation Component (IGC):** This component is responsible for producing the meaning-effect (Interpretant) of a sign (Representamen) in the context of the current Object state as represented by the ORC. The Interpretant is not merely a textual output but represents the system's situated understanding or disposition to act. Implementing the IGC might involve:
    - *Dynamic System State:* Treating the overall activation state of the LSM's internal network, influenced by RPC input and ORC context, as the substrate of the Dynamic Interpretant. Specific patterns of activation would correspond to different interpretations.
    - *Goal Representation Modules:** Translating interpreted inputs (e.g., linguistic commands, perceived goal states) into explicit internal goal representations that drive subsequent behavior.
    - *Grounded Reasoning Modules:* Incorporating modules capable of performing inference directly over the representations within the ORC (world model) and integrated conceptual knowledge (potentially a sophisticated SRC from the Saussurean pathway, now grounded via the ORC). This could include causal reasoning engines [45], spatial reasoners, or physical simulators.
    - *Planning and Policy Modules:* Utilizing techniques from AI planning or reinforcement learning to generate sequences of actions (motor commands, linguistic utterances) aimed at achieving goals derived from the Interpretant, considering the constraints and dynamics represented in the ORC.

 doi:10.20944/preprints202504.1830.v1

4. **Semiosis Loop and Action Execution:** The architecture must explicitly implement the cyclical flow of Peircean semiosis.
    - *Interpretation:* The IGC generates an Interpretant based on input from the RPC (Representamen) and the current state of the ORC (Object).
    - *Action Selection:* Based on the Interpretant (e.g., derived goal, situational assessment), an action is selected or generated (this could be a linguistic output generated via the RPC, or a motor command sent to actuators/simulation).
    - *Execution & Effect:* The action is executed in the environment (real or simulated).
    - *Feedback/Observation:* The consequences of the action on the Dynamic Object are observed via the perceptual interface or simulation feedback, leading to updates in the ORC.
    - *Iteration:* This updated Object state then informs the interpretation of subsequent Representamens entering the RPC, closing the loop and allowing for continuous adaptation and learning.

Training Methodologies

Training a Peircean-inspired LSM shifts the focus heavily towards **experiential and interactive learning**, moving significantly beyond static text pre-training:

- **Reinforcement Learning (RL):** RL is a natural fit for training the semiosis loop. The LSM agent receives rewards based on achieving goals, accurately predicting environment dynamics, successfully interpreting commands, or maintaining consistency between its internal model and observed reality. This allows the system to learn optimal policies for interpretation and action through trial-and-error interaction with its environment [53,56]. Intrinsic motivation objectives (e.g., curiosity-driven exploration) might also be crucial for learning rich world models.
- **Imitation Learning / Behavioral Cloning:** Training the LSM to mimic successful behaviors demonstrated by humans or expert systems within the target environment. This can bootstrap the learning process, especially for complex motor skills or interaction protocols.
- **Self-Supervised Learning on Interaction Data:** Leveraging the vast amount of data generated during interaction for self-supervised objectives. Examples include predicting future perceptual states given current state and action, learning inverse dynamics (predicting action needed to reach a state), ensuring cross-modal consistency (predicting textual description from visual input, or vice-versa), or learning affordances by observing interaction outcomes.
- **Curriculum Learning:** Gradually increasing the complexity of the environment, the tasks, or the required semiotic reasoning to facilitate stable learning, starting with simple object interactions and progressively moving towards more complex scenarios.
- **Leveraging Pre-trained Components:** While interaction is key, pre-trained models (e.g., vision encoders, LLM cores for the RPC) can provide valuable initial representations and accelerate learning, followed by extensive fine-tuning within the interactive loop.

Operational Flow

The operation of a Peircean LSM in a typical scenario (e.g., a robot responding to a command) would involve:

1. **Multimodal Input:** Receives input via RPC (e.g., spoken command "Bring me the red apple from the table" + visual input of the scene).
2. **Grounded Perception & Object Update:** RPC processes sensory data; ORC updates its internal world model (identifies objects like 'apple', 'table', notes their properties like 'red', location).
3. **Representamen Grounding:** Links linguistic Representamens ("red apple," "table") to specific entities identified within the ORC's world model. Resolves indexicals based on perception and world state.

 doi:10.20944/preprints202504.1830.v1

4. **Interpretant Generation:** IGC interprets the command in the context of the grounded world state, generating an internal goal (e.g., `Goal: Possess(Agent, Object_Apple1) where Apple1.color=red AND Apple1.location=Table2`).
5. **Planning/Action Selection:** A planning module uses the goal and the ORC's model (including physical constraints, affordances) to generate a sequence of motor actions (navigate to table, identify correct apple, grasp, return).
6. **Execution & Monitoring:** Executes actions, continuously monitoring perceptual feedback via RPC/ORC to detect errors, confirm success, or adapt the plan dynamically if the world state changes unexpectedly (e.g., apple rolls).
7. **Goal Completion & Reporting:** Upon successful execution, updates the ORC (agent now possesses apple) and potentially generates a linguistic Representamen via RPC ("Here is the red apple").

Advantages and Challenges

The Peircean pathway directly confronts the symbol grounding problem by integrating perception, action, and world modeling into the core architecture. Its key advantage is the potential for **truly grounded understanding and robust interaction with dynamic environments**. It inherently supports **multimodality** and **embodiment**, crucial for robotics and situated AI. The dynamic semiosis loop enables **continuous learning and adaptation** based on experience.

However, the challenges are immense:

- **Complexity:** The architecture is significantly more complex than standard LLMs or even the Saussurean pathway, requiring tight integration of perception, language, reasoning, world modeling, and action components.
- **World Modeling:** Building accurate, scalable, and efficiently updatable world models (ORC) remains a major AI research frontier.
- **Perception:** Robust real-world perception is still challenging, susceptible to noise, occlusion, and ambiguity.
- **Training:** Training via RL in complex environments is notoriously data-hungry, computationally expensive, and can suffer from instability and exploration problems. Defining appropriate reward functions is difficult.
- **Simulation Gap:** Transferring models trained in simulation to the real world often faces challenges due to discrepancies between simulated and real physics/sensor data.
- **Evaluation:** Defining metrics to evaluate "grounded understanding" or the quality of the internal Interpretant remains difficult.

The Peircean-inspired LSM pathway therefore represents a long-term, ambitious vision for AI. Tackling grounding head-on through the explicit modeling of the Representamen-Object-Interpretant triad and the interactive semiosis loop promises AI systems with unprecedented capabilities for understanding, reasoning about, and acting within the real world. While the implementation hurdles are substantial, pursuing this direction appears essential for moving beyond the limitations of current form-focused AI towards systems possessing genuine semantic intelligence.

10. Practical Example: Adapting LangChain for Saussurean Semiosis (Python Sketch)

To concretize the architectural concepts outlined for the Saussurean-inspired LSM (Section 8), this section presents a practical, albeit simplified, implementation sketch using Python and the LangChain framework [22]. LangChain provides tools for composing applications involving LLMs, making it suitable for demonstrating how components approximating the Signifier Processing Component (SPC), Signified Representation Component (SRC), and Signifier-Signified Linking Mechanism (SSLM)

 doi:10.20944/preprints202504.1830.v1

might interact. It is crucial to emphasize that this example is a conceptual illustration intended to highlight the *principle* of integrating explicit conceptual knowledge (approximating the Signified) with LLM-based signifier processing, rather than a fully realized LSM architecture.

Goal and Scenario

The goal is to demonstrate a system where an LLM's response generation is explicitly informed by structured information retrieved from a knowledge source, representing the link between a textual Signifier and its associated Signified concept's properties. We will simulate a scenario where a user asks for information about a specific entity (e.g., the "Quokka"), and the system retrieves predefined facts about this entity from a mock knowledge base (our SRC proxy) before synthesizing the final answer using the LLM (our SPC).

Implementation Using LangChain

```python
import os
from langchain_openai import ChatOpenAI
from langchain_core.prompts import PromptTemplate
from langchain_core.runnables import RunnablePassthrough, RunnableLambda
from langchain_core.output_parsers import StrOutputParser
from dotenv import load_dotenv

# --- Setup ---
# Load environment variables (ensure OPENAI_API_KEY is set)
load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")

if not openai_api_key:
    print("Error: OPENAI_API_KEY not found in environment variables.")
    # Handle error appropriately, e.g., exit or raise exception
    exit()

# --- Component Approximations ---

# 1. Signifier Processing Component (SPC) Proxy: Initialize the LLM
# We use ChatOpenAI, representing the core engine for processing text (Signifiers)
try:
    llm = ChatOpenAI(temperature=0.1, model_name="gpt-4", openai_api_key=openai_api_key
    )
    # Using gpt-4 for better synthesis, but gpt-3.5-turbo could also work
except Exception as e:
    print(f"Error initializing LLM: {e}")
    exit()

# 2. Signified Representation Component (SRC) Proxy: A Simple Knowledge Base
# This dictionary simulates a KB holding structured info about concepts (Signifieds)
# Keys are simplified Signifiers, values represent properties of the Signified concept
knowledge_base = {
    "quokka": {
        "type": "Small macropod marsupial",
        "habitat": "Small islands off the coast of Western Australia, primarily
    Rottnest Island",
        "diet": "Herbivorous (leaves, stems, bark)",
        "conservation_status": "Vulnerable",
        "notable_feature": "Often described as 'the world's happiest animal' due to its
     seemingly smiling expression."
    },
    "axolotl": {
        "type": "Neotenic salamander (paedomorphic)",
        "habitat": "Lake Xochimilco complex near Mexico City",
```

 doi:10.20944/preprints202504.1830.v1


```
        "diet": "Carnivorous (worms, insects, small fish)",
        "conservation_status": "Critically Endangered",
        "notable_feature": "Extraordinary regenerative abilities; retains larval features throughout life."
    },
    # Add more entities as needed
}

# 3. Signifier-Signified Linking Mechanism (SSLM) Proxy: KB Query Function
# This function simulates the SSLM retrieving Signified data based on an identified Signifier
# It performs a simple lookup in our SRC proxy.
# A real SSLM would need sophisticated entity recognition and linking.
def query_knowledge_base(entity_name: str) -> dict:
    """
    Looks up an entity in the knowledge base (SRC proxy).
    Returns the dictionary of properties (Signified info) or a 'not found' message.
    """
    # Simple case-insensitive lookup for this example
    entity_key = entity_name.lower().strip().replace("the ", "") # Basic normalization
    print(f"[SSLM Simulation] Querying SRC for Signifier: '{entity_key}'")
    result = knowledge_base.get(entity_key, {"error": f"Information on '{entity_name}' not found in the knowledge base."})
    print(f"[SSLM Simulation] Retrieved Signified Data: {result}")
    return result

# --- LSM Workflow Simulation using LangChain Expression Language (LCEL) ---

# Define the prompt template for the final synthesis step (informed generation)
# This prompt instructs the SPC (LLM) to use both the original query (containing the initial Signifier)
# and the retrieved SRC context (Signified information)
synthesize_prompt_template = PromptTemplate.from_template(
    """Your task is to answer the user's question based *primarily* on the provided knowledge base context.
Use the context to formulate a comprehensive and accurate answer.
If the context indicates the information is not found, state that clearly.
Do not add information not present in the context unless specifically asked to elaborate generally beyond it.

Original Question (contains target Signifier):
{original_question}

Knowledge Base Context (retrieved Signified properties):
{signified_context}

Synthesized Answer:
"""
)

# Define the processing chain using LCEL
# This chain orchestrates the interaction between SPC, SRC, and SSLM proxies
chain = (
    # Pass the original question through using a dictionary structure
    {"original_question": RunnablePassthrough()}
    # Identify the key Signifier from the question (simplistic extraction for demo)
    # In a real system, this might be a separate NLP step or sophisticated prompting
    | RunnablePassthrough.assign(entity=lambda x: x["original_question"].lower().split("about the ")[-1].split("?")[0].strip())
    # Invoke the SSLM proxy (query_knowledge_base) using the extracted entity Signifier
    # to get the Signified context from the SRC proxy
```

 doi:10.20944/preprints202504.1830.v1


```
    | RunnablePassthrough.assign(signified_context=RunnableLambda(lambda x:
    query_knowledge_base(x['entity'])))
    # Prepare the input for the final synthesis prompt
    | (lambda x: {"original_question": x["original_question"], "signified_context": x["
    signified_context"]})
    # Format the prompt using the original question and the retrieved Signified context
    | synthesize_prompt_template
    # Invoke the SPC proxy (LLM) for the final synthesis
    | llm
    # Parse the LLM's output string
    | StrOutputParser()
)

# --- Running the Example ---

# Input query containing the Signifier "Quokka"
user_query = "Tell me about the Quokka?"

print(f"User Query: {user_query}\n")
print("--- Running Saussurean LSM Simulation Chain ---")

# Invoke the chain
try:
    final_response = chain.invoke(user_query)
    print("\n--- Final Synthesized Response ---")
    print(final_response)
except Exception as e:
    print(f"\nAn error occurred during chain execution: {e}")

# Example with an entity not in the KB
user_query_unknown = "What can you tell me about the Blobfish?"
print(f"\nUser Query: {user_query_unknown}\n")
print("--- Running Saussurean LSM Simulation Chain (Unknown Entity) ---")
try:
    final_response_unknown = chain.invoke(user_query_unknown)
    print("\n--- Final Synthesized Response ---")
    print(final_response_unknown)
except Exception as e:
    print(f"\nAn error occurred during chain execution: {e}")
```

Commentary and Semiotic Interpretation:

1. **Code Explanation:**
   - The `ChatOpenAI` instance (`llm`) acts as the **SPC**, responsible for processing the final prompt and generating the textual response.
   - The `knowledge_base` dictionary serves as a simplified proxy for the **SRC**, storing structured data representing the properties associated with specific concepts (Signifieds like 'Quokka' or 'Axolotl').
   - The `query_knowledge_base` function, wrapped in `RunnableLambda`, acts as the core of the **SSLM** simulation. It takes an identified entity name (a potential Signifier extracted from the user query) and retrieves the corresponding structured information (Signified properties) from the SRC proxy.
   - The LangChain Expression Language (LCEL) sequence orchestrates the workflow. It first extracts a simplified entity/Signifier ("quokka") from the input question. Then, it calls the SSLM proxy (`query_knowledge_base`) to fetch the relevant Signified data from the SRC proxy. Finally, it formats the `synthesize_prompt_template` with both the original question and the retrieved Signified context, feeding this combined information to the SPC (`llm`) for the final answer generation. The `StrOutputParser` ensures a clean text output.

 doi:10.20944/preprints202504.1830.v1

2. **Semiotic Interpretation:**
    - *Signifier Input:* The user query ("Tell me about the Quokka?") provides the initial Signifier ("Quokka").
    - *SSLM Action:* The chain structure, specifically the `RunnableLambda` calling `query_knowledge_base` after entity extraction, simulates the SSLM linking the identified Signifier ("quokka") to its corresponding entry in the SRC proxy.
    - *SRC Activation:* The retrieval of the dictionary `{"type": "Small macropod...", ...}` represents the activation or fetching of the structured information associated with the **Signified** concept 'Quokka' within our SRC proxy. This retrieved data embodies the conceptual properties Saussure associated with the signified, albeit in a simple, structured format here.
    - *Informed SPC Generation:* The final LLM call is crucial. Unlike a standard LLM which would answer based solely on its internal, distributional knowledge of the Signifier "Quokka" (learned from its vast training data, potentially containing inaccuracies or missing details), this setup *forces* the SPC (`llm`) to use the explicitly provided Signified context from the SRC proxy. The prompt directs it to base the answer *primarily* on this context. The LLM's task shifts from pure pattern completion based on signifier statistics to synthesizing a response grounded in the provided conceptual (Signified) information. This demonstrates the core principle of the Saussurean-inspired LSM: leveraging an explicit conceptual layer to inform and constrain language generation.
    - *Handling Absence:* When queried about "Blobfish" (assuming it's not in `knowledge_base`), the SSLM proxy returns an error message. The final prompt includes this error message as the context. The SPC, guided by the prompt, correctly reports that the information was not found in the knowledge base, demonstrating graceful handling when a Signifier cannot be linked to a known Signified within the SRC. A standard LLM might hallucinate information about a Blobfish based on its general training data.
    - *Contrast with Standard LLM:* A direct query to a standard LLM about "Quokka" might yield a similar-sounding answer, derived from its vast internal distributional knowledge (*langue*). However, that knowledge lacks the explicit structure and verifiability of our SRC proxy. The LLM's answer could be subtly inaccurate, incomplete, or even hallucinated, reflecting the statistical nature of its knowledge about the Signifier rather than access to a defined Signified concept. The LSM sketch, by incorporating the SRC lookup, introduces a layer of explicit conceptual grounding (within the limits of the SRC's accuracy and coverage), making the answer generation process more constrained and potentially more reliable regarding the specific facts stored in the SRC.
3. **Limitations of the Sketch:** This example significantly simplifies the components of a true Saussurean LSM. The SRC is merely a small dictionary, not a vast, dynamic knowledge graph or learned concept space. The SSLM is rudimentary, relying on simple string matching for entity linking, whereas a real system would need sophisticated Named Entity Recognition (NER) and Entity Disambiguation. The integration via prompting is a common technique but differs from deeper architectural fusion involving cross-attention or shared embedding spaces as envisioned in Section 8.

Even with these significant simplifications, this example effectively illustrates the core *architectural intent* of a Saussurean-inspired LSM: augmenting the LLM's signifier processing (SPC) with explicit, structured information representing conceptual meaning (SRC), thereby enriching the generation process. It highlights a pathway towards systems that explicitly bridge the Signifier-Signified gap identified by Saussure, constraining language generation with conceptual knowledge distinct from purely distributional patterns. While this approach focuses on conceptual grounding within a largely symbolic system and does not directly address the grounding in external reality central to the Peircean

 doi:10.20944/preprints202504.1830.v1

pathway, it demonstrates how current tools like LangChain can serve as valuable platforms for prototyping and exploring components of LSMs designed to imbue AI with a more robust layer of semantic understanding tied to defined concepts.

11. Practical Example: Adapting LangChain for Peircean Semiosis (Python Sketch)

Moving beyond the conceptual grounding of the Saussurean pathway, this section provides a practical implementation sketch illustrating principles of the more ambitious Peircean-inspired LSM (Section 9). The core objective here is to demonstrate, using LangChain, how an AI system can interpret linguistic signs (Representamens) by referencing and interacting with an explicit, albeit simulated, representation of a world state (the Object Representation Component or ORC proxy), thereby grounding its interpretation (Interpretant) and subsequent actions. This example focuses on simulating the dynamic semiosis loop: input sign -> object state consultation -> interpretation/planning -> action affecting object state -> feedback. Again, this is a simplified illustration aiming to capture the essence of the Peircean triad interaction, not a complete LSM implementation.

Goal and Scenario

The goal is to simulate an agent that can understand and execute simple commands involving manipulating objects within a defined, dynamic environment. The scenario involves a virtual "room" containing a few objects with specific properties (color, location). The agent receives a textual command (Representamen), checks the current state of the room (Object state), decides on an action based on its interpretation (Interpretant), executes the action modifying the room state (affecting the Dynamic Object proxy), and reports the outcome.

Implementation Using LangChain Agents and Custom Tools

```python
import os
from langchain_openai import ChatOpenAI
from langchain.agents import Tool, AgentExecutor, create_react_agent
from langchain import hub # To pull the react prompt
from dotenv import load_dotenv
import json # For structured tool output/input

# --- Setup ---
# Load environment variables (ensure OPENAI_API_KEY is set)
load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")

if not openai_api_key:
    print("Error: OPENAI_API_KEY not found in environment variables.")
    exit()

# --- Component Approximations ---

# 1. Representamen Processing & Interpretant Generation Core (RPC/IGC Proxy):
# The LLM serves as the core reasoning engine within the agent.
# It processes the input text (Representamen), interacts with tools (accessing ORC),
# generates thoughts/plans (Interpretant generation), and formulates responses.
try:
    llm = ChatOpenAI(temperature=0, model_name="gpt-4", openai_api_key=openai_api_key)
    # GPT-4 is preferred for better reasoning needed by ReAct agents
except Exception as e:
    print(f"Error initializing LLM: {e}")
    exit()

# 2. Object Representation Component (ORC) Proxy: Simulated Environment State
# A simple dictionary representing the 'world state' (Dynamic Object proxy).
```

 doi:10.20944/preprints202504.1830.v1


```python
# Keys are object IDs, values are dictionaries of properties.
environment_state = {
    "obj_1": {"name": "red cube", "color": "red", "type": "cube", "location": "table"},
    "obj_2": {"name": "blue sphere", "color": "blue", "type": "sphere", "location": "floor"},
    "obj_3": {"name": "green pyramid", "color": "green", "type": "pyramid", "location": "table"},
    "loc_1": {"name": "table", "contains": ["obj_1", "obj_3"]},
    "loc_2": {"name": "floor", "contains": ["obj_2"]}
}
print(f"--- Initial Environment State (ORC Proxy) ---")
print(json.dumps(environment_state, indent=2))
print("-" * 40)

# 3. Tools: Bridge between Agent (IGC/RPC) and Environment (ORC Proxy)
# These tools allow the agent to perceive and act upon the simulated world.

def check_environment_state(query: str) -> str:
    """
    Simulates 'perceiving' the ORC state. Answers questions about objects/locations.
    Input: A natural language query about the environment state.
    Output: A string describing the relevant state or 'Cannot determine'.
    (This uses the LLM itself to interpret the query against the state dict - a simplification.
     A real system might have structured queries or direct state access.)
    """
    print(f"[Tool Call: CheckEnvironmentStateTool] Query: '{query}'")
    # Simple implementation: Just return the whole state for the LLM agent to parse.
    # A more robust tool would parse the query and return specific info.
    state_summary = json.dumps(environment_state, indent=2)
    print(f"[Tool Call: CheckEnvironmentStateTool] Returning state summary.")
    # Optional: Use another LLM call here to answer the query based on the state,
    # making the tool itself smarter, but for this demo, let the main agent handle it.
    # return f"Current state relevant to '{query}': {state_summary}"
    return f"Current full environment state: {state_summary}"


def modify_environment_state(action_description: str) -> str:
    """
    Simulates 'acting' upon the ORC state. Parses a description of an action
    (e.g., 'move red cube to floor') and attempts to update the environment_state dict.
    Input: A string describing the action to perform.
    Output: A confirmation or error message string.
    (Again, this uses the LLM via the agent's reasoning to decide *what* action description
     to pass. The parsing here is rudimentary for demonstration).
    """
    print(f"[Tool Call: ModifyEnvironmentStateTool] Requested Action: '{action_description}'")
    action_description = action_description.lower()
    success = False
    message = "Action could not be parsed or executed."

    # VERY Simplified Action Parsing (Replace with robust parsing logic)
    if "move" in action_description and "to" in action_description:
        parts = action_description.split("move ")[1].split(" to ")
        if len(parts) == 2:
            object_name = parts[0].strip()
            target_location_name = parts[1].strip()

            obj_id_to_move = None
```

 doi:10.20944/preprints202504.1830.v1


```
            current_location_id = None
            target_location_id = None

            # Find object ID and current location
            for obj_id, details in environment_state.items():
                if details.get("name") == object_name:
                    obj_id_to_move = obj_id
                    current_loc_name = details.get("location")
                    for loc_id, loc_details in environment_state.items():
                        if loc_details.get("name") == current_loc_name:
                            current_location_id = loc_id
                            break
                    break

            # Find target location ID
            for loc_id, details in environment_state.items():
                if details.get("name") == target_location_name:
                    target_location_id = loc_id
                    break

            if obj_id_to_move and current_location_id and target_location_id:
                try:
                    # Update object's location property
                    environment_state[obj_id_to_move]["location"] =
    target_location_name
                    # Remove object from old location's 'contains' list
                    if obj_id_to_move in environment_state[current_location_id].get("
    contains", []):
                        environment_state[current_location_id]["contains"].remove(
    obj_id_to_move)
                    # Add object to new location's 'contains' list
                    if "contains" not in environment_state[target_location_id]:
                        environment_state[target_location_id]["contains"] = []
                    environment_state[target_location_id]["contains"].append(
    obj_id_to_move)

                    success = True
                    message = f"Action successful: Moved '{object_name}' to '{
    target_location_name}'."
                    print(f"[Tool Call: ModifyEnvironmentStateTool] State Updated:")
                    print(json.dumps(environment_state, indent=2))
                except Exception as e:
                    message = f"Error updating state: {e}"
            else:
                 message = f"Could not find object '{object_name}' or location '{
    target_location_name}'."

    if not success:
        print(f"[Tool Call: ModifyEnvironmentStateTool] Action Failed/Not Parsed.")
    return message

# Create LangChain Tool instances
tools = [
    Tool(
        name="CheckEnvironmentState",
        func=check_environment_state,
        description="Useful for getting the current status of objects and locations in
    the environment. Input should be a question about the state (e.g., 'Where is the
    red cube?', 'What objects are on the table?'). Output is a description of the
    current state.",
    ),
```

 doi:10.20944/preprints202504.1830.v1


```
    Tool(
        name="ModifyEnvironmentState",
        func=modify_environment_state,
        description="Use this tool to change the state of the environment by performing
     actions like moving objects. Input should be a description of the action (e.g., '
    move red cube to floor'). Output is a confirmation or error message.",
    ),
]

# 4. Agent Executor (Orchestrator & Semiosis Loop Simulation)
# We use a ReAct agent, which explicitly reasons (Thought), decides on an Action (Tool
    call),
# and observes the result (Observation) in a loop.
# Pull the ReAct prompt template
react_prompt = hub.pull("hwchase17/react")

# Create the agent
agent = create_react_agent(llm, tools, react_prompt)

# Create the agent executor, which runs the loop
agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True,
    handle_parsing_errors=True)
# verbose=True shows the Thought->Action->Observation cycle

# --- Running the Example ---

# Input command (Linguistic Representamen)
user_command = "Please move the red cube to the floor."
print(f"\nUser Command (Representamen): {user_command}\n")
print("--- Running Peircean LSM Simulation (Agent Executor) ---")

# Invoke the agent executor to start the semiosis loop
try:
    final_response = agent_executor.invoke({"input": user_command})
    print("\n--- Agent Final Response ---")
    print(final_response.get("output", "No output field found."))
except Exception as e:
    print(f"\nAn error occurred during agent execution: {e}")


print("\n--- Final Environment State (ORC Proxy) ---")
print(json.dumps(environment_state, indent=2))
print("-" * 40)
```

Commentary and Semiotic Interpretation:

1. **Code Explanation:**
    - The `ChatOpenAI` instance (`llm`) acts as the reasoning core within the LangChain Agent. It processes the user command (Representamen), generates internal reasoning steps ("Thoughts," approximating Interpretant generation), decides which Tool (Action) to use, and formats the final response. It serves roles in both the RPC and IGC.
    - The `environment_state` dictionary acts as the **ORC proxy**, storing the current state of the simulated world (the Dynamic Object proxy).
    - The `CheckEnvironmentStateTool` allows the agent to "perceive" the state of the ORC. When called, it returns the current state description. This simulates the agent accessing information about the Dynamic Object.
    - The `ModifyEnvironmentStateTool` allows the agent to "act" upon the ORC proxy. It takes an action description formulated by the agent's reasoning process, parses it (simplistically

 doi:10.20944/preprints202504.1830.v1

here), and updates the `environment_state` dictionary. This simulates the agent's action affecting the Dynamic Object.

- The `AgentExecutor` with a ReAct agent orchestrates the **Semiosis Loop**. `verbose=True` allows us to observe the cycle:
  - **Thought:** The LLM reasons about the input command and the current goal.
  - **Action:** The LLM decides to use a tool (`CheckEnvironmentState` or `ModifyEnvironmentState`) and specifies the input for that tool.
  - **Observation:** The agent receives the output from the executed tool (either the state description or the action confirmation/error).
  - This Observation feeds back into the next Thought step, allowing the agent to reassess, plan further actions, or formulate the final answer.

2. **Semiotic Interpretation:**
   - *Representamen Input:* The user command ("Please move the red cube to the floor.") serves as the initial linguistic Representamen.
   - *Object Consultation (ORC Access):* The agent will likely first use the `CheckEnvironmentStateTool` (Action) to understand the current layout (Observation). This step represents accessing the **Object** representation (ORC proxy) to ground the interpretation of the command. The LLM processes this state information.
   - *Interpretant Generation (IGC Simulation):* Based on the input Representamen and the retrieved Object state, the LLM generates internal reasoning ("Thoughts"). This involves identifying the target object ("red cube"), its current location ("table"), the target location ("floor"), and formulating a plan to use the modification tool. This thought process simulates the generation of a situated, grounded **Interpretant** – an understanding of the required action in context.
   - *Action on Object (Affecting Dynamic Object):* The agent decides to use the `ModifyEnvironmentStateTool` (Action) with an input like "move red cube to floor". The tool's execution successfully updates the `environment_state` dictionary (the ORC proxy). This step directly models the agent's action impacting the **Dynamic Object** (proxy).
   - *Feedback and Loop Continuation:* The agent receives the confirmation message from the tool (Observation: "Action successful..."). This feedback confirms the state change. The agent might perform a final check or, concluding the task is done, formulate the final response. This feedback loop is crucial for dynamic interaction and error handling (though error handling is minimal in this sketch).
   - *Final Output (Representamen):* The agent generates a final textual response (e.g., "Okay, I have moved the red cube to the floor."), which is another Representamen communicating the outcome.
   - *Grounding Demonstrated:* Unlike the Saussurean example which focused on linking Signifiers to abstract concepts (Signifieds), this Peircean example demonstrates grounding by linking linguistic Representamens ("red cube", "floor") to specific entities and locations within an explicit world state representation (ORC proxy). The agent's interpretation and action are directly dependent on and affect this grounded state. It reasons *about* the state of the ORC proxy, not just about correlations between words.

3. **Limitations of the Sketch:** This example offers a glimpse into Peircean principles but is highly simplified.
   - *ORC Simplicity:* The `environment_state` dictionary is extremely basic. A real ORC would need to handle complex geometries, physics, uncertainty, continuous states, and potentially learn object properties dynamically.

 doi:10.20944/preprints202504.1830.v1

- *Lack of True Perception/Action:* The "perception" (`CheckEnvironmentStateTool`) and "action" (`ModifyEnvironmentStateTool`) are simulated through function calls manipulating a dictionary based on text descriptions processed by the LLM. There's no actual vision, robotics, or physics simulation. The grounding is symbolic-to-symbolic (text command -> dictionary state -> text output).
- *Rudimentary IGC:* The agent's reasoning (Interpretant generation) relies entirely on the LLM's capabilities within the ReAct framework. It lacks dedicated modules for robust planning, causal reasoning, or specialized spatial understanding as envisioned for a full Peircean LSM. Tool input/output parsing is also simplistic.
- *Limited Semiosis:* The loop is demonstrated, but the complexity of real-world, continuous semiosis involving multiple asynchronous inputs, uncertainty, and deep learning from interaction is not captured.

Even with its simplifications, this example effectively illustrates the core *architectural intent* of a Peircean-inspired LSM. Integrating sign processing (RPC/LLM) with an explicit world representation (ORC proxy) via interactive tools enables a grounded interpretation (IGC/Agent Thought) that leads to actions affecting that world state, thus closing the semiosis loop. It highlights a pathway towards systems that connect language to a dynamic representation of reality, enabling more situated and capable AI interactions.

12. Conclusion and Future Directions

This paper has argued that while Large Language Models represent a remarkable achievement in capturing the statistical structure and distributional patterns of language—effectively modeling the Saussurean system of signifiers and the Peircean manipulation of symbolic representamens—they suffer from a fundamental semantic gap. Analyses grounded in both Saussurean and Peircean semiotics converge on the conclusion that LLMs lack robust internal representations corresponding to conceptual signifieds and, crucially, remain disconnected from the dynamic objects and real-world referents that ground meaning. This detachment manifests as significant limitations in genuine understanding, reliable reasoning, factual consistency, and situated interaction. To address these shortcomings, we have introduced the concept of Large Semiosis Models (LSMs), a proposed paradigm shift towards AI architectures explicitly designed to model and integrate the core components of semiotic processes: the sign vehicle (Signifier/Representamen), conceptual meaning (Signified/Interpretant), and reference to reality (Object).

The vision for LSMs, guided by core principles of explicit semiotic modeling, integrated grounding mechanisms (perceptual, interactional, knowledge-based), rich meaning representations (conceptual, causal), and dynamic processing loops, promises substantial advancements over current LLMs. As delineated in Section 7, realizing LSMs holds the potential for AI systems exhibiting deeper semantic understanding, more robust and versatile reasoning (including causal and commonsense inference), enhanced reliability and trustworthiness through grounding and consistency checking, more natural and meaningful human-AI collaboration, truly capable embodied agents interacting effectively with the physical world, accelerated scientific discovery through integrated knowledge processing, and potentially safer and more interpretable AI through architectural design choices favoring modularity and verifiable reference. These potential benefits underscore the imperative to move beyond the current focus on scaling linguistic form towards architectures embracing semantic substance and grounded meaning.

However, the path towards realizing functional LSMs is fraught with significant scientific and engineering challenges that must be acknowledged and systematically addressed. These hurdles span multiple domains of AI research:

 doi:10.20944/preprints202504.1830.v1

- **Architectural Complexity and Integration:** Designing and implementing architectures that seamlessly integrate heterogeneous components—neural sequence processors, perceptual modules, symbolic knowledge bases or reasoners, dynamic world models, reinforcement learning agents—is extraordinarily complex. Ensuring efficient communication, coherent information flow, and end-to-end differentiability (where needed for training) across these diverse computational paradigms remains a formidable challenge requiring novel architectural blueprints.
- **Robust Grounding Mechanisms:** While various grounding strategies were proposed (perceptual, interactional, knowledge-based), achieving robust, scalable, and truly effective grounding remains an open research problem. How can symbols be reliably linked to noisy, high-dimensional perceptual data? How can abstract concepts be grounded effectively? How can knowledge-based grounding avoid the brittleness and incompleteness of current KBs? Solving the symbol grounding problem [7] in practice is central to the LSM endeavor.
- **Scalable World Modeling:** The Peircean pathway, in particular, relies heavily on the ability to build and maintain accurate, dynamic internal world models (the ORC). Developing representations that can capture the richness of real-world environments, including object persistence, physical dynamics, causality, uncertainty, and affordances, at scale and efficiently update them based on partial or noisy observations, is one of the most profound challenges in AI [38,57].
- **Rich and Flexible Meaning Representation:** Moving beyond distributional embeddings to represent conceptual knowledge (Signifieds) and generate grounded Interpretants requires significant advances in knowledge representation. Finding formalisms that combine the flexibility and learning capability of neural networks with the structure, compositionality, and reasoning power of symbolic systems (neuro-symbolic AI; [52]) is a critical research avenue. How can concepts be represented to support both classification and deep reasoning?
- **Effective Learning Paradigms:** Training LSMs, especially Peircean-inspired ones involving interaction loops, demands learning paradigms beyond supervised learning on static datasets. While RL [56], self-supervised learning on interaction data, and imitation learning are promising directions, significant challenges remain in sample efficiency, exploration strategies, reward design (especially for complex cognitive tasks), and ensuring safe learning in real-world or high-fidelity simulated environments. How can LSMs learn effectively from limited interaction or generalize robustly from simulation to reality (the sim-to-real gap)?
- **Computational Resources:** LSMs, integrating multiple complex components and potentially requiring extensive interaction-based training, are likely to be even more computationally demanding than current large-scale LLMs, posing significant resource challenges for development and deployment. Research into efficient architectures and training methods will be crucial.
- **Evaluation Metrics:** Current NLP benchmarks primarily evaluate performance on text-based tasks, often focusing on linguistic fluency or specific skills in isolation. Evaluating the core capabilities envisioned for LSMs—such as the degree of grounding, the robustness of reasoning across contexts, situational awareness, the coherence of the internal world model, or the meaningfulness of interaction—requires the development of entirely new evaluation methodologies, benchmarks, and theoretical frameworks [9].

Addressing these challenges necessitates a concerted, multi-disciplinary research effort. Key future directions include:

- **Development of Novel Architectures:** Research into hybrid neuro-symbolic architectures, modular designs integrating diverse reasoning and representation systems, and architectures explicitly supporting dynamic world modeling and grounding loops.
- **Learnable World Models:** Advancing research in unsupervised or self-supervised learning of causal models, intuitive physics, object permanence, and spatial relations directly from sensory and interaction data.

 doi:10.20944/preprints202504.1830.v1

- **Multimodal Grounding Research:** Improving techniques for fusing information across modalities and developing robust methods for linking linguistic symbols to perceptual data and embodied experiences beyond simple correlation.
- **Advanced Learning Algorithms:** Designing RL and interactive learning algorithms that are more sample-efficient, handle sparse rewards, support hierarchical task decomposition, enable safe exploration, and facilitate transfer learning between simulation and reality.
- **Data Curation and Simulation Environments:** Creating rich, large-scale datasets that explicitly link language to perception, action, and structured knowledge; developing high-fidelity, interactive simulation environments suitable for training and evaluating embodied or situated LSMs.
- **New Evaluation Benchmarks:** Designing challenging benchmarks that specifically probe for grounded understanding, robust reasoning across diverse contexts, situational awareness, compositional generalization, and meaningful interaction, moving beyond purely text-based metrics.
- **Theoretical Foundations:** Further developing the theoretical understanding of grounding, meaning representation, and semiosis within computational systems, potentially drawing deeper insights from cognitive science, linguistics, and philosophy of mind alongside semiotics.
- **Ethical Considerations:** Proactively investigating the ethical implications of potentially more capable, grounded, and autonomous AI systems enabled by the LSM paradigm, focusing on issues of safety, alignment, bias amplification through grounded representations, and societal impact.

The journey from current Large Language Models to future Large Semiosis Models represents a fundamental shift towards addressing the core limitations of contemporary AI concerning meaning and understanding. Although the path is arduous and requires confronting deep scientific and engineering challenges, the semiotic perspective provides both a clear diagnosis of the problem and a principled direction for the solution. Embracing the explicit modeling of semiotic relationships and prioritizing the grounding of symbols in conceptual meaning and reference to reality offers a compelling vision for the future of artificial intelligence through the pursuit of LSMs—a future where machines might not only process language fluently but engage with the world, and with us, through a lens of genuine, grounded understanding. This endeavor appears critical if we aim to develop AI systems that are not only powerful but also reliable, trustworthy, and ultimately beneficial.

References

1. OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774* **2023**, [2303.08774].
2. Gemini Team.; Google. Gemini: A Family of Highly Capable Multimodal Models. *arXiv preprint arXiv:2312.11805* **2023**, [2312.11805].
3. Anthropic. Introducing the next generation of Claude. Anthropic News, 2024. Accessed May 15, 2024.
4. Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. Meta AI Blog, 2024. Accessed May 15, 2024.
5. Mitchell, M.; Krakauer, D.C. The fallacy of AI functionality. *Science* **2023**, *379*, 1190–1191. https://doi.org/10.1126/science.adh1804.
6. Stokke, A. Why Large Language Models Are Poor Reasoners. *Inquiry* **2024**, pp. 1–21. Advance online publication, https://doi.org/10.1080/0020174X.2024.2323368.
7. Harnad, S. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena* **1990**, *42*, 335–346. https://doi.org/10.1016/0167-2789(90)90087-6.
8. Bisk, Y.; Holtzman, A.; Thomason, J.; Andreas, J.; Bengio, Y.; Chai, J.; et al. Experience Grounds Language. *arXiv preprint arXiv:2309.11776* **2023**, [2309.11776].
9. Frank, M.C. Symbolic behavior in networks and the emergence of grounding. *Behavioral and Brain Sciences* **2024**, *47*, e32. https://doi.org/10.1017/S0140525X2300100X.
10. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; et al. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys* **2023**, *55*, 1–38. https://doi.org/10.1145/3571730.

 doi:10.20944/preprints202504.1830.v1

11. Rawte, V.; Sheth, A.; Das, A. A Survey of Hallucination in Large Foundation Models. *arXiv preprint arXiv:2309.05922* **2023**, [2309.05922].
12. Varshney, L.R.; Alemzadeh, H.; Chen, F. Hallucination Detection: Robust and Efficient Detection of Large Language Model Hallucinations using Entropy and Variance-Based Metrics. *arXiv preprint arXiv:2402.14406* **2024**, [2402.14406].
13. Zhang, Y.; Li, Y.; Cui, L.; Cai, D.; Liu, L.; Wang, W.Y. Mitigating Hallucinations of Large Language Models via Knowledge Consistent Alignment. *arXiv preprint arXiv:2401.01307* **2024**, [2401.01307].
14. Valmeekam, K.; Marquez, M.A.R.; Mustafa, M.; Sreedharan, S.; Kambhampati, S. Can LLMs Plan? On the Logical Reasoning Capabilities of Large Language Models. *arXiv preprint arXiv:2305.15771* **2023**, [2305.15771].
15. Liu, Z.; Li, S.; Zhang, Y.; Yuan, Z.; Wu, C.H.; Fung, P. Evaluating the Logical Reasoning Ability of Large Language Models. *arXiv preprint arXiv:2402.06940* **2024**, [2402.06940].
16. Ullman, S. Large language models and the alignment problem. *Philosophical Transactions of the Royal Society B: Biological Sciences* **2023**, *378*, 20220143. https://doi.org/10.1098/rstb.2022.0143.
17. Gendron, N.; Larochelle, H.; Bengio, Y. Towards Biologically Plausible Models of Reasoning. *arXiv preprint arXiv:2401.06409* **2024**, [2401.06409].
18. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, New York, NY, USA, 2021; FAccT '21, pp. 610–623. https://doi.org/10.1145/3442188.3445922.
19. Pavlick, E. The Illusion of Understanding in Large Language Models. *arXiv preprint arXiv:2311.14747* **2023**, [2311.14747].
20. de Saussure, F. *Course in General Linguistics*; Philosophical Library, 1959. Original work published 1916.
21. Peirce, C.S. *Collected Papers of Charles Sanders Peirce*; Vol. 1–8, Harvard University Press, 1931–1958.
22. Chase, H. LangChain Python Library. GitHub Repository, 2023.
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; et al. Attention is All You Need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017); Guyon, I.; Luxburg, U.V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.V.N.; Garnett, R., Eds. Curran Associates, Inc., 2017, pp. 5998–6008.
24. Gu, A.; Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752* **2023**, [2312.00752].
25. Kim, Y.; Lee, H.; Kim, G. Probing Attention Patterns in State-of-the-Art Transformers. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2024, Vol. 38.
26. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, 2019; pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423.
27. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; et al. Language Models are Few-Shot Learners. In Proceedings of the Advances in Neural Information Processing Systems 33 (NeurIPS 2020); Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.F.; Lin, H.T., Eds. Curran Associates, Inc., 2020, pp. 1877–1901.
28. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; et al. Training language models to follow instructions with human feedback. In Proceedings of the Advances in Neural Information Processing Systems 35 (NeurIPS 2022); Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; Oh, A., Eds. Curran Associates, Inc., 2022, pp. 27730–27744.
29. Rafailov, R.; Sharma, A.; Mitchell, E.; Ermon, S.; Manning, C.D.; Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In Proceedings of the Advances in Neural Information Processing Systems 36 (NeurIPS 2023); Globerson, A.; Saenko, K.; Zhang, C.; Mandt, S., Eds., 2023.
30. Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T.B.; Chess, B.; Child, R.; et al. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* **2020**, [2001.08361].
31. Hoffmann, J.; Borgeaud, S.; Mensch, A.; Buchatskaya, E.; Cai, T.; Rutherford, E.; et al. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556* **2022**, [2203.15556].
32. Li, X.; Yu, Z.; Zhang, Z.; Wang, C.; Liu, Y.; Jin, Q. The Era of Big Small Language Models: A Survey. *arXiv preprint arXiv:2401.17919* **2024**, [2401.17919].

 doi:10.20944/preprints202504.1830.v1

33. Chan, S.C.Y.; Santoro, A.; Lampinen, A.K.; Wang, J.X.; Singh, S.; Hill, F. Data Distributional Properties Drive Emergent Few-Shot Learning in Transformers. In Proceedings of the Advances in Neural Information Processing Systems 35 (NeurIPS 2022); Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; Oh, A., Eds. Curran Associates, Inc., 2022, pp. 28169–28181.
34. Yuksekgonul, M.; Jurafsky, D.; Zou, J. Attribution and Alignment: Effects of Preference Optimization on the Referential Capabilities of Large Language Models. *arXiv preprint arXiv:2402.00903* **2024**, [2402.00903].
35. Barsalou, L.W. Grounded cognition. *Annual Review of Psychology* **2008**, *59*, 617–645. https://doi.org/10.1146/annurev.psych.59.103006.093639.
36. Lake, B.M.; Murphy, G.L. Word meaning in minds and machines. *Psychological Review* **2021**, *128*, 401–420. https://doi.org/10.1037/rev0000263.
37. Marcus, G. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. *arXiv preprint arXiv:2002.06177* **2020**, [2002.06177].
38. Chen, Z.; Wang, W. Emergent World Models in Large Language Models. *arXiv preprint arXiv:2402.08788* **2024**, [2402.08788].
39. Marcus, G.; Davis, E. GPT-4's successes, and GPT-4's failures. Marcus on AI Substack, 2023.
40. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings of the Advances in Neural Information Processing Systems 33 (NeurIPS 2020); Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.F.; Lin, H.T., Eds. Curran Associates, Inc., 2020, pp. 9459–9474.
41. Pan, S.; Luo, L.; Wang, Y.; Chen, C.; Wang, J.; Wu, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *arXiv preprint arXiv:2306.08302v3* **2024**, [2306.08302].
42. Azaria, A. A Theoretical Framework for Hallucination Mitigation. *arXiv preprint arXiv:2402.03407* **2024**, [2402.03407].
43. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Proceedings of the Advances in Neural Information Processing Systems 35 (NeurIPS 2022); Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; Oh, A., Eds. Curran Associates, Inc., 2022, pp. 24824–24837.
44. Dziri, N.; Milton, S.; Yu, M.; Zaiane, O.; Reddy, S. Faith and Fate: Limits of Transformers on Compositionality. In Proceedings of the Advances in Neural Information Processing Systems 36 (NeurIPS 2023); Globerson, A.; Saenko, K.; Zhang, C.; Mandt, S., Eds., 2023.
45. Kıcıman, E.; Ness, R.; Sharma, A.; Tan, C. Causal Reasoning and Large Language Models: Opening a New Frontier for Causality. *arXiv preprint arXiv:2305.00050* **2023**, [2305.00050].
46. Zečević, M.; Singh M., D.D.S.; Kersting, K. Evaluating Causal Reasoning Capabilities of Large Language Models. *arXiv preprint arXiv:2305.12517* **2023**, [2305.12517].
47. Jin, Z.; Feng, J.; Xu, R.; Zheng, W.; Liu, S.; Chen, J.; et al. Can Large Language Models Truly Understand Causal Statements? A Study on Robustness. *arXiv preprint arXiv:2402.10249* **2024**, [2402.10249].
48. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems 26 (NeurIPS 2013); Burges, C.J.C.; Bottou, L.; Welling, M.; Ghahramani, Z.; Weinberger, K.Q., Eds. Curran Associates, Inc., 2013, pp. 3111–3119.
49. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedings of the Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014; pp. 1532–1543. https://doi.org/10.3115/v1/D14-1162.
50. Manning, C.D.; Schütze, H. *Foundations of Statistical Natural Language Processing*; MIT Press, 1999.
51. Santaella, L. *Teoria Geral dos Signos: Como as linguagens significam as coisas*; Pioneira, 2000.
52. Garcez, A.S.d.; Lamb, L.C. Neurosymbolic AI: The 3rd Wave. *arXiv preprint arXiv:2012.05876* **2020**, [2012.05876].
53. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks* **2015**, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.
54. Pan, A.; Kushman, N.; Liang, P.; Pasupat, P. Logic and Language Models. 2024, [2401.08143].
55. Lin, B.Y.; Lee, W.; Liu, J.; Ren, P.; Yang, Z.; Ren, X. Generating Benchmarks for Factuality Evaluation of Language Models. *arXiv preprint arXiv:2311.08491* **2023**, [2311.08491].

 doi:10.20944/preprints202504.1830.v1

56. Sutton, R.S.; Barto, A.G. *Reinforcement learning: An introduction*, 2nd ed.; MIT Press, 2018.
57. Lake, B.M.; Ullman, T.D.; Tenenbaum, J.B.; Gershman, S.J. Building machines that learn and think like people. *Behavioral and Brain Sciences* **2017**, *40*, E253. https://doi.org/10.1017/S0140525X1600183X.