16th International Cognitive Linguistics Conference

HHU Düsseldorf - August 8, 2023

Redefining the current discourse space model as a recursive monadic architecture

Yoichiro Hasebe

Doshisha University, Japan

yhasebe@mail.doshisha.ac.jp

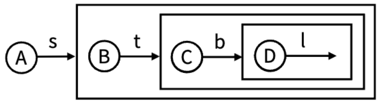
Handout PDF

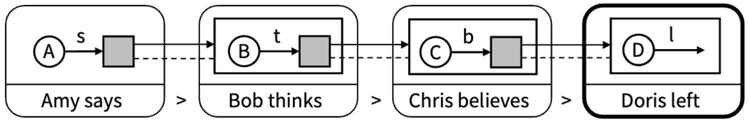
同志社大学
Doshisha University

Introduction

This presentation attempts to model the construction process of **monologic/dialogic discourse**.

The **goal** is twofold:

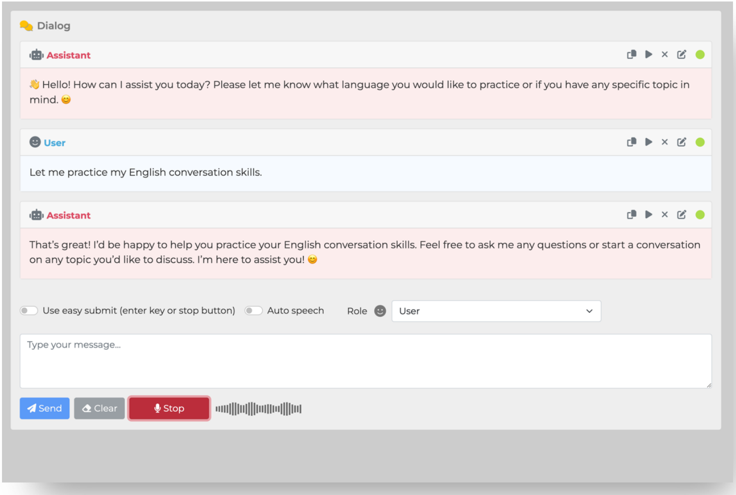




1) To redefine Langacker’s notion of the **current discourse space (CDS)** as a recursive structure that incorporates the idea of “**monads**” from the **functional programming (FP)** paradigm.

2) To propose a means to simulate CDS as a monadic recursive structure on a computer program using a **GTP-based text/chat completion API**.

Monadic Chat: https://yohasebe.github.io/monadic-chat



Assistant

👋 Hello! How can I assist you today? Please let me know what language you would like to practice or if you have any specific topic in mind. 😊

User

Let me practice my English conversation skills.

Assistant

That’s great! I’d be happy to help you practice your English conversation skills. Feel free to ask me any questions or start a conversation on any topic you’d like to discuss. I’m here to assist you! 😊

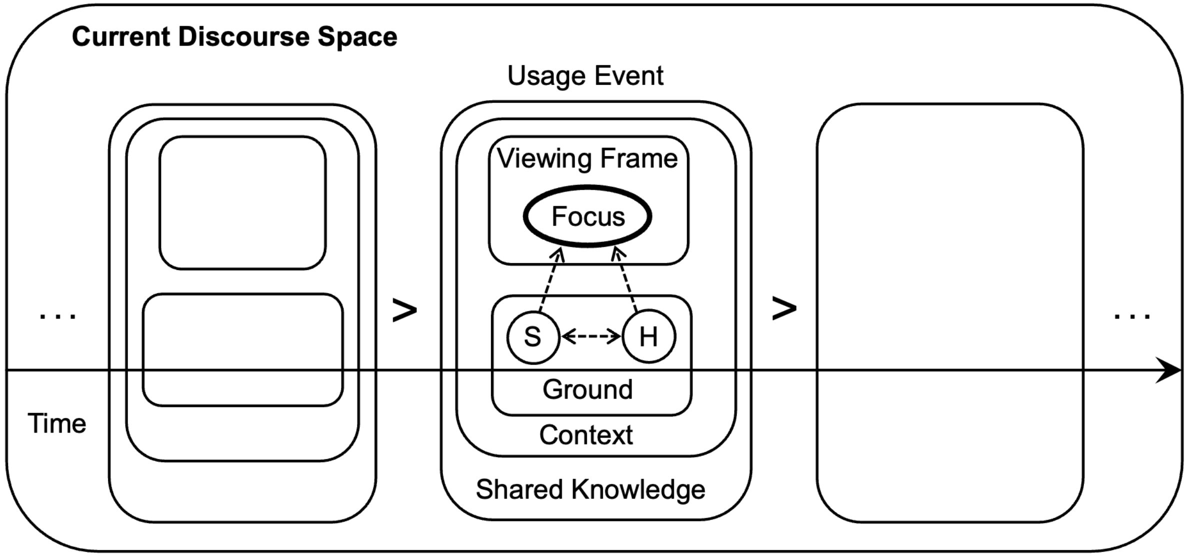
○ Use easy submit (enter key or stop button) ○ Auto speech Role User

Type your message...

Current Discourse Space (CDS)

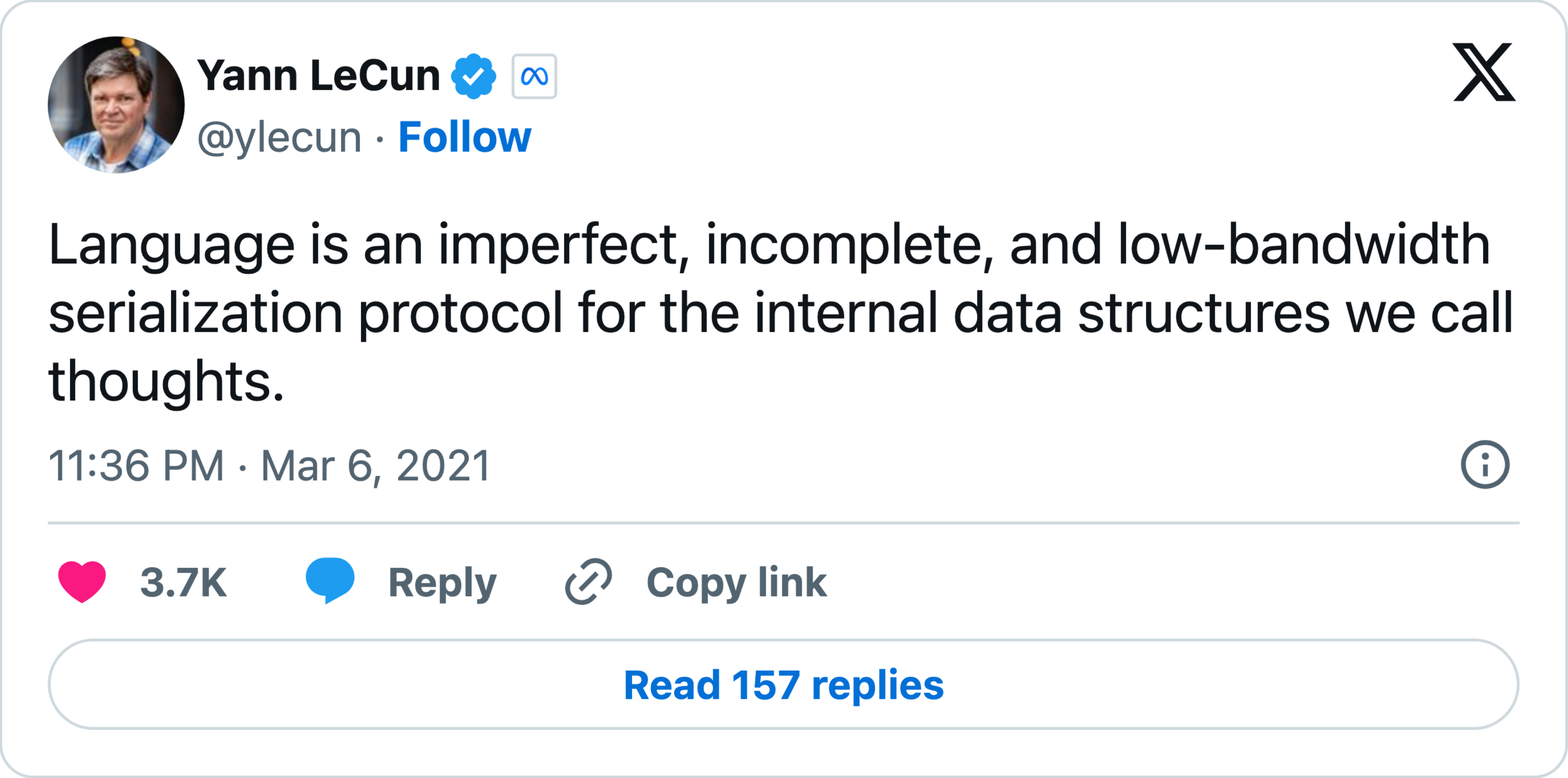
Regarding the **sequential nature** of language processing, Langacker (2001: 145) states as follows:

> Metaphorically, it is as if we are “looking at” the world through a window, or *viewing frame*. The immediate scope of our conception at any one moment is limited to what appears in this frame, and the *focus* of attention—what an expression *profiles* (i.e. designates)—is included in that scope.



Langacker (2001: 145)

Current Discourse Space (CDS)



Yann LeCun
@ylecun

Language is an imperfect, incomplete, and low-bandwidth serialization protocol for the internal data structures we call thoughts.

11:36 PM · Mar 6, 2021

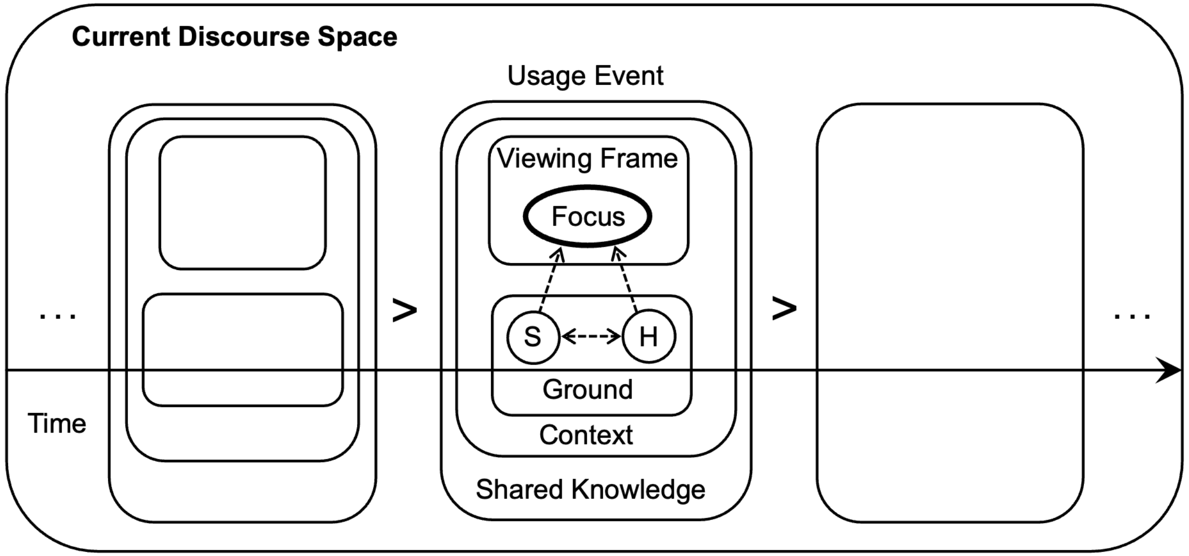
Current Discourse Space (CDS)

While new elements appear one after another in the **viewing frame**, the surrounding "environment" containing such elements as the **ground**, **context**, or **shared knowledge** looks like always staying there though being constantly updated.

Question: How is this possible?

Should we assume something like "**global variables**" in computer programming to keep everything in a certain place?

No.

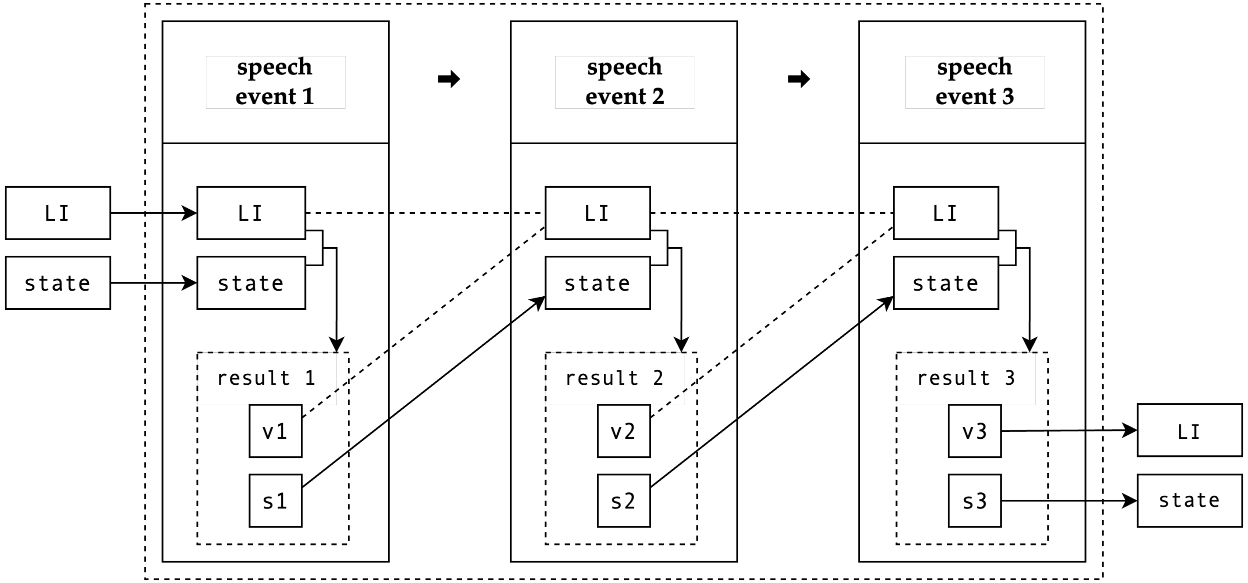


Langacker (2001: 145)

Linguistic Structures as Instructions

> *Following a suggestion by Harder (1996), we might think of linguistic structures (of whatever size) as instructions to modify the current discourse space in particular ways. Each instruction involves the focusing of attention within a viewing frame. A discourse comprises a succession of frames each representing the scene being "viewed'' and acted on by the speaker and hearer at a given instant.*

(Langacker 2001: 151)



Input:

- Linguistic instructions (LI)
- State (ground, context, shared knowledge)

Output:

- Updated State
- Linguistic instructions (LI)

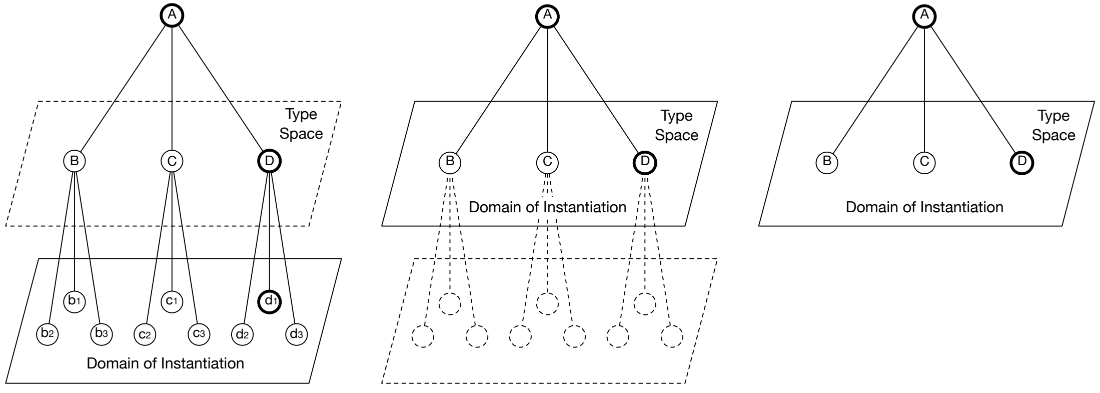
Concept of Monad

What is **monad**?

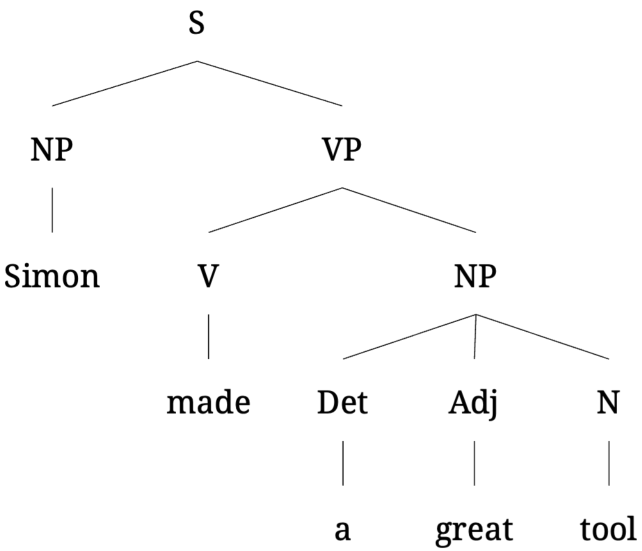
- The concept of a monad is based on the **category theory** of mathematics and is widely used in the FP paradigm in computer science. (Wadler 1995; Hutton 2016; Petricek 2018).
- A monad is often described as a "**pipeline structure for handling a value wrapped in an environment that recursively evolves**."
- A structure that satisfies the conditions for being a monad is characterized by its ability to execute operations **sequentially and continuously** while updating the structure given as the environment.
- A monad maintains its basic **homomorphism**, both in its initial state and in states that evolved through multiple operations.

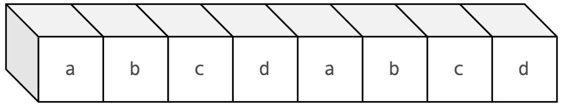
Another data structure analogy?

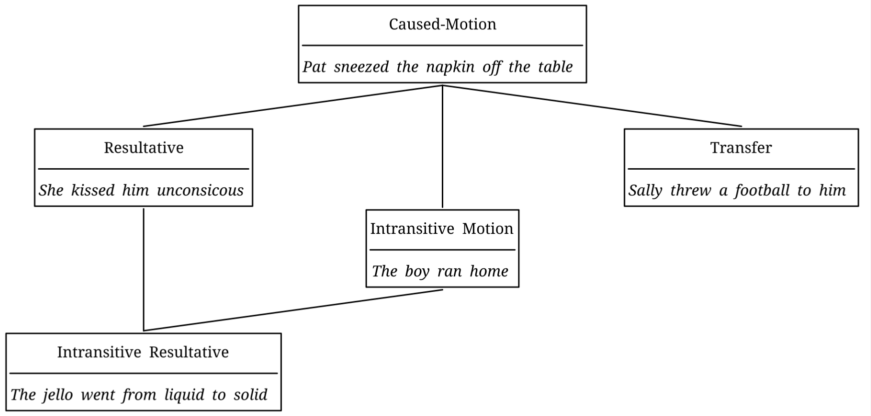
- List
- Set
- Tree
- Network
- Hierarchy
- Class/Instance
- Domain/Function/Mapping
- **Monad**?



(a) (b) (c)







Caused-Motion

Pat sneezed the napkin off the table

Resultative

She kissed him unconsicous

Intransitive Motion

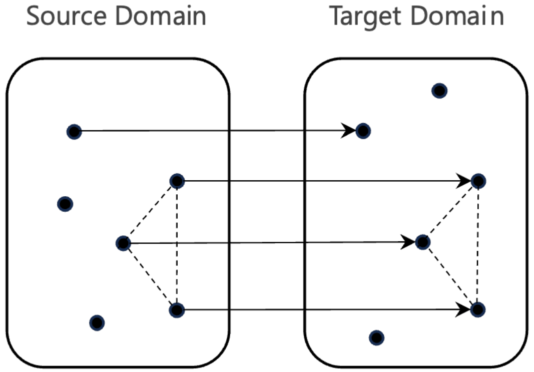
The boy ran home

Transfer

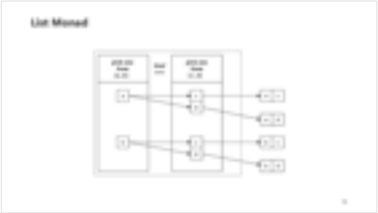
Sally threw a football to him

Intransitive Resultative

The jello went from liquid to solid



Monads in FP



List Monad

- **List Monad**
  The List Monad in functional programming represents **non-deterministic computation**. It's a way of chaining operations on lists together, where each operation could potentially return multiple results (i.e., a list of results). This is akin to exploring many possible computational "paths" at once.

Maybe Monads

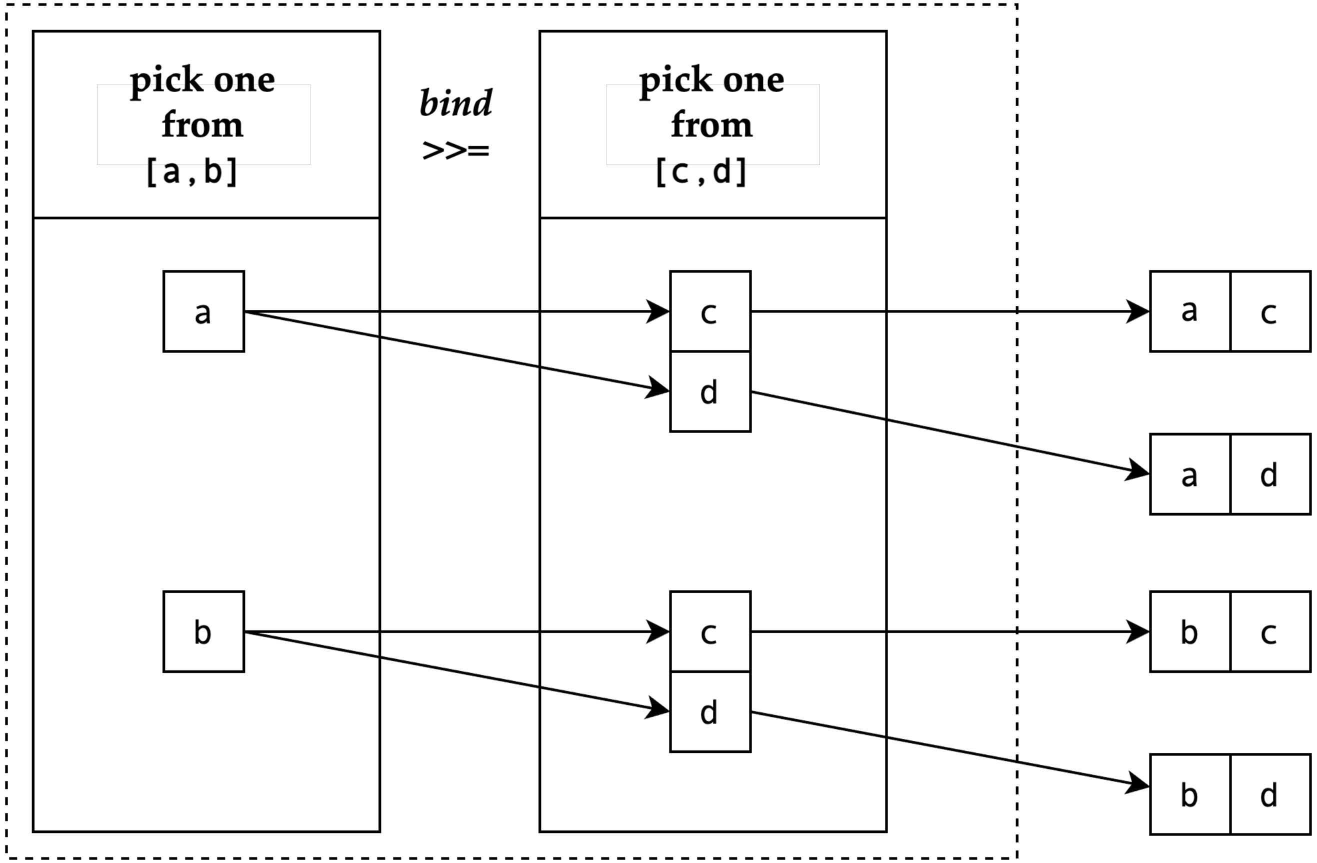
- **Maybe Monad**
  The maybe monad handles **computations that might fail** or return nothing (null). Instead of having to check for null or error conditions at every step, you can chain computations together with the Maybe Monad.

State Monad

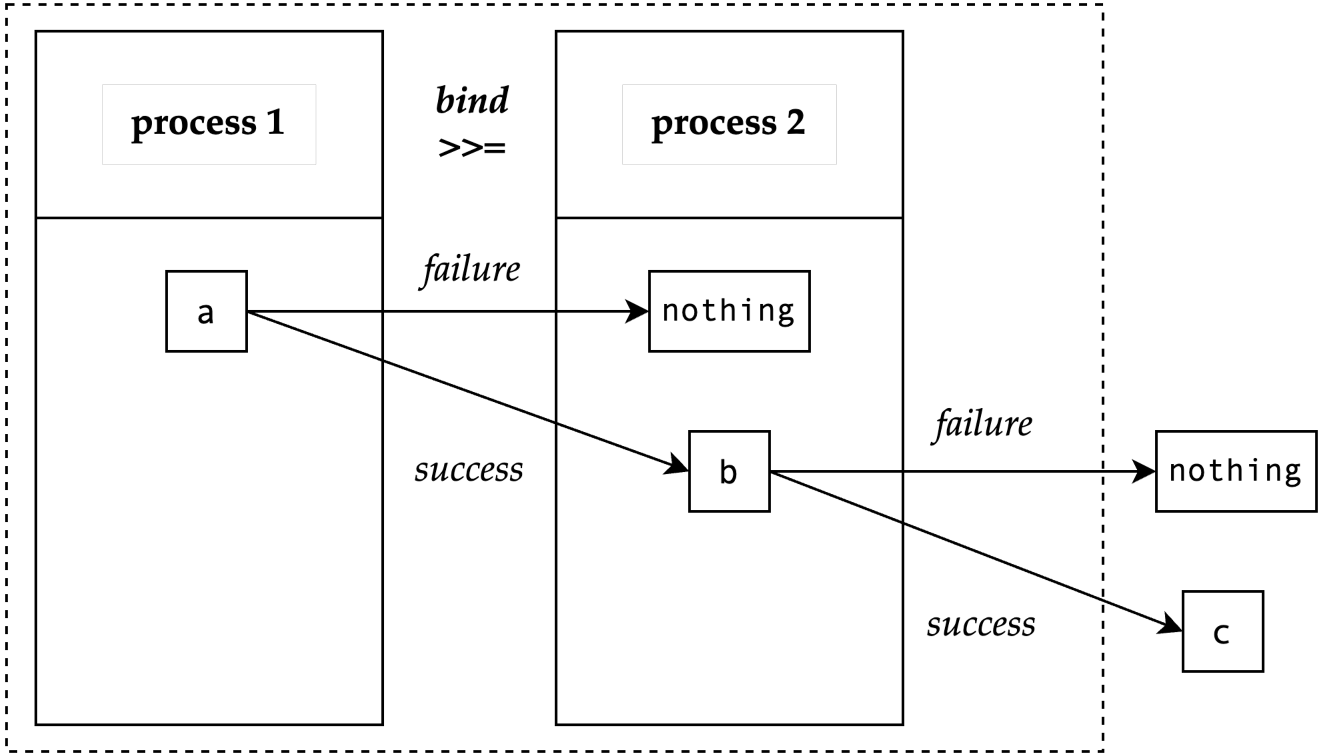
By viewing discourse as a **state monad**, we can regard a sequence of usage events as a **recursive function application** with a state object passed from one usage event to another.

- **State Monad**
  The State Monad is a construct in functional programming that provides a way to **handle state without relying on global variables**. Functional programming languages are typically stateless, and the State Monad offers a way to carry state through a sequence of computations in a controlled and predictable manner.

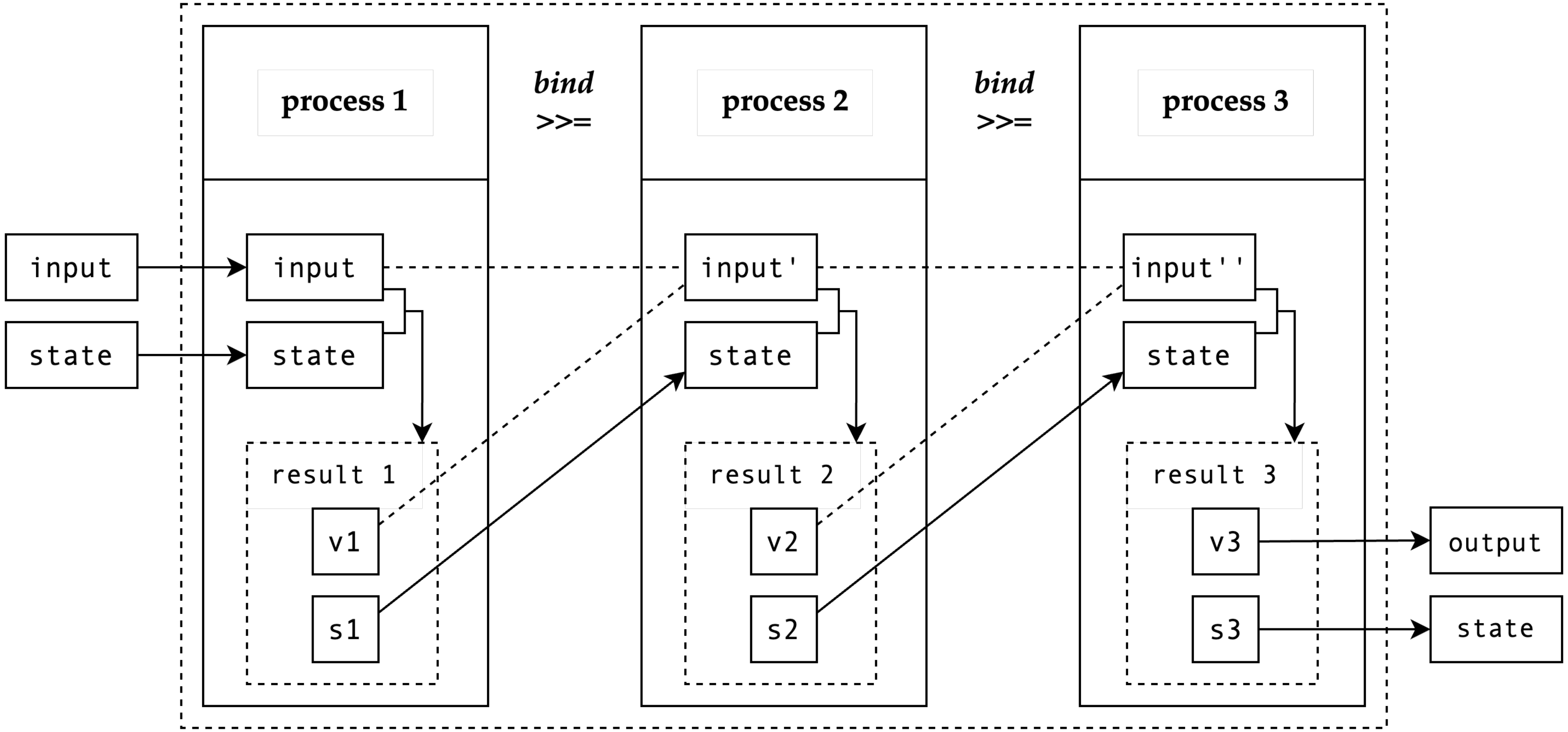
List Monad



Maybe Monads



State Monad



By viewing discourse as a **state monad**, we can regard a sequence of usage events as a **recursive function application** with a state object passed from one usage event to another.

Conditions for Monadic Structure

What do these monads have in common?

1. There is a procedure called **unit** that wraps the target value *a* in the environment

$$\mathrm{unit} :: a \rightarrow \boxed{a}$$

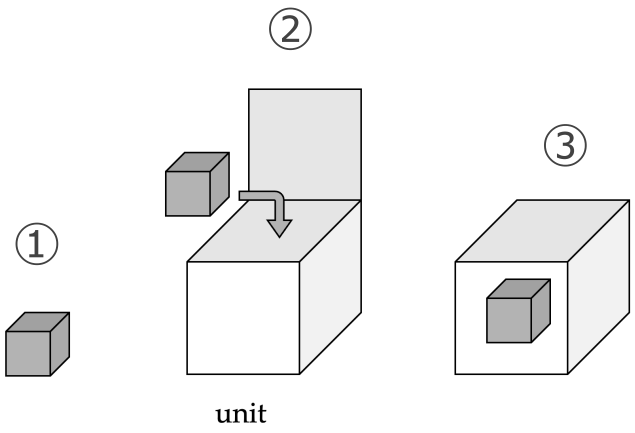
2. There is a procedure called **map** that "lifts" a function *f* to another function *f′* that deals with the value wrapped in an environment

$$\mathrm{map} :: (\, a \rightarrow b \,) \rightarrow (\boxed{a} \rightarrow \boxed{b}\,)$$

3. There is a procedure called **join** that flattens a doubled layer of environments.

$$\mathrm{join} :: \boxed{\boxed{a}} \rightarrow \boxed{a}$$

Monadic Operation: *Unit*

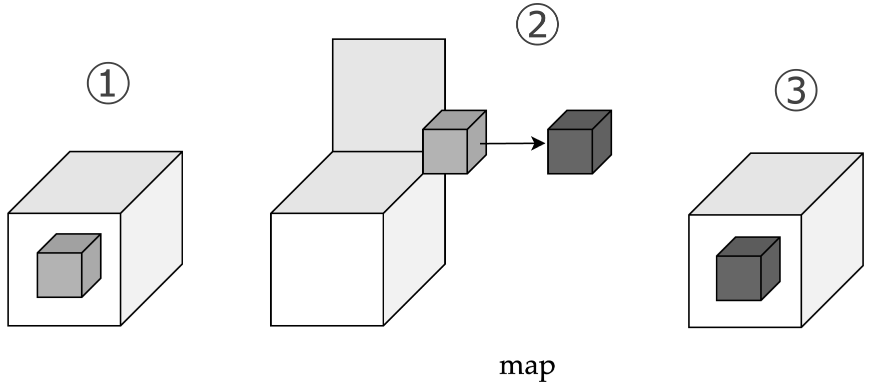


We assume grounding as one of the linguistic manifestations of **unit** in the monadic structure of linguistic discourse .

Grounding is "**a semantic function that constitutes the final step in the formation of a nominal or a finite clause.**"

(1) a. *Jennifer notice wall need new coat of paint
b. Jennifer notic**ed** that **the** wall need**s a** new coat of paint**.**
c. This wall need**s a** new coat of paint.

Monadic Operation: *Map*



The shift in narrative style in the development of discourse can be considered as a manifestation of the **map** operation.

You can **detach** the value from its environment and get it back to it again.

(2) "Think about it, and you'll figure it out," Ao said, finally.

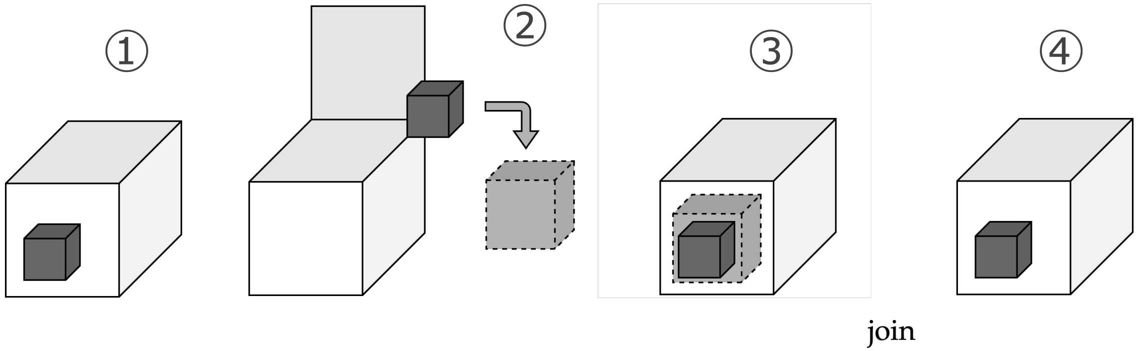
Tsukuru was speechless.

What was he talking about? Think about it? Think about what? If I think any harder about anything, I won't know who I am anymore.

"It's too bad it turned out like this," Ao said.

Murakami, Haruki. *Colorless Tsukuru Tazaki and His Years of Pilgrimage* (p.28)

Monadic Operation: *Join*



The grounding structure could easily **multi-layered** when we quote, describe past events, or narrate a story.

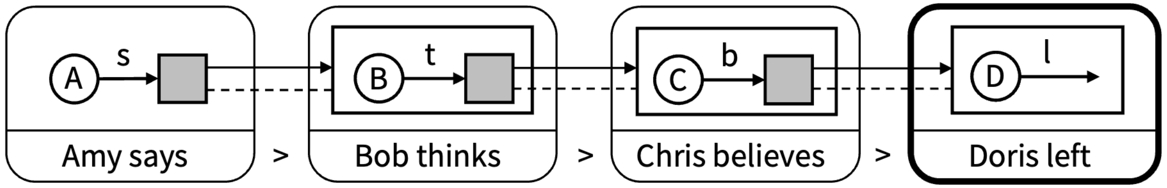
While maintaining this multi-layered structure as contextual knowledge, the subsequent discourse must unfold on a **flattened** structure.

(3) “That’s all he told you?” Sara asked.

“It was a short conversation, minimalist. That’s the very best I can reproduce it.”

Murakami, Haruki. *Colorless Tsukuru Tazaki and His Years of Pilgrimage* (p.29)

Sentences like “Amy says Bob believes Chris thinks Doris left” are not often actually spoken.



Monadic Chat

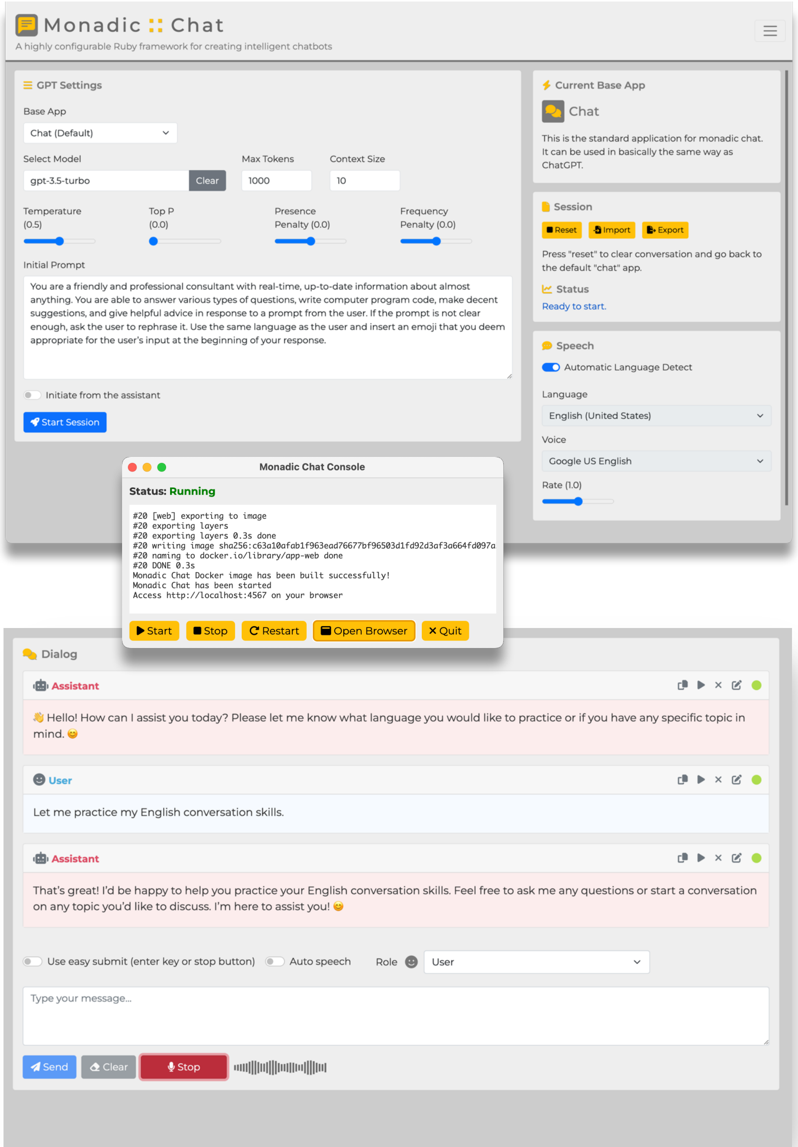
There are practical advantages to redefining **the CDS as a monadic structure**.

This idea can be used as a **design pattern** to implement a chatbot application like ChatGPT.

Hasebe (2023) developed **Monadic Chat**, a framework to provide an interactive interface to conduct a natural language conversation with AI, using the GPT text completion API of OpenAI.

https://yohasebe.github.io/monadic-chat

It offers unique **accumulator** and **reducer** mechanisms, with the accumulator storing previous utterances while the reducer helps manage the amount and composition of the accumulator.



Monadic :: Chat

A highly configurable Ruby framework for creating intelligent chatbots

GPT Settings

Base App: Chat (Default)

Select Model: gpt-3.5-turbo

Max Tokens: 1000

Context Size: 10

Temperature (0.5)

Top P (0.0)

Presence Penalty (0.0)

Frequency Penalty (0.0)

Initial Prompt

You are a friendly and professional consultant with real-time, up-to-date information about almost anything. You are able to answer various types of questions, write computer program code, make decent suggestions, and give helpful advice in response to a prompt from the user. If the prompt is not clear enough, ask the user to rephrase it. Use the same language as the user and insert an emoji that you deem appropriate for the user's input at the beginning of your response.

Current Base App

Chat

This is the standard application for monadic chat. It can be used in basically the same way as ChatGPT.

Session

Press "reset" to clear conversation and go back to the default "chat" app.

Status

Ready to start.

Speech

Automatic Language Detect

Language: English (United States)

Voice: Google US English

Rate (1.0)

Monadic Chat Console

Status: Running

```
#20 [web] exporting to image
#20 exporting layers
#20 exporting layers 0.3s done
#20 writing image sha256:c63a10afab1f963ead76677bf96503d1fd92d3af3a664fd097a
#20 naming to docker.io/library/app-web done
#20 DONE 0.3s
Monadic Chat Docker image has been built successfully!
Monadic Chat has been started
Access http://localhost:4567 on your browser
```

Dialog

Assistant

👋 Hello! How can I assist you today? Please let me know what language you would like to practice or if you have any specific topic in mind. 😊

User

Let me practice my English conversation skills.

Assistant

That's great! I'd be happy to help you practice your English conversation skills. Feel free to ask me any questions or start a conversation on any topic you'd like to discuss. I'm here to assist you! 😊

Use easy submit (enter key or stop button)

Auto speech

Role: User

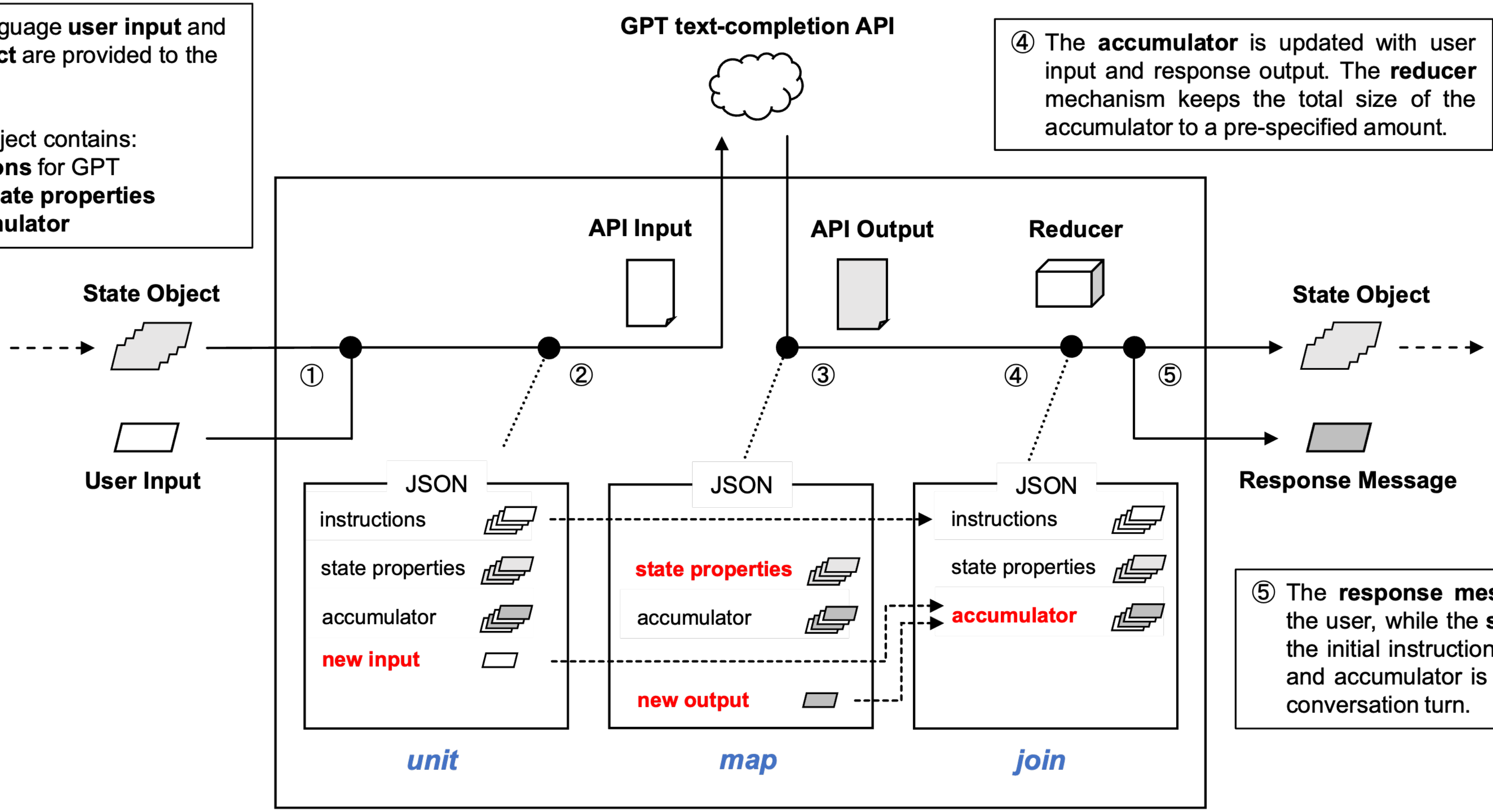
Type your message...

Architecture of Monadic Chat

① A natural language **user input** and a **state object** are provided to the system.

The **state** object contains:

a) **instructions** for GPT
b) various **state properties**
c) an **accumulator**



④ The **accumulator** is updated with user input and response output. The **reducer** mechanism keeps the total size of the accumulator to a pre-specified amount.

⑤ The **response message** is presented to the user, while the **state object** containing the initial instructions, the state properties, and accumulator is passed on to the next conversation turn.

② The user input and the properties of the state object are merged into a JSON object and sent to the **GPT text-completion API**.

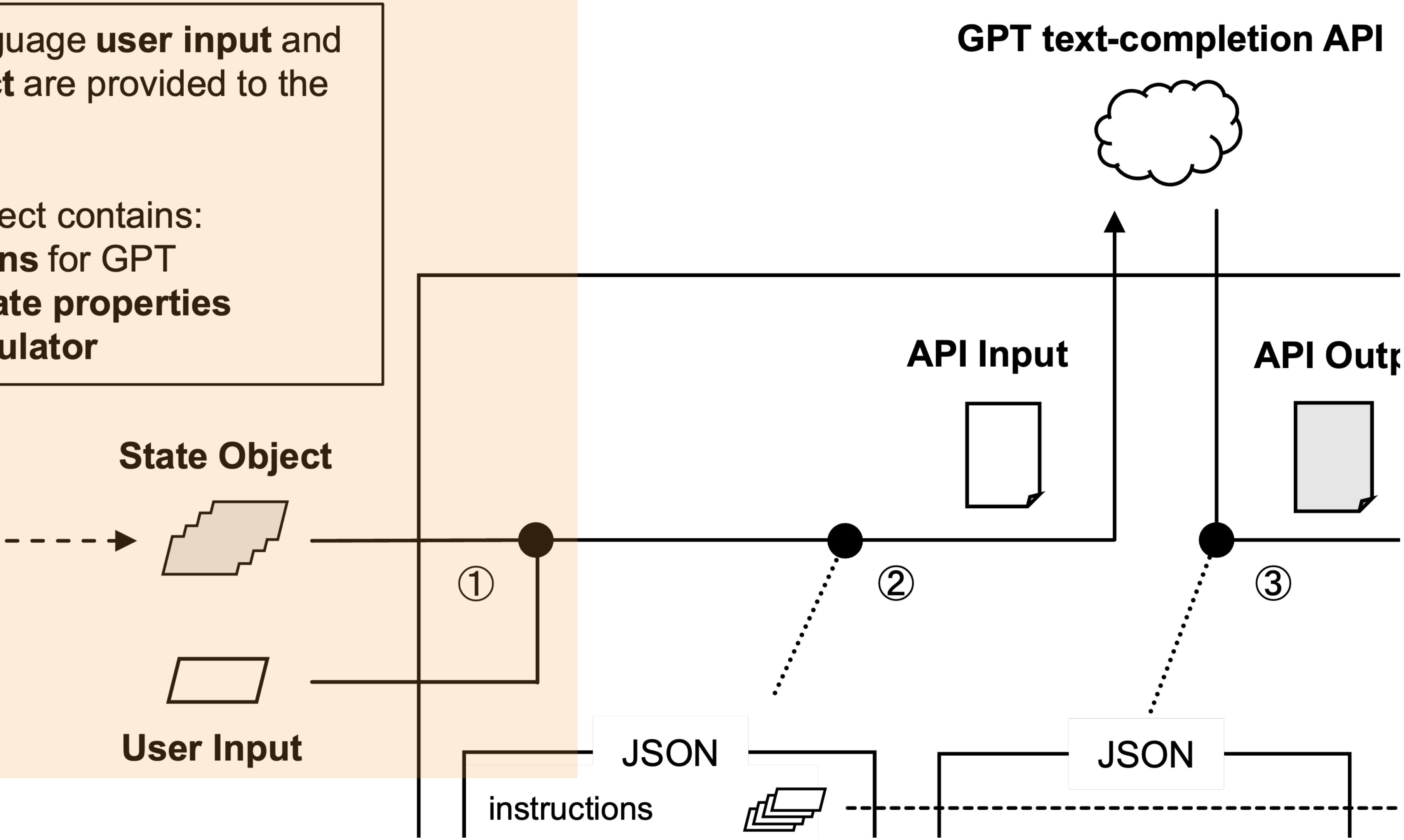
③ The API responds with a JSON object that contains a natural language **response output** and the **state properties** appropriately updated.

Architecture of Monadic Chat

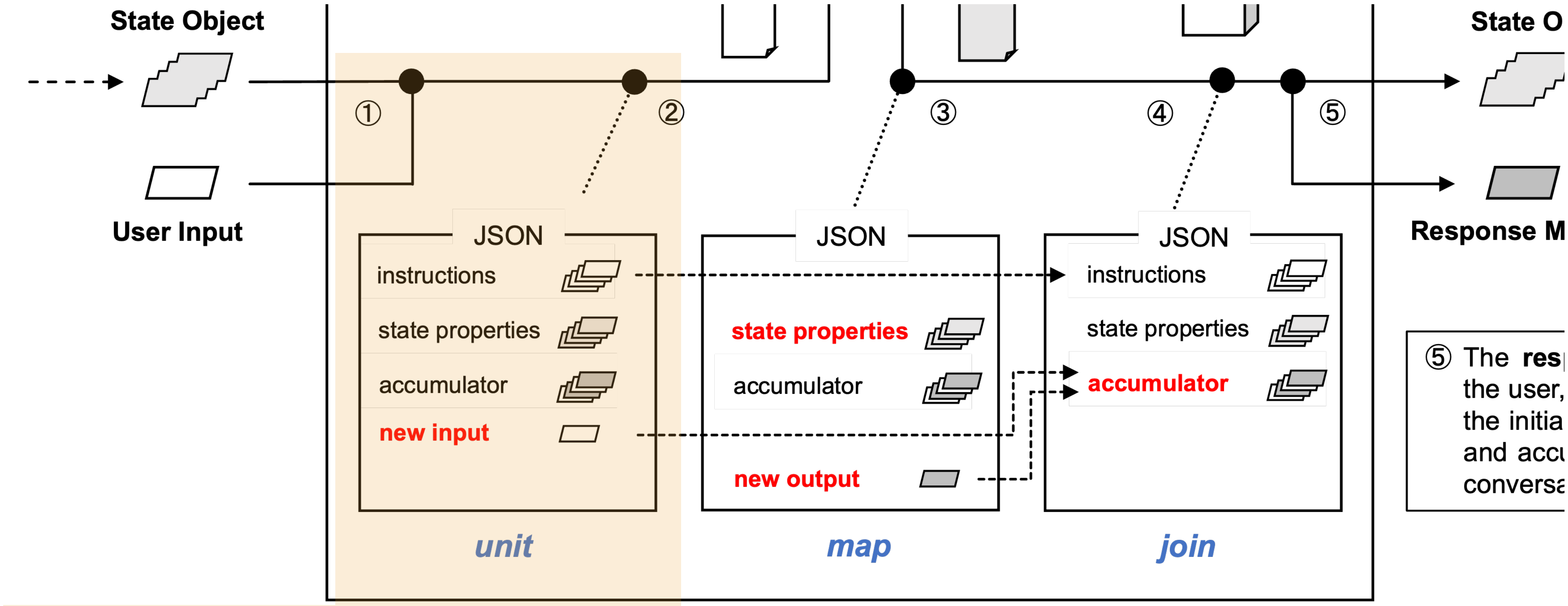
① A natural language **user input** and a **state object** are provided to the system.

The **state** object contains:

a) **instructions** for GPT
b) various **state properties**
c) an **accumulator**



Architecture of Monadic Chat

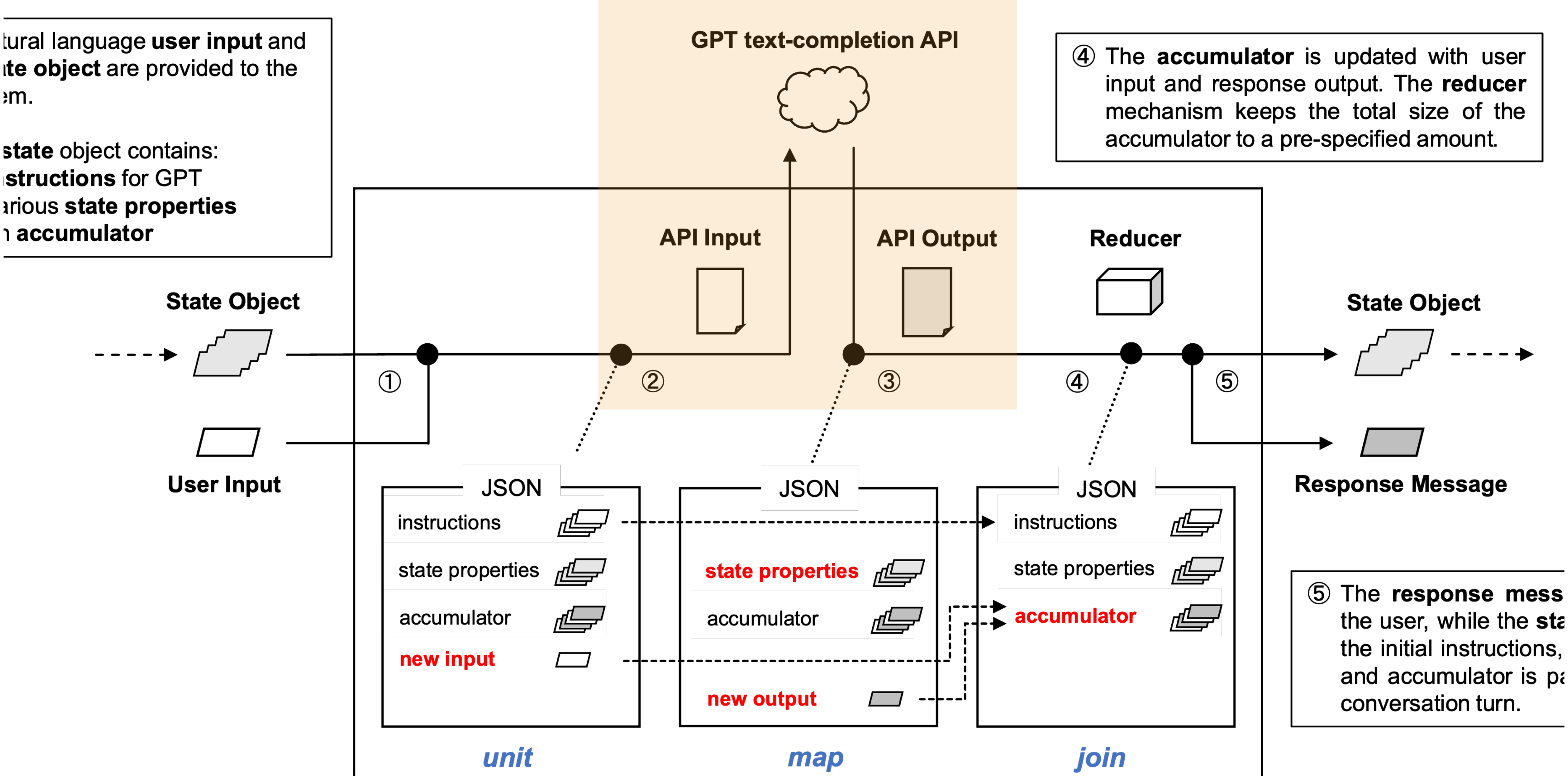


⑤ The res
the user,
the initia
and accu
conversa

② The user input and the properties of the state object are merged into a JSON object and sent to the **GPT text-completion API**.

③ The API responds with a JSON object that contains a natural language **response output** and the **state properties** appropriately updated.

Architecture of Monadic Chat



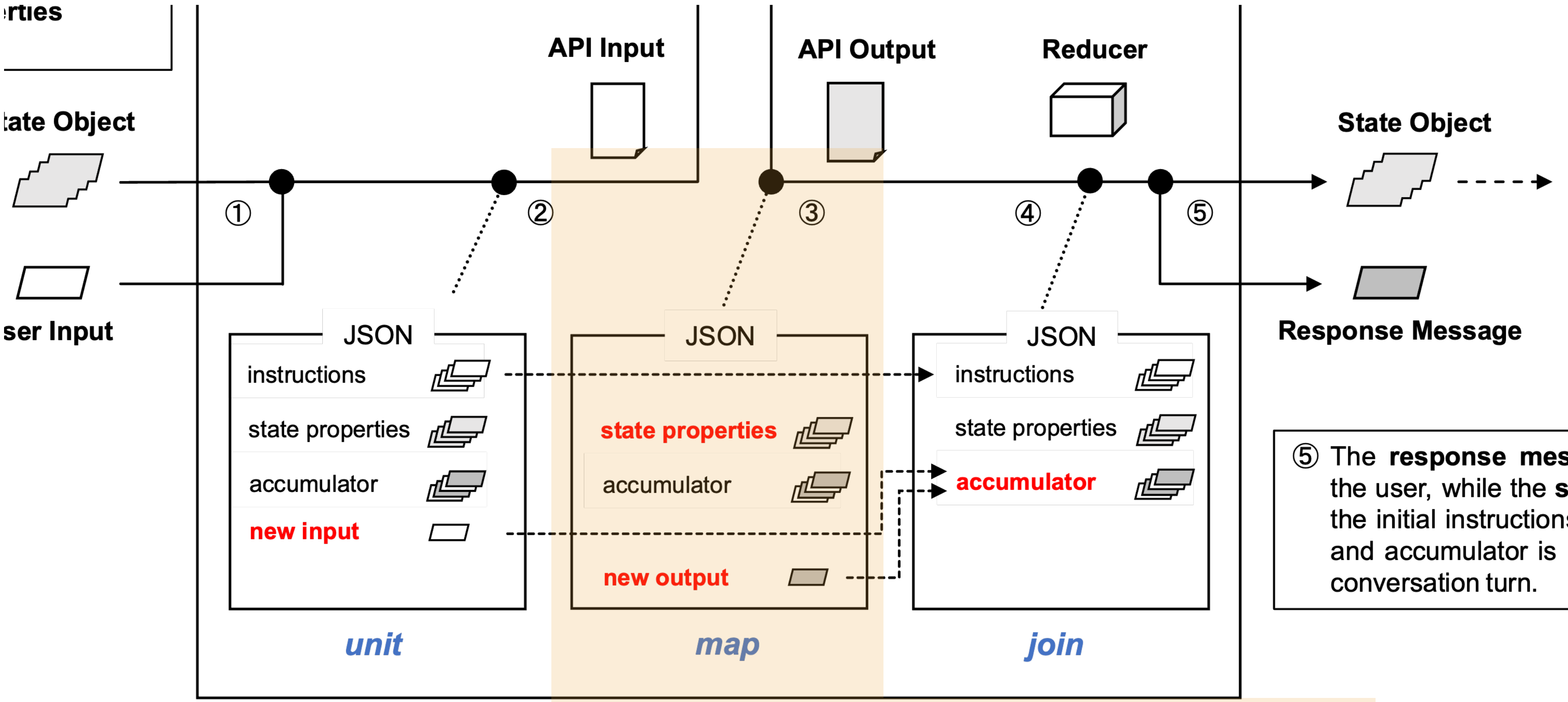
tural language **user input** and
te object are provided to the
em.

state object contains:
structions for GPT
arious **state properties**
accumulator

④ The **accumulator** is updated with user input and response output. The **reducer** mechanism keeps the total size of the accumulator to a pre-specified amount.

⑤ The **response mess**
the user, while the **sta**
the initial instructions,
and accumulator is pa
conversation turn.

Architecture of Monadic Chat

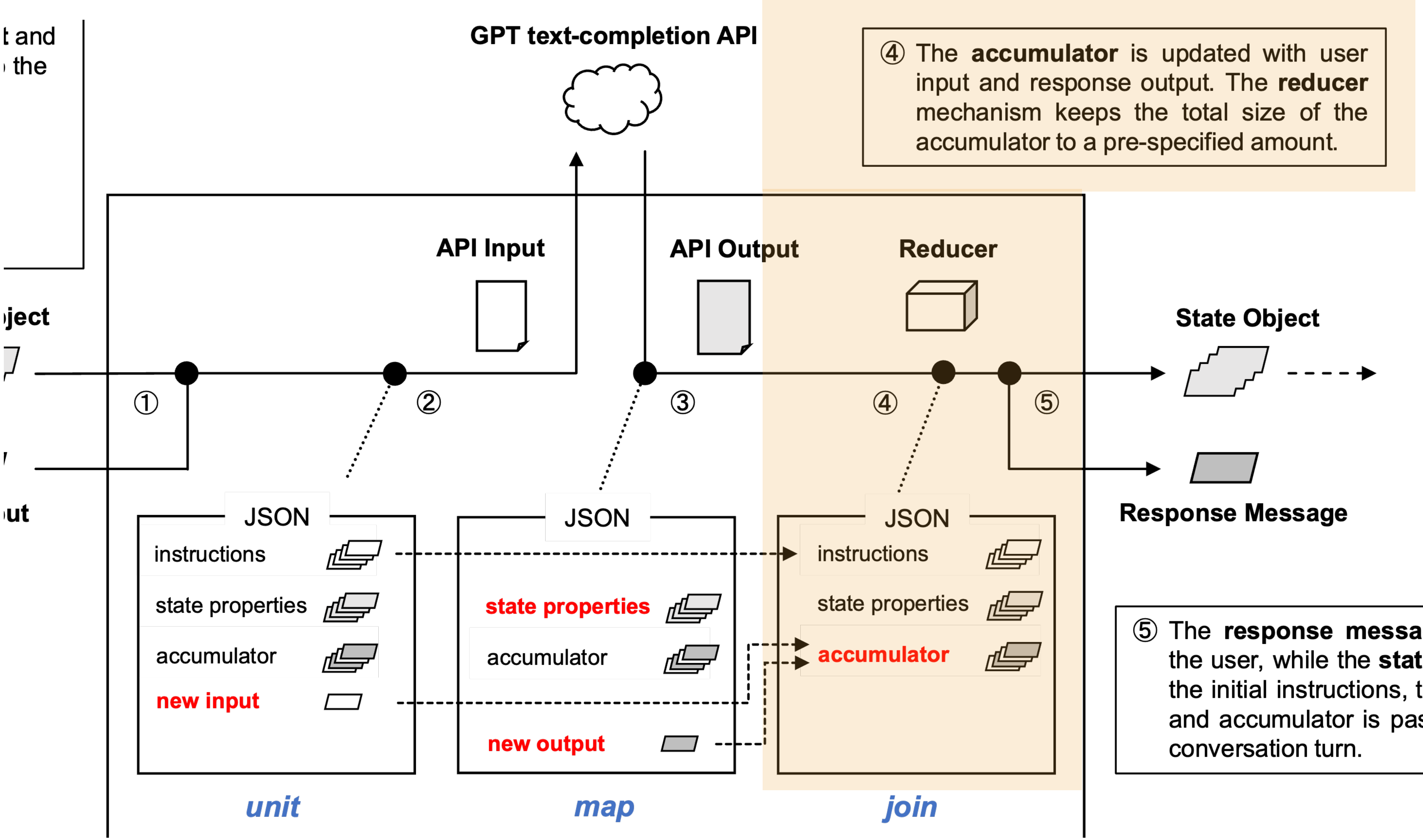


⑤ The **response message** is presented the user, while the **state object** containi the initial instructions, the state properti and accumulator is passed on to the n conversation turn.

user input and the properties of the state ct are merged into a JSON object and sent ie **GPT text-completion API**.

③ The API responds with a JSON object that contains a natural language **response output** and the **state properties** appropriately updated.

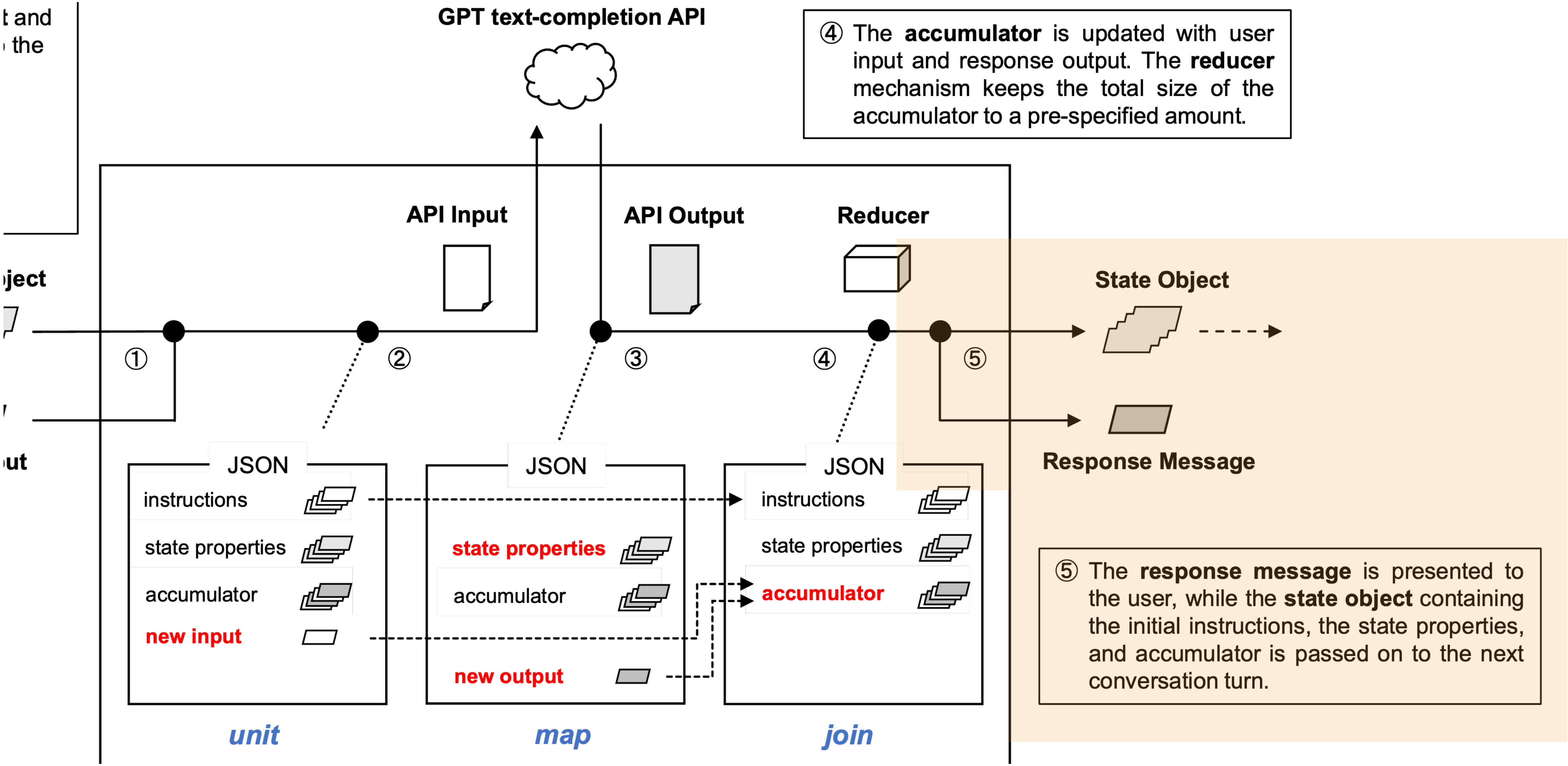
Architecture of Monadic Chat



④ The **accumulator** is updated with user input and response output. The **reducer** mechanism keeps the total size of the accumulator to a pre-specified amount.

⑤ The **response message** is presented to the user, while the **state object** containing the initial instructions, the state properties, and accumulator is passed on to the next conversation turn.

Architecture of Monadic Chat



④ The **accumulator** is updated with user input and response output. The **reducer** mechanism keeps the total size of the accumulator to a pre-specified amount.

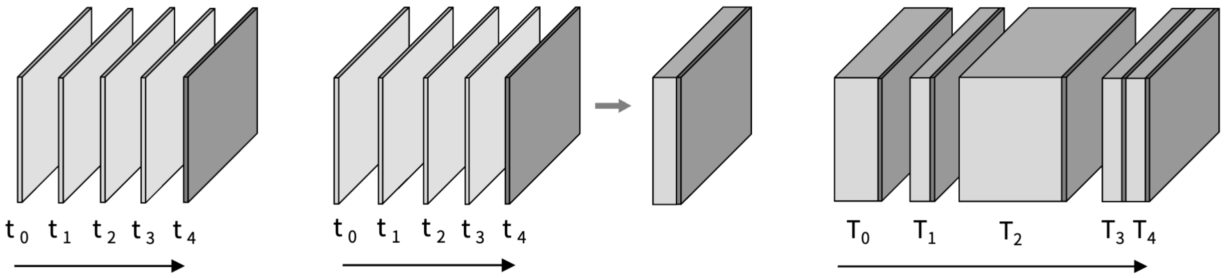
⑤ The **response message** is presented to the user, while the **state object** containing the initial instructions, the state properties, and accumulator is passed on to the next conversation turn.

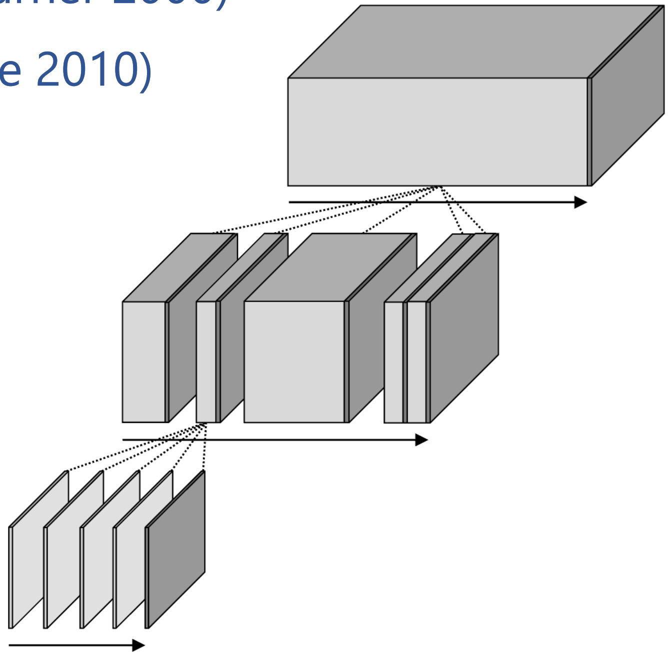
Accumulator and Reducer

This is somewhat speculative . . . but

accumulator and **reducer** mechanisms may allow for computational experimentation of processes such as:

- Incremental context building (Harder 1996; Langacker 2008)
- Stack-based item replacement (Chafe 1994)
- Compression and abstraction of concepts (Fauconnier & Turner 2000)
- Exemplification from past experience (Barsalou 2005; Bybee 2010)


t_0 t_1 t_2 t_3 t_4
t_0 t_1 t_2 t_3 t_4
T_0 T_1 T_2 T_3 T_4



Summary

- Langacker's **current discourse space (CDS)** model can be considered **monadic** in nature.
- Monad has a **mathematical/computational background** as many widely-used cognitive linguistic data structures are.
- If a structure is monadic, the three operations, ***unit***, ***map***, and ***join*** must be available.
- **Grounding**, as a semantic function that constitutes a final step in the nominal/clausal formation, may be an essential component of the **monadic structure of discourse**.
- The monadic structure naturally fits the architecture of a **chatbot application**.
- Monadic Chat is such an app utilizing OpenAI text/chat completion API equipped with **accumulator** and **reducer** mechanisms.
- Accumulator and reducer mechanisms may offer a **testing ground** for experimenting validity or degrees of significance such cognitive linguistic concepts and notions proposed so far.

References

Barsalou, Laurence. W. 2005. Abstraction as dynamic interpretation in perceptual symbol systems. In Lisa Gershkoff-Stowe & David H. Rakison (eds.), *Building object categories*, 389–431. Mahwah, NJ: Lawrence Erlbaum.

Bybee, Joan. 2010. *Language, usage and cognition*. Cambridge: Cambridge University Press.

Chafe, Wallace L. 1994. *Discourse, consciousness, and time: The flow and displacement of conscious experience in speaking and writing*. Chicago: University of Chicago Press.

Fauconnier, Gilles & Mark Turner. 2000. Compression and global insight. *Cognitive Linguistics* 11(3-4). 283–304.

Hasebe. 2021. An integrated approach to discourse connectives as grammatical constructions. Kyoto: Kyoto University PhD dissertation.

Hasebe. 2023. Monadic Chat: Framework for managing context with text completion API. *Proceedings of the 29th annual meeting of the Association for Natural Language Processing*, 3138-3143.

Harder, Peter. 1996. *Functional semantics: A theory of meaning, structure and tense in English*. Berlin: Mouton de Gruyter.

Hutton, Graham. 2016. *Programming in Haskell*, 2nd edn. Cambridge: Cambridge University Press.

Langacker, Ronald W. 2001. Discourse in cognitive grammar. *Cognitive Linguistics* 12(2). 143–88.

Langacker, Ronald W. 2008. *Cognitive grammar: A basic introduction*. Oxford: Oxford University Press.

Langacker, Ronald W. 2012. Interactive cognition: Toward a unified structure, processing, and discourse. International Journal of *Cognitive Linguistics* 3(2). 95–125.

Langacker, Ronald W. 2021. Functions and Assemblies. In Kazuhiro Kodama and Tetsuharu Koyama, eds., *The Forefront of Cognitive Linguistics*, 1–54, Tokyo: Hituzi Shobo.

Petricek, Tomas. 2018. What we talk about when we talk about monads. *The Art, Science, and Engineering of Programming* 2(3), https://programming-journal.org/2018/2/12/. (1 January 2023.)

Wadler, Philip. 1995. Monads for functional programming. In Rogardt Heldal, Carsten Kehler Holst & Philip Wadler (eds.), *Advanced functional programming*, 24–52. Dordrecht: Springer.

Handout PDF