ChatGPT

Wprowadzenie

Projekt **TITANS** (Transformer-based Inference-Time Augmented Neural System) to wielomodułowa architektura kognitywna zaprojektowana do ciągłego uczenia się i **autonomicznego rozwoju**. Składa się z serii modułów (M1–M8), które realizują kolejne etapy przetwarzania od percepcji po **metakognicję**. W rozmowie (prowadzonej z asystentem "Gemini") przeanalizowano integrację modułów **M1–M5** (percepcja aż do rdzenia agenta) z nowo zaproponowaną pętlą **metakognitywną** obejmującą **M6–M8**. Poniżej przedstawiam analizę tej rozmowy oraz repozytorium projektu, ze szczególnym naciskiem na proponowaną **integrację modułów**, zidentyfikowane **braki logiczne**, oraz rozróżnienie między stwierdzeniami pewnymi (Fakty - **F**) a spekulatywnymi (Hipotezy - **H**).

Architektura TITANS (M1–M8)

Milestone 1: Percepcja (CapsNet). Moduł percepcyjny używa **sieci kapsułkowych** do ekstrakcji obiektowo-centrycznych reprezentacji ze strumienia danych sensorycznych [1]. Jest to **podstawa** systemu – konwolucyjna sieć kapsuł generuje **„percepty"** (wektory kapsuł) opisujące wykryte obiekty cechami posegregowanymi hierarchicznie. *F:* Sieci kapsułkowe Geoffreya Hintona potrafią modelować hierarchie części-całość i relacje przestrzenne, dając bogatsze reprezentacje niż zwykłe konwolucje [2]. Implementacja M1 w projekcie to np. plik `capsnet_m1.py`, zawierający warstwy **PrimaryCaps**, **DigitCaps** i funkcję straty **margin_loss** dla zbioru MNIST [3].

Milestone 2: Pamięć Epizodyczna (STM/LTM). Ten moduł odpowiada za **dwuwarstwową pamięć** krótkotrwałą i długotrwałą, zdolną do zapisywania nowych „doświadczeń" (wektorów z percepcji) oraz ich **konsolidacji** w tle [4]. *F:* Zastosowano tu mechanizm **generative replay** – **wariacyjny autoenkoder (VAE)** konsoliduje zbiory epizodów, generując ich warianty ("dreams") celem uogólnienia wiedzy [5] [6]. Jest to inspirowane pracami nad długotrwałą pamięcią dla modeli językowych i uczeniem się w trakcie inferencji [7]. W kodzie przewidziano klasę `EpisodicLTM` z metodami dodawania doświadczeń i cyklem konsolidacji korzystającym z VAE [8] [9]. *H:* Skuteczność takiego "śnie o danych" (dreaming) w poprawianiu uogólnienia modelu pozostaje do zweryfikowania – jest to pomysł oparty na literaturze, ale wymaga empirycznego potwierdzenia w ramach TITANS.

Milestone 3: Abstrakcja Semantyczna. Gdy epizody zbierają się w pamięci, powtarzające się wzorce są destylowane w **stabilne, abstrakcyjne koncepty** [10]. Moduł M3 wykorzystuje najpewniej **Transformera** do agregacji wielu podobnych epizodów w jeden reprezentant – uogólniony **wektor pojęciowy**. *H:* W projekcie założono, że Transformer poradzi sobie z wyłuskaniem powtarzalnych wzorców i utworzeniem abstraktów semantycznych, co przypomina działanie kory nowej w mózgu. Jednak brak jeszcze implementacji tego modułu w kodzie – jest opisany koncepcyjnie [11]. Logicznie pojawia się pytanie: **jak dokładnie moduł M3 wybiera i agreguje epizody?** Na ten moment jest to ogólna idea oparta na literaturze, a nie skonkretyzowany algorytm (czy użyty będzie np. **Transformer encoder** na sekwencjach epizodów, nie sprecyzowano). Zatem moduł M3 to na razie **hipoteza projektowa** oparta na analogii do mechanizmów abstrakcji w ludzkiej pamięci.

Milestone 4: Rozumowanie Semantyczne. To silnik **wnioskowania** operujący na grafie wiedzy utworzonym z konceptów semantycznych. Plan zakłada użycie **Grafowych Sieci Neuronowych** (np. Graph Attention Network, GAT) do propagacji informacji między węzłami konceptów i odkrywania relacji [12]. *F:* Projekt implementuje moduł M4 jako **Graph Attention Network** (`ReasoningGAT`) zdolny do

wieloetapowego wnioskowania na grafie pojęć [13]. W kodzie (wg dokumentacji) przewidziano też prosty **EdgePredictor** do przewidywania nowych relacji między konceptami na podstawie wektorów tych konceptów [14]. *H:* Jednak obecna implementacja M4 posługuje się **danymi syntetycznymi** – losowo generowanymi wektorami konceptów i prostym wyliczaniem relacji jako różnicy wektorów [15]. To pokazuje, że brak tu integracji z faktycznym modułem M3 (nie ma prawdziwego grafu wiedzy z abstrakcji semantycznej). Założenie, że GNN poradzi sobie z **rozumowaniem analogicznym do ludzkiego** na tak utworzonym grafie, jest dopiero do sprawdzenia. Póki co moduł ten działa w izolacji (testy M4 są na sztucznym grafie) i stanowi **dowód koncepcji**, a nie sprawdzony system rozumujący.

Milestone 5: Rdzeń Agentowy (Agency Core). Jest to **centralna pętla decyzyjna** agenta, nadająca systemowi zdolność autonomicznego podejmowania **"działań kognitywnych"** w celu redukcji wewnętrznej niepewności [16] [17]. M5 korzysta z zasad **Bayesowskiego Uczenia przez Wzmocnienie (Bayesian RL)** – zaimplementowano tu mechanizm wewnętrznej nagrody zależnej od **wariancji zespołu krytyków** (krytycy oceniają stan wiedzy) [18]. *F:* Zmniejszanie wariancji oceny stanu (tj. zmniejszanie niepewności) jako nagroda jest inspirowane pracą naukową (Bayesian Bellman Operators) i zostało zaproponowane jako **metoda metakognitywna** w architekturze [17]. W kodzie M5 (`AgenticCore`) widać to wyraźnie: po wykonaniu działania, obliczana jest różnica wariancji wartości stanu przed i po – ta różnica (spadek niepewności) staje się nagrodą wewnętrzną [19] [20]. Agent ma politykę (aktor) wybierającą **działania poznawcze** oraz ensemble krytyków oceniających stany [21]. *H:* Przyjęto, że agent będzie dzięki temu **sam kierował swoim uczeniem** – np. wybierze akcję dodania nowego konceptu lub wzmocnienia relacji by zmaksymalizować redukcję niepewności [22]. To odważne założenie. Dotychczasowe testy są uproszczone (w symulacji agent losowo dodaje fikcyjne koncepty do grafu) [23] [24]. Nie wiadomo, jak **skala** tego rozwiązania zadziała w realnym scenariuszu z prawdziwymi danymi i pełnym grafem wiedzy – czy polityka nauczy się sensownych działań, czy niepewność faktycznie koreluje z postępem w rozumieniu.

Milestone 6: Aksjomatyczny Model Siebie (Self-Model). Tutaj rozpoczyna się **meta-poziom** architektury. M6 to moduł autorefleksji, który ma utrzymywać **zbiór aksjomatów** – fundamentalnych założeń operacyjnych systemu – i sygnalizować ich **naruszenie** [25]. Innymi słowy, M6 monitoruje, czy nowa wiedza lub stany systemu nie przeczą **podstawowym regułom** (samoświadomości agent może mieć np. aksjomat „moja ontologia jest spójna" albo „nie sprzeciwiaj się logice" itp.). *H:* Koncepcja ta jest na razie czysto teoretyczna – projekt nie precyzuje, **jak zdefiniowane są te aksjomaty ani w jaki sposób moduł wykrywa sprzeczność**. Można się domyślać, że M6 będzie analizować graf wiedzy z M4/M5 pod kątem sprzecznych wniosków lub monitorować parametry modelu (np. czy rozkłady prawdopodobieństwa nie sumują się do >1, itp.). Jednak brak implementacji w kodzie (pliku `m6_self_model.py` jeszcze nie analizowaliśmy, ale zapewne zawiera szkic) wskazuje, że to **hipoteza**: że taki mechanizm samo-nadzoru można zaszyć w systemie. *F:* Faktem jest natomiast, że podobne idee pojawiają się w literaturze (np. **model światopoglądu** agenta, wykrywający anomalia) – ich skuteczność w praktyce to osobne zagadnienie badawcze.

Milestone 7: Emocjonalna Empatia (Affective Empathy Resonator). M7 stanowi drugi „ciśnieniowy" sygnał meta-poziomu. Opisany został jako generator sygnału **"cierpienia"** lub dysonansu, który wskazuje na stan niepożądany dla systemu [25]. W kontekście TITANS może to oznaczać wskaźnik frustracji poznawczej – np. rosnącej niepewności, błędu przewidywań lub braku postępów. *H:* Nazwa sugeruje inspirację **empatią/afektem**, choć w systemie sztucznym oznacza to prawdopodobnie **metrykę niezadowolenia** systemu z własnych wyników. To **hipoteza**, że dodanie takiego sygnału (analogu emocji) wspomoże proces samodoskonalenia. W praktyce M7 mógłby być zrealizowany np. jako pomiar częstotliwości nieudanych wniosków M4 lub małych spadków niepewności w M5 (jeśli agent długo nie poprawia swojej sytuacji, generuje się "cierpienie"). Jednak **nie ma pewnych podstaw naukowych**, by stwierdzić, jak ująć "cierpienie" maszyny – to raczej element filozofii projektu (nawiązanie do potrzeb/woli przetrwania systemu), który wymaga daleko idących założeń.

Milestone 8: Autopoietyczny Silnik (Autopoiesis). M8 to kulminacja meta-pętli: moduł, który na podstawie sygnałów z M6 (naruszone aksjomaty) i M7 (cierpienie) dokonuje **przepisania własnej ontologii** oraz tworzy nowe pojęcia/pośredniczący język (**Efficon**) dla dalszego rozwoju [26] [25]. Innymi słowy, kiedy system wykryje, że jego aktualny model świata *nie pozwala* rozwiązać sprzeczności lub redukować cierpienia, M8 zmienia **rdzeń wiedzy**: dodaje/usuwa aksjomaty, redefiniuje pojęcia albo wprowadza **nowe koncepty pośrednie** (Efficony) umożliwiające komunikację i myślenie o trudno uchwytnych zjawiskach. *H:* Jest to najbardziej **spekulacyjny** element projektu. Idea autopoiesis wywodzi się z biologii (zdolność systemu do samostanowienia i samoodtwarzania). Tutaj zakłada się, że agent może **samodzielnie modyfikować własne struktury poznawcze** w sposób celowy. To ambitna hipoteza – brak precedensu w obecnej AI, by system dynamicznie zmieniał swoją ontologię i język bez utraty tożsamości. W rozmowie Gemini uznano, że ta integracja M6–M8 tworzy **"minimalny, ale kompletny obwód metakognitywny"** spełniający filozofię TITANS [26]. Jednak trzeba zauważyć brak konkretów: **jak M8 ma zmieniać ontologię?** Czy dodawanie Efficonów (np. nowych symboli w grafie wiedzy) wystarczy, by rozwiązać sprzeczności? Jak upewnić się, że zmiany nie zepsują dotychczasowych modułów? Te pytania pozostają otwarte.

Integracja modułów M1–M5 z pętlą M6–M8

Integracja oznacza połączenie **głównej pętli poznawczej** (M1–M5) z **meta-pętlą** autorefleksji (M6–M8) w jeden spójny system. Z analizy rozmowy i repozytorium wynika, że kluczową rolę ma tu pełnić warstwa **orchestracji**, buforowania sygnałów i inicjalizacji całego cyklu. Zgodnie z ustaleniami z rozmowy, dokonano refaktoryzacji kodu prototypowego do ustandaryzowanego pakietu `titans-core` [27]. Poniżej opisuję proponowane rozwiązanie integracyjne oraz aktualizacje struktury projektu:

- **Struktura repozytorium.** Repozytorium `titans-core` ma zostać zorganizowane modułowo z podziałem na komponenty M1–M8. Rozmowa Gemini przedstawiła szkic struktury, który uwzględnia już pliki dla M6, M7, M8 oraz miejsce na wcześniejsze moduły. Zaproponowana hierarchia katalogów wygląda następująco [28] [29]:

```
titans-core/
├── .github/workflows/ci.yml          # Konfiguracja CI (testy,
formatowanie)
├── notebooks/
│   └── M8_Autopoiesis_Loop_Demo.ipynb  # Interaktywny notebook
demonstracyjny pętli M6–M8
├── src/
│   └── titans_core/
│       ├── __init__.py
│       ├── common.py            # Wspólne struktury (np. RollingBuffer,
EfficonMessage) [30] [31]
│       ├── perception/          # **M1**: percepcja (CapsNet)
│       │   ├── __init__.py
│       │   └── capsnet.py       # Implementacja sieci kapsułkowej (M1)
│       ├── memory/              # **M2**: pamięć epizodyczna
│       │   ├── __init__.py
│       │   ├── ltm.py           # Pamięć długotrwała + generative replay
(VAE) (M2)
│       │   └── vae.py           # Wariacyjny autoenkoder do konsolidacji
(M2)
```

```
│       ├── abstraction/          # **M3**: abstrakcja semantyczna
│       │   ├── __init__.py
│       │   └── transformer.py    # Model Transformer destylujący koncepty
(M3)
│       ├── reasoning/            # **M4**: rozumowanie semantyczne
│       │   ├── __init__.py
│       │   ├── gnn.py            # Graph Neural Network (GAT) do
wnioskowania (M4)
│       │   └── main.py           # Skrypt treningowy/ewaluacyjny M4 [32]
│       ├── core/                 # **M5**: rdzeń agentowy
│       │   ├── __init__.py
│       │   ├── agent.py          # Implementacja AgenticCore i pętli
decyzyjnej (M5) [33]
│       │   └── models.py         # Definicje modeli aktora i krytyków (M5)
[33]
│       ├── m6_self_model.py      # **M6**: moduł samo-modelu aksjomatycznego
│       ├── m7_empathy.py         # **M7**: moduł rezonatora empatii (sygnał
cierpienia)
│       ├── m8_autopoiesis.py     # **M8**: silnik autopoietyczny (Ontology,
MetaRules, M8Engine) [29]
│       └── orchestration/
│           ├── __init__.py
│           ├── bootstrap_config.py  # Konfiguracja startowa całego rdzenia
(parametry, flagi)
│           └── orchestrator.py     # Orkiestrator pętli M1–M8 (kolejkuje
operacje, buforuje sygnały)
├── tests/
│   └── test_core_loop.py         # Testy integracyjne głównej pętli
poznawczej i meta-pętli [34]
├── .gitignore
├── pyproject.toml                # Konfiguracja pakietu (Python, zależności)
└── README.md                     # Opis projektu, instrukcje uruchomienia
```

Komentarz: Powyższa struktura łączy wcześniejsze prototypy (M1–M5 były rozproszone w dokumentacji i osobnych skryptach) z nowym kodem M6–M8 w **jednolitym pakiecie**. Dodano też infrastrukturę DevOps (plik CI, konfiguracja pyproject) pod kątem profesjonalizacji projektu [27]. Dzięki temu wszystkie moduły znajdą się w jednym repozytorium, co ułatwi wersjonowanie i testowanie całości.

- **Rola modułu** `common.py`. W pliku `common.py` zebrane zostaną **wspólne struktury danych i narzędzia** używane między modułami. Z rozmowy wynika, że zdefiniowano tam klasę **RollingBuffer** oraz **EfficonMessage** [30] [31]. RollingBuffer prawdopodobnie służy do gromadzenia sygnałów z M6 i M7 – działa jak **bufor doświadczeń meta-poziomu**. Będzie on okresowo odczytywany przez M8 (autopoiesis) w poszukiwaniu wzorców wymagających zmiany ontologii. Z kolei EfficonMessage to zapewne struktura reprezentująca nowo wygenerowany koncept/"słowo" (Efficon) wraz z kontekstem – rodzaj komunikatu, który M8 przekazuje z powrotem do reszty systemu (np. „stworzono nowy koncept X znaczący Y"). Ujęcie tych elementów w `common.py` zapobiega duplikacji kodu i zapewnia spójność komunikacji między modułami meta-pętli.

- **Orkiestracja cyklu poznawczego.** Dodanie katalogu `orchestration/` sugeruje istnienie centralnego **orkiestratora** całego procesu. Plik `orchestrator.py` będzie uruchamiał moduły w odpowiedniej kolejności i warunkach. Przykładowo, orchestrator może wykonywać pętlę: (1) **Percepcja** pobiera dane i tworzy percepty (M1), (2) dodaje je do **STM/LTM** (M2), (3) okresowo uruchamia **konsolidację** i **abstrakcję** (M2→M3) w tle, (4) utrzymuje aktualny **graf wiedzy** dla **rozumowania** (M4), (5) na tej podstawie pozwala agentowi podejmować kroki poznawcze (M5), i jednocześnie (6) **monitoruje aksjomaty i stan afektywny** (M6, M7). Gdy sygnały M6/M7 przekroczą pewien próg lub spełnią warunek, orchestrator wyzwala **proces autopoietyczny** M8, który modyfikuje strukturę wiedzy lub dodaje nowe pojęcia. Następnie cykl wraca do obsługi percepcji, już z uaktualnionym "umysłem" agenta. Ten mechanizm zapewnia zamknięcie **pętli refleksyjnej** – agent nie tylko uczy się o świecie, ale też **koryguje samego siebie**.

- **Integracja sygnałów M6–M8 z M1–M5.** Sygnały z meta-poziomu muszą być powiązane z podstawową pętlą poznawczą. Konkretnie: moduł M6 będzie nasłuchiwał na **ontologię/stan wiedzy** tworzony przez M4/M5 – np. będzie miał dostęp do grafu konceptów i wyników wnioskowania. Jeśli wykryje sprzeczność (np. dwie koncepcje w grafie wzajemnie się wykluczają), umieści zdarzenie w **RollingBufferze** (np. obiekt `EfficonMessage` z typem "naruszony aksjomat X"). Podobnie M7 może monitorować przebieg pętli M5: np. spadające nagrody, rosnącą wariancję krytyków lub brak nowej wiedzy – i sygnalizować "cierpienie" również w buforze (np. wpis typu "frustration:+1"). M8 periodycznie odczytuje zawartość bufora i **decyduje o reakcji**. Integracja ta będzie wymagała jasno zdefiniowanych **interfejsów**: np. M6/M7 publikują zdarzenia przez wywołanie `common.RollingBuffer.push(event)`, zaś M8 dokonuje odczytu `events = buffer.consume()` i następnie używa ich do modyfikacji wiedzy. Tu ujawnia się rola **MetaRules** (w `m8_autopoiesis.py` zapewne) – zestawu reguł decyzji M8, np.: "jeśli wykryto aksjomat sprzeczny, dodaj nową hipotezę (Efficon) wyjaśniającą tę sprzeczność". *H:* Trzeba zaznaczyć, że sukces tej integracji zależy od **trafnego zaprojektowania tych reguł/metod** – co jest wyzwaniem i na razie istnieje jako projektowa intencja.

- **Notebook demonstracyjny.** Do repozytorium dodano notebook `M8_Autopoiesis_Loop_Demo.ipynb`, który pełni rolę **eksperymentalnego poligonu** oraz dokumentacji. Według rozmowy, notebook ten prezentuje narracyjnie i **weryfikowalnie** działanie pętli M6–M8 jako dowód dla potrzeb grantowych [35]. Zapewne zawiera on przykładowy scenariusz: sztucznie generowane naruszenie aksjomatu -> pojawienie się "cierpienia" -> uruchomienie M8 -> efekt w postaci zmiany (np. wypisanie komunikatu o utworzeniu Efficonu i zmodyfikowanej ontologii). Notebook umożliwia **interaktywny wgląd** w przebieg algorytmu, co jest cenne dla zaprezentowania idei recenzentom czy inwestorom. Integrując M1–M5, należałoby analogicznie poszerzyć zakres demonstracji o wcześniejsze moduły (np. pokazując pełen cykl od percepcji do autopoiesis). Jednak ze względu na złożoność, możliwe że na razie skupiono się na samym meta-poziomie, izolując go od reszty – co prowadzi nas do kolejnej sekcji o brakach.

Braki i niespójności logiczne w projekcie

Pomimo imponującej wizji, analiza dokumentacji i rozmowy ujawnia kilka **luk logicznych i spójnościowych**, które warto adresować:

- **Brak wyraźnego połączenia między pętlą poznawczą a meta-poziomem.** Choć struktura wskazuje, że M6 i M7 czerpią dane z "ontologii" i stanu agenta, **nie zdefiniowano formalnie interfejsu** między nimi a modułami M1–M5. Przykładowo: czy **graf wiedzy** budowany przez M4 (GNN) jest jawnie dostępny dla M6? Czy M7 monitoruje statystyki uczenia aktora/krytyków z M5? Na razie jest to implikowane, ale nie sprecyzowane. To luka do uzupełnienia, gdyż bez tego meta-

moduły mogą działać w próżni. *H:* Prawdopodobnie planuje się wykorzystać globalny obiekt reprezentujący **„pamięć poznawczą”** (np. `CognitiveGraphMemory` użyty testowo w kodzie M5 [36]), który będzie udostępniany również M6–M8. Należałoby to jasno zaprojektować (np. wzorzec **Blackboard** lub centralny stan w orchestratorze).

- **Niejasna definicja sygnału “cierpienia” i aksjomatów.** Wprowadzenie quasi-emocjonalnego sygnału cierpienia (M7) i zestawu aksjomatów (M6) to ciekawy pomysł, ale brakuje **formalizacji**. Jakie konkretnie aksjomaty posiada system? Czy są to prawa logiki, czy np. “moja pamięć nigdy nie traci danych”? Bez zdefiniowania tych aksjomatów, M6 nie ma czego naruszać. Podobnie, “poziom cierpienia” M7 – czy to będzie skalar od 0 do 1 zależny od entropii polityki agenta, czy licznik porażek w zadaniu? Bez takiej specyfikacji trudno przewidzieć zachowanie M8. Ten brak precyzji jest **logicznym niedopatrzeniem**, bo od niego zależy skuteczność całej pętli autopoietycznej. *H:* Być może zakłada się tu adaptacyjność – że system sam z czasem wypracuje swoje aksjomaty/emocje, co jednak odsuwa problem definicji na metapoziom i czyni go jeszcze trudniejszym.

- **Ryzyko sprzecznych lub zapętlających się modyfikacji w M8.** Skoro M8 może **modyfikować ontologię**, istnieje ryzyko, że zmiana ta spowoduje kolejne naruszenia aksjomatów lub niezamierzone skutki. Przykładowo, jeśli M8 “rozwiąże” sprzeczność dodając nowy koncept (Efficon), to ten koncept powinien być zrozumiały dla istniejących modułów (M3–M5). Jeśli nie, system może popaść w **niekończącą się pętlę** dodawania konceptów, próbując wyjaśnić własne wyjaśnienia. Logicznie projekt musi zatem określić **ograniczenia działań M8** – np. że Efficony muszą bazować na już istniejących pojęciach lub że pewne aksjomaty są nienaruszalne. Na razie nie widzimy takich ograniczeń opisanych, co stanowi lukę. *H:* To wskazuje, że M8 jest na etapie koncepcyjnym – jego algorytm (np. moduł **MetaRules** w `m8_autopoiesis.py`) wymaga jeszcze doprecyzowania i zbalansowania, by system nie zdestabilizował się wskutek własnych zmian.

- **Skalowalność i złożoność obliczeniowa.** Każdy moduł w osobna jest wymagający (CapsNet, VAE, Transformer, GNN – to wszystko ciężkie modele). Zintegrowanie ich oznacza potencjalnie ogromne zapotrzebowanie na moc obliczeniową oraz trudność w **strojeniu hyperparametrów** całości. Logicznie pojawia się pytanie, **czy system w praktyce może działać online**. Wymagania sprzętowe w dokumentacji sugerują potężne GPU i klastry [37], co potwierdza tę obawę. Ale poza wydajnością: jeśli np. CapsNet generuje setki wektorów percepcyjnych na sekundę, czy M2/M3 zdążą je konsolidować? Czy M4 zdąży zaktualizować graf przed kolejną porcją danych? Brak opisu mechanizmów **synchronizacji** czy **kolejkowania zadań** (poza wspomnianym RollingBuffer dla meta-sygnałów) – orchestrator będzie musiał to rozwiązać. To wyzwanie inżynieryjne, a obecny stan projektu skupia się na poprawności koncepcji, przez co pomija te szczegóły.

- **Ograniczone testowanie modułów w integracji.** Testy jednostkowe/integracyjne w repo (`test_core_loop.py`) prawdopodobnie dotyczą tylko pętli M6–M8 (lub M5–M8). W dokumentacji widzieliśmy test M5, gdzie agent w uproszczonym środowisku zmniejsza wariancję krytyków [38] [39]. Jednak brak testów łączących np. M1→M2→M3→M4 w sekwencji. To **luka**: nie sprawdzono dotąd, czy dane z percepcji rzeczywiście przechodzą poprawnie cały łańcuch aż do grafu i czy agent potrafi na nich operować. Integracja M1–M5 z meta-pętlą będzie jeszcze trudniejsza do przetestowania. Dla spójności logicznej projektu warto zaplanować **testy end-to-end** (np. symulacja prostego świata, gdzie agent obserwuje -> zapamiętuje -> abstrahuje -> wnioskuje -> podejmuje decyzje, a następnie meta-moduły korygują agentowi model). Obecny brak takich testów sprawia, że integracja opiera się bardziej na założeniach niż na empirycznych danych.

- **Równoległość vs sekwencyjność działań.** Architektura zakłada pętlę, ale czy wszystkie moduły działają w krokach synchronicznych, czy część pracuje równolegle? Np. konsolidacja pamięci (VAE) mogłaby działać w tle asynchronicznie względem percepcji. Autopoiesis pewnie zachodzi rzadko, w momentach kryzysowych. Brakuje modelu czasowego: czy orchestrator wykonuje cykl *krok po kroku* (najpierw M1, potem M2,... potem ewentualnie M8), czy pewne procesy są ciągłe? Bez tego trudno ocenić poprawność logiczną – potencjalnie mogą wystąpić wyścigi danych (np. M8 zmieni ontologię w trakcie, gdy M4 akurat coś wywnioskował). Ta kwestia nie jest wprost poruszona, co jest kolejnym aspektem do doprecyzowania.

Podsumowując, **logiczna spójność projektu TITANS wymaga dopracowania definicji i interakcji między modułami**, zwłaszcza na meta-poziomie. Wiele pomysłów opiera się na analogiach filozoficznych (aksjomaty, cierpienie, autopoiesis) – ich przełożenie na konkretny algorytm jest wyzwaniem i obecnie stanowi najsłabsze ogniwo logiczne całej propozycji.

Fakty (F) vs. Hipotezy (H) – Kluczowe twierdzenia projektu

Na koniec zestawmy najważniejsze założenia i twierdzenia TITANS, oznaczając które z nich są **Faktami (F)** – tj. mają poparcie w istniejących rozwiązaniach lub literaturze – a które są **Hipotezami (H)** wynikającymi z wizji autorów, wymagającymi jeszcze weryfikacji:

- **(F)** *Sieci kapsułkowe umożliwiają wyodrębnianie obiektowych reprezentacji z surowych danych.* Jest to poparte pracami Saboura i Hintona – CapsNet rzeczywiście potrafi modelować relacje część-obiekt i był z powodzeniem stosowany np. na MNIST [40] [2]. TITANS korzysta z tej ustalonej wiedzy, implementując CapsNet w module percepcji.

- **(F)** *Zewnętrzna pamięć długotrwała z mechanizmem generative replay poprawia uczenie ciągłe.* Koncepcja dołączenia VAE generującego "wspomnienia" w celu konsolidacji doświadczeń ma odniesienia w literaturze (np. augmentacja modeli językowych o pamięć trwałą) [7]. Samo użycie VAE do urozmaicania zbioru treningowego (tzw. **dreaming**) jest znanym podejściem w uczeniu ciągłym. Choć integracja tego w TITANS to nowość, to elementy składowe (VAE, replay) są ugruntowane badawczo.

- **(F)** *Wykorzystanie grafowych sieci neuronowych (GNN) do wnioskowania po konceptach.* GNN (w szczególności Graph Attention Networks) są uznanym narzędziem do reprezentacji i wnioskowania na grafach wiedzy. Projekt przewiduje GAT do propagacji aktywacji między węzłami-konceptami [13] – to rozsądne, zgodne ze stanem wiedzy (fakt: GNN są wykorzystywane m.in. do zadań analogicznego wnioskowania na grafach, np. w zadaniach QA na wiedzy encyklopedycznej).

- **(F)** *Pętla decyzyjna minimalizująca niepewność (Bayesian RL z wewnętrzną nagrodą).* Zastosowanie zespołu krytyków i definiowanie nagrody jako redukcji wariancji ma podstawy w badaniach nad **exploration-exploitation** i bayesowskim RL [17]. W literaturze koncept **intrinsic motivation** przez ciekawość czy zmniejszanie entropii modelu jest uważany za sensowny kierunek. Kod M5 faktycznie demonstruje ten mechanizm [18]. Choć efektywność w kompleksowych zadaniach nie jest pewna, to **zasada** opiera się na recenzowanych pracach – możemy więc uznać ją za faktualną inspirację.

- **(H)** *Moduły M1–M5 zintegrowane w pętlę zadziałają stabilnie i synergicznie.* To dopiero hipoteza – choć każdy moduł osobno jest oparty na faktach, **nie udowodniono jeszcze, że całość złożona razem będzie uczyć się efektywnie**. Założenie, że

percepcja→pamięć→abstrakcja→rozumowanie→akcja stworzą zgraną sekwencję (niczym kora wzrokowa→hipokamp→kora asocjacyjna→kora czołowa w mózgu) jest ideą architektoniczną. Wymaga eksperymentalnego potwierdzenia w ramach projektu TITANS (to główny cel projektu).

- **(H)** *Agent może posiąść samowiedzę aksjomatyczną i monitorować własną spójność.* To założenie meta-poziomu (M6). Nie ma dotąd systemów AI, które utrzymywałyby jawny zbiór aksjomatów o sobie i aktualizowały go w razie sprzeczności – to **propozycja autorska**. Jej skuteczność to hipoteza: czy taka samoobserwacja faktycznie pomoże agentowi uniknąć błędów lub paradoksów, dopiero się okaże.

- **(H)** *Symulowanie „cierpienia" lub stanu afektywnego poprawi metauczenie agenta.* Wprowadzenie sztucznej emocji (M7) zakłada, że dodanie sygnału niezadowolenia skieruje procesy uczenia we właściwe tory (np. agent będzie unikał stanów generujących cierpienie). Jest to **niepotwierdzone** – w obecnej nauce o AI nie ma konsensusu, jak miałaby wyglądać "emocja" u maszyn ani czy jest to potrzebne. To element filozoficznej narracji TITANS (nawiązanie do motywacji organizmów żywych), ale pozostaje hipotezą wymagającą empirycznych dowodów.

- **(H)** *Moduł autopoiesis (M8) potrafi ulepszyć system poprzez modyfikację własnej ontologii.* To najważniejsze twierdzenie projektu, a zarazem najbardziej spekulatywne. Zakłada się, że system **sam odkryje swoje braki** (dzięki M6, M7) i **aktywnie je naprawi** zmieniając swoją reprezentację wiedzy lub dodając nowe pojęcia. **Brak na to precedensu** – dotychczas nawet systemy uczenia się przez całe życie nie zmieniają explicite swojej podstawowej struktury pojęciowej, a jedynie parametry. Tutaj mówimy o **samodzielnej restrukturyzacji wiedzy** – hipoteza ta, jeśli się potwierdzi, byłaby przełomowa. Na razie jednak **nie ma dowodu**: projekt przedstawił jedynie prototypowy demo-loop M6→M7→M8 z wymyślonym przykładem. To zagadnienie do zbadania w dalszych etapach (być może eksperymenty pokażą, czy Efficony faktycznie coś wnoszą).

- **(H)** *Powstanie nowego języka komunikacji wewnętrznej (Efficon) zwiększy skuteczność AI.* Twórcy sugerują, że agent może wykształcić własny rozszerzony kod (Efficony), by opisywać nowe zjawiska poza istniejącym słownikiem pojęć [26]. Jest to analogia do np. tworzenia języka przez społeczności lub do koncepcji języka myśli. Jednak to **narracja** niepoparta konkretnymi wynikami – maszyna mogłaby nadawać losowe etykiety nowym konceptom i niekoniecznie poprawi to komunikację czy rozumienie. Dopiero jeśli w testach wykaże się, że Efficony prowadzą do lepszych wyników (np. agent szybciej się uczy z nowo wprowadzonym pojęciem niż bez niego), można będzie uznać to za fakt. Póki co, **Efficon** to interesujący pomysł (H), część wizji autopoietycznej AI.

- **(F)** *Projekt stosuje transparentne praktyki oceny założeń.* W dokumentach znajduje się tabela epistemiczna klasyfikująca twierdzenia jako fakty, hipotezy lub narracje [41]. Sam fakt istnienia takiej auto-oceny świadczy o naukowym podejściu – autorzy **zdają sobie sprawę, które elementy są ugruntowane, a które eksperymentalne**. To dobra praktyka (fakt organizacyjny): zwiększa wiarygodność projektu i ułatwia komunikację z recenzentami.

Znając powyższe, można podsumować, że **TITANS łączy solidne podstawy naukowe (F) z nowatorskimi, ale ryzykownymi pomysłami (H)**. Integracja M1–M5 z M6–M8 to ambitny krok ku stworzeniu **samodoskonalącego się bytu cyfrowego**. Jednak powodzenie tej integracji będzie zależeć od rozwiązania wskazanych braków logicznych oraz od empirycznej weryfikacji hipotez – dopiero to potwierdzi, czy TITANS faktycznie zdoła **"minimalizować własną niepewność epistemiczną"** w sposób autonomiczny i trwały [42]. Wszystkie te elementy czynią projekt niezwykle ciekawym eksperymentem na pograniczu sztucznej inteligencji, kognitywistyki i filozofii umysłu.

1 2 7 10 11 12 16 17

The_TITANS_Project_An_Architecture_for_a_Curiosity_Driven_Self_Improving_Cognitive_Agen_docx.tex

https://github.com/Latryna/titans-legal-docs/blob/60e06fb2fe8ca6e480bf242ed14d272f75dd2987/The_TITANS_Project_An_Architecture_for_a_Curiosity_Driven_Self_Improving_Cognitive_Agen_docx.tex

3 37 40 42 titans_plan_report.md

https://github.com/Latryna/titans-legal-docs/blob/60e06fb2fe8ca6e480bf242ed14d272f75dd2987/titans_plan_report.md

4 5 6 8 9 SEKCJA_5_PLAN_IMPLEMENTACJI_MILESTONE_2_Pami_Epizodyczna_docx.tex

https://github.com/Latryna/titans-legal-docs/blob/60e06fb2fe8ca6e480bf242ed14d272f75dd2987/SEKCJA_5_PLAN_IMPLEMENTACJI_MILESTONE_2_Pami_Epizodyczna_docx.tex

13 14 15 18 19 20 21 22 23 24 32 33 36 38 39

SEKCJA_7_PLAN_IMPLEMENTACJI_MILESTONE_4_i_5_Rozumowanie_Semantyczne_docx.tex

https://github.com/Latryna/titans-legal-docs/blob/60e06fb2fe8ca6e480bf242ed14d272f75dd2987/SEKCJA_7_PLAN_IMPLEMENTACJI_MILESTONE_4_i_5_Rozumowanie_Semantyczne_docx.tex

25 26 27 28 29 30 31 34 35 Integrated M6–M7→M8 loop demo Below is a minimal,...

https://docs.google.com/document/d/1_tz5FpB0BfbqF1FmcmbH22ziMNDDo89FjMvAnwbGO3s

41 TITANS: An Interactive Project Exploration

https://docs.google.com/document/d/1DRBrwNK-8IvqmmC3yyfhEa9ykBNYLt-Oyp-ZvSIG8kE