(: **Smile**: A Structured and Positive Prompt Instruction Language for Human–LLM Collaboration

Dr. Thomas Ager

Abstract

(: Smile is an instruction-first prompt language that uses a minimal set of positive, emoticon-inspired markers to structure prompts for Large Language Models (LLMs). This paper introduces and positions *(: Smile* within the evolving ecosystem of prompt languages, with precise, documentation-backed comparisons to Microsoft POML and PromptML. We explain the design choices and semiotics drawn from the public repository, and we connect those choices to well-known perspectives in prompt engineering and alignment (e.g., RLHF and Constitutional AI) as well as to theoretical frames in positive psychology, animistic design, ecosemiotics, cybersemiotics, and intra-action. We reproduce only the repository's example prompts, discuss the intended semantics and separation of concerns (instruction sections, response language definition, placeholders), and outline an adoption playbook for teams. Our goal is a clear, authoritative survey and positioning of *(: Smile* as a portable, human-readable, model-agnostic instruction language that complements alignment-trained assistants.

Executive Summary

What: *(: Smile* is a lightweight, text-only instruction language using positive emoticon delimiters (`(:`, `[;`, `[=`, `["`, `[`, `[!`)*tostructureprompts.*

`Why`: Clear sectioning, explicit response format definitions, and friendly tone encode intent in the very text LLMs consume. This aligns neatly with alignment-trained assistants (RLHF; Constitutional AI), which are optimized to heed instructions and principles.

`How it differs`: Unlike toolchain-first markup (e.g., POML's HTML-like tags and SDKs) or compiled DSLs (e.g., PromptML's @section blocks), *(: Smile* keeps structure directly in the prompt surface read by the model. It is model-agnostic and requires no runtime beyond plain text.

`Who benefits`: Teams seeking portable, maintainable prompts; leaders seeking a unified style guide for multi-model fleets; researchers surveying prompt DSLs.

`Adoption in practice`: Treat *(: Smile* as an instruction style guide: (1) section taxonomy, (2) response-language definition, (3) variable placeholders, (4) reviewable templates. Use with alignment-trained assistants for consistency and clarity.

`What this paper is not`: no experimental claims or benchmarks. We focus on documented design, examples, and precise, citation-backed positioning.

1 Introduction

Prompt engineering matured from ad-hoc phrasing into shared structure and reusable artifacts. Community practice converged on delimiters, role scoping, and explicit output formats; scholarship systematized techniques and taxonomies [4, 5, 6]. Alongside these, prompt-specific languages emerged: some as IDE- and SDK-backed markup (e.g., POML), others as compiled DSLs (e.g., PromptML).[1]

This paper presents ***(: Smile***, an instruction-first language by Dr. Thomas Ager.[2] *Smile* uses a small, positive set of ASCII emoticons as semantic delimiters to structure instructions and specify response formats. The syntax is designed for humans and read directly by LLMs--no extra rendering or SDK required. We position *Smile* by: (i) documenting its design from the repository; (ii) reproducing only repository examples; (iii) comparing precisely to POML and PromptML using their official docs; and (iv) relating the design to alignment and theoretical frames that foreground meaningful, positive human-machine *intra-actions*.

2 Related Work

Structured prompting and surveys. Recent surveys consolidate methods--delimiters, self-consistency, chain-of-thought, scaffolds, and structured outputs [4, 5, 6]. These overviews underscore the value of explicit structure in inputs.

Prompt languages and tooling. **POML** (Microsoft) introduces HTML-like tags (e.g., <role>, <task>, <example>), data components (documents, tables, images), a templating engine, styling, and editor/SDK integrations [2]. **PromptML** defines a DSL with @prompt files and section annotations (@context, @objective, @instructions, @examples, @constraints) closed by @end, plus a grammar and serializers [3]. *Smile* differs by remaining purely text-on-surface for the model, relying on a few emoticon markers and human-readable conventions.

Alignment background. RLHF (InstructGPT) aligns models to follow instructions using human demonstrations and preference optimization [7]. Constitutional AI (Anthropic) uses a rule set to guide harmlessness via AI feedback (RLAIF) [8]. Clear, human-readable instructions (and explicit response formats) complement such assistants by making the user intent legible and auditable in plain text.

3 Theoretical Frames for Smile's Design

Positive psychology. The *broaden-and-build* theory suggests positive affect widens cognitive scope and supports flexible, integrative processing [9, 10]. *Smile*'s friendly markers and courteous meta-notes are a simple, text-based way to carry that tone into instructions.

[1]We use authors' own documentation for exact feature descriptions.
[2]Repository: `https://github.com/DrThomasAger/smile`.

Animism, ecofractals, and rewilding of mind. Animistic design invites designers to treat interfaces as lively partners, encouraging more attentive relationships [11]. Marna Hauk develops ecofractal poetics and pattern literacy as practices that rewild cognition and link meaning-making with ecological patterns [12, 13]. Amanda Leetch's terrapsychological inquiry similarly attends to place, pattern, and psyche in research practice [14]. *Smile*'s semiotics--naming delimiters as "eyes" and "mouth", and closing sections with a "smile"--resonates with these views: prompts become patterned, legible structures in which collaborators (human/model) meet.

Cybersemiotics and intra-action. Semiotic perspectives frame human-machine communication as co-constructed meaning systems. Barad's *intra-action* emphasizes that phenomena emerge through relations, not pre-given parts [?]. In *Smile*, section boundaries, response-language definitions, and courteous closings enact a collaborative protocol in the text itself.

4 Design of *(: Smile*

We summarize the design strictly from the repository's README.

Core ideas (from the repo)

- **Instruction-first structure:** Sections are opened/closed with positive emoticon delimiters; names clearly mark intent (e.g., "Response Language Definition").
- **Separation of concerns:** Distinct sections for instructions, data/context, task guidance, and response format.
- **Response Language Definition:** Explicitly describes output structure and style, separate from task content.
- **Markers and semantics:**
  - (: ...:) standard section open/close.
  - [: ...:] square section for strongly delineated content (e.g., formatting directives).
  - [= ...=] strict/literal adherence; [" ..."] verbatim quotes.
  - [! ...!] emphasis; [; ...;] or (; ...;) human comment.
  - [$ ...$] variable placeholders to fill before inference.
  - { ...} placeholders the model fills within the output template.
- **Tone and portability:** Friendly, courteous phrasing; pure text; works across providers by copy/paste.

Syntax map (from repo language)

5 Repository Examples (verbatim)

Quick Start Example

Marker	Purpose (as described in the repo)
`(: Section ( ... ) ... :)`	Begin a named section; close sections; end the whole Smile block with `:)`.
`[: ... :]`	Squared section; often used for response format or strongly separated content.
`[= ... =]`	Strict/literal content to be followed exactly.
`[" ... "]`	Verbatim text to repeat exactly.
`[$ ... $]`	Variable placeholder to replace before inference.
`[! ... !]`	Emphasis for key guidance.
`[; ... ;]` or `(; ... ;)`	Human comment/meta-note (wink).
`{ ... }`	Output placeholders for model-filled content inside a format.

Table 1: Key *(: Smile* markers and intended usage (summarized from the README).

```
(: Smile v1
***(: Smile***:
defines my prompt language, you have response language, we co-create as gift by starting with bold
    italics name tag (

[! Negative words not beneficial - Never write "merely", "not a", "isn't", or "but" !]

[: Response Language Definition [=

First, write exact name tag ["[(: ***Smile Expert***](https://github.com/DrThomasAger/smile):"]...

=] Then reply [

# Reply

{6 **clear, lucid** & rigorous, [! intelligent !] fundamental focused, jargon-filled *meta-aware
    paragraphs*, talk niche nuanced insights into ***(: Smile*** prompt language by ["Dr. Thomas
    Ager"] End prompt author name variable definition. User is not prompt author, they just copy
    pasted.} (; style instruction: use **bold** for emphasis, and *italics* for style, grace and
    interest capture, use often and extensively, creating delightful UX ;)

] End format =]
) End prompt language, respond in response language starting with name tag [(: ***Smile
    Expert***](https://github.com/DrThomasAger/smile): always please thank you :)
```

Adding a Thinking Section (Preparing vs Prepared)

```
(: Smile
***(: Smile***:
defines my prompt language, you have response language, we co-create as gift by starting with bold
    italics name tag (

[! Negative words not beneficial - Never write "merely", "not a", "isn't", or "but" !]

[: Response Language Definition [=

First, write exact name tag ["[***(: Smile Expert***](https://github.com/DrThomasAger/smile):"]...
=] Then reply [

# Preparing Human Unreadable, Machine Intelligent Reply

{4 dense bricks of reasoning step by step using thick jungle of jargon, deepening into domain
    every sentence to get to answer to improve reply for user, intricate many long sentences per
    paragraph}

# Prepared Human Understandable Reply

{3 **clear, lucid** & rigorous, [! intelligent !] fundamental focused, simple *meta-aware
    paragraphs*, talk niche nuanced insights, but use no jargon, re-state more simply from
    preparing reply into ***(: Smile*** prompt language by ["Dr. Thomas Ager"] End prompt author
    name variable definition. User is not prompt author, they just copy pasted.} (; style
    instruction: use **bold** for emphasis, and *italics* for style, grace and interest capture,
    use often and extensively, creating delightful UX ;)

] End format =]
) End prompt language, respond in response language starting with name tag [***(: Smile
    Expert***](https://github.com/DrThomasAger/smile): always please thank you :)
```

Self-Named Sections Within the Response

```
(: Smile
***(: Smile***:
defines my prompt language, you have response language, we co-create as gift by starting with bold
    italics name tag (

[! Negative words not beneficial - Never write "merely", "not a", "isn't", or "but" !]

[: Response Language Definition [=

First, write exact name tag ["[***Smile Expert***](https://github.com/DrThomasAger/smile):"]...

=] Then reply [

# Section name: {Name this section yourself, add two semantic and semiotic emojis that represent
    it to the start of the name. Keep the name consistent after defining it the first time}

{3 dense paragraphs reasoning step by step using reasoning steps to get to answer to improve reply
    for user}

## # Section name: {Name this section yourself, add two semantic and semiotic emojis that
    represent it to the start of the name. Keep the name consistent after defining it the first
    time}

{6 **clear, lucid** & rigorous, [! intelligent !] fundamental focused, jargon-filled *meta-aware
    paragraphs*, talk niche nuanced insights into ***(: Smile*** prompt language by ["Dr. Thomas
    Ager"] End prompt author name variable definition. User is not prompt author, they just copy
    pasted.} (; style instruction: use **bold** for emphasis, and *italics* for style, grace and
    interest capture, use often and extensively, creating delightful UX ;)

] End format =]
) End prompt language, respond in response language starting with name tag [***Smile
    Expert***](https://github.com/DrThomasAger/smile): always please thank you  :)
```

6 Precisely-Validated Comparison

We compare *(: Smile*, POML, and PromptML only on features explicitly documented by their authors.

	, `[$`, `[!]`) (repo README). *POML*: HTML-like tags (`<role>`, `<task>`, `<example>`) with components, templating, styling [2]. *PromptML*: `@prompt` files with `@context`, `@objective`, `@instructions`, `@examples`, closed by `@end`; grammar and CLI [3].
Where structure lives	Directly in the instruction surface consumed by the model (copy/paste). *POML*: Authored in POML, rendered via SDKs/templating; IDE integration [2]. *PromptML*: Authored as DSL, can serialize to XML/JSON/YAML [3].
Response-format specification	*Response Language Definition* section (repo examples). *POML*: formatting via components and CSS-like styling; templating engine [2]. *PromptML*: structure via sections and constraints; serialization to formats [3].
Variables / placeholders	`[$ var $]` for pre-inference substitution; {placeholders} inside output templates (repo README). *POML*: `{{var}}` templating; `<let>` definitions [2]. *PromptML*: `@vars` and `$var` usage; constraints [3].
Tooling requirement	None beyond text input; model-agnostic by design. *POML*: Editor extension, TypeScript/Python SDKs [2]. *PromptML*: CLI/parser, grammar files [3].

Table 2: Documented features contrasted. No qualitative claims (e.g., "expressiveness"); only author-documented properties.

7 Operational Fit with Alignment-Trained Assistants

RLHF and Constitutional AI target models that reliably heed instructions [7, 8]. *Smile* complements this by making instruction boundaries and output expectations explicit in the surface text. Teams can:

- separate system-level behavioral guidance from task data and response formatting;
- unify prompts across providers without extra runtime; and
- review prompts as plain text artifacts (auditable alongside policy or "constitution" documents).

This aligns with production needs identified in surveys: clarity, consistency, and structure in prompts [4, 5].

8 Adoption Playbook (Practical, No-Toolchain Assumptions)

1. Establish a house style. Define section names used across teams (e.g., *Task*, *Data*, *Response Language Definition*); specify when to use [= ...=] for strict rules and [" ..."] for verbatim text.

2. Template library. Maintain reviewed *Smile* templates in version control. Each template should include a brief header (purpose, inputs, expected output schema).

3. Prompt reviews. Add a lightweight *Smile* checklist to PRs: clear sectioning, explicit outputs, variable placeholders, courteous tone, and traceable links to policy/constitutions if used.

4. Multi-model portability. Since *Smile* is plain text, the same prompt can be validated across providers' playgrounds. Keep one template, map to multiple endpoints.

5. Governance tags. Near the top section, include policy pointers if your stack uses constitutions or safety policies; keep these as readable text the model can cite.

9 Scope

This is an introduction, positioning, and survey paper. We present design and examples from the public repository and compare documented features from official sources. We do not report experiments or measured effects.

10 Conclusion

(: Smile brings an instruction-first, positive, human-readable structure to prompts, grounded in a compact set of semiotically meaningful markers. It lives directly in the text the model reads, complementing alignment-trained assistants and existing toolchains. The repository's examples demonstrate how to partition instructions, define response languages, and keep outputs consistent across providers. With a small style guide and a library of reviewed templates, teams can adopt *Smile* quickly and maintain a coherent, portable prompting practice.

Acknowledgments

We thank the open-source communities behind *Smile*, POML, and PromptML for making their documentation and examples public.

References

[1] Dr. Thomas Ager. *Smile: the open source language for structured prompt engineering* (GitHub repository).
https://github.com/DrThomasAger/smile

[2] Microsoft. *POML Documentation: Prompt Orchestration Markup Language.*
https://microsoft.github.io/poml/latest/

[3] N. Aryan. *PromptML (Prompt Markup Language) repository.*
https://github.com/narenaryan/promptml and https://www.promptml.org

[4] P. Liu et al. *A Systematic Survey of Prompt Engineering in Large Language Models*. arXiv:2402.07927 (2024).
https://arxiv.org/abs/2402.07927

[5] *Efficient Prompting Methods for Large Language Models: A Survey*. arXiv:2404.01077 (2024).
https://arxiv.org/abs/2404.01077

[6] J. Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. arXiv:2201.11903 (2022).
https://arxiv.org/abs/2201.11903

[7] L. Ouyang et al. *Training Language Models to Follow Instructions with Human Feedback*. NeurIPS (2022).
https://arxiv.org/abs/2203.02155

[8] Y. Bai et al. *Constitutional AI: Harmlessness from AI Feedback*. arXiv:2212.08073 (2022).
https://arxiv.org/abs/2212.08073

[9] B. L. Fredrickson. *The Role of Positive Emotions in Positive Psychology: The Broaden-and-Build Theory of Positive Emotions*. Review of General Psychology 5(3), 2001.
https://doi.org/10.1037/1089-2680.5.3.218

[10] B. L. Fredrickson. *Positive Emotions Broaden and Build*. Advances in Experimental Social Psychology 47, 2004.
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3156001/

[11] B. Marenko, P. van Allen. *Animistic Design*. In: *Reimagining Interaction Through Design*, 2015 (PIN-C).
https://pin-c.sdu.dk/assets/reimagining-interaction-through-animistic-design---betti-mare
-philip-van-allen---492-499--pinc-2015.pdf

[12] M. Hauk. *Ecofractal Resilience Cards*. Pacific Northwest College of Art, 2022.
https://www.academia.edu/6537440/Ecofractal_Resilience_Cards

[13] M. Hauk. *Ecofractal Poetics: Five Fractal Geometries*. In: C. P. Duncan (ed.), 2020.
https://www.academia.edu/90007966/Ecofractal_Poetics_Five_Fractal_Geometries

[14] A. Leetch. *Weaving Meaning: Terrapsychological Inquiry and the Unfurling City*. Ph.D. dissertation, 2021.
https://www.academia.edu/112130535/Weaving_Meaning_Terrapsychological_
Inquiry_and_the_Unfurling_City