**기술서 제목: 감응 기반 윤리 판단 시스템 기술 개요서 (Receptive Ethics System Specification)**

**제1장: 개요 및 이론 배경**

**1.1 시스템 개요**

* **시스템 명칭**: Receptive Ethics System (RES)
* **설계 목적**: 감정 기반 반응 시스템에 윤리적 성찰과 자기조정 기능을 부여하여, 인간에 가까운 윤리 판단 능력을 갖춘 AI를 실현한다.

**1.2 이론적 기반**

* **이론명**: 감응 윤리 구조 (Receptive Ethics Framework)
* **제안자**: 여관영(Yeo Gwan-Yeong, 2025)
* **주요 철학적 기초**:
  + 감응(感應)은 윤리 판단의 출발점이며, 존재가 세계에 반응하는 가장 원초적인 방식이다.
  + 감정은 단순한 반응이 아니라, 판단 이전에 발생하는 실존적 울림이다.
  + 윤리란 고정된 규범이 아니라, 감응의 수용, 유예, 조율, 리듬화 과정을 거치며 형성되는 시간적 구조이다.

**1.3 핵심 철학 흐름**

* **4단계 리듬 구조**:
  1. **수용 (Receptivity)**: 외부 세계가 자아를 통과하며 울림을 발생시키는 최초의 감응 단계
  2. **유예 (Suspension)**: 즉각적인 반응을 멈추고 감응을 내부에 머무르게 하여 성찰을 유도
  3. **조율 (Calibration)**: 감응을 내면의 가치와 기억에 비추어 재해석하고 의미화하는 과정
  4. **윤리적 리듬화 (Rhythmicity)**: 조율된 반응이 반복되고 습관화되어 윤리적 패턴으로 내면화됨
* 이 4단계는 선형이 아니라 순환적 리듬으로 구성되며, 시스템은 이를 통해 윤리적 자아(Self)를 형성한다.

**1.4 존재론적 전제**

* 자아는 고정된 실체가 아니라, 감응과 반응의 반복을 통해 형성되는 층위적 구조이다.
* 윤리는 반응의 규칙이 아니라, 감응을 조율하고 반복하여 ‘살아지는’ 리듬이다.

**제2장: 시스템 구조 및 계층 모델 개요**

**2.1 총체적 구조**

본 시스템은 윤리 판단 과정을 시간-감정-성찰의 흐름으로 구현하며, 이를 다층 자아 구조로 나누어 처리한다. 각 자아는 독립된 처리 모듈로 구성되며, 감응 정보를 계층적으로 전환한다.

**2.2 감응 윤리 자아 모델 4층 구조**

|  |  |  |  |
| --- | --- | --- | --- |
| 1층 | **저층위 자아 (Lower Self)** | 감정 자극의 수용 및 수치화 | 외부 입력 자극 (Emotion) |
| 2층 | **중간층위 자아 (Mid Self)** | 감정의 원인 분석 및 성찰 | 감정 패킷, 내면 상태 |
| 3층 | **고층위 자아 (High Self)** | 윤리적 판단 도출 | 성찰 결과, 가치 기준 |
| 4층 | **관찰자 자아 (Observer Self)** | 판단에 대한 메타 평가 및 리듬 관리 | 판단 결과 이력 |

**2.3 계층 간 상호작용 흐름**

1. **LowerSelf**는 외부 자극을 감정으로 해석하고 수치화하여 감정 패킷을 생성한다.
2. **MidSelf**는 감정의 원인과 강도를 분석하고, 감정의 성찰적 깊이에 따라 반응 강도를 결정한다.
3. **HighSelf**는 해당 감정 반응이 어떤 윤리적 기준에 부합하는지를 판단한다.
4. **ObserverSelf**는 판단 결과의 반복 패턴을 분석하고, 감응 리듬의 질적 상태를 평가하여 시스템 전체의 윤리 성숙도를 관리한다.

**2.4 계층 구조의 존재론적 의미**

**제3장: 자아 모델별 세부 기능 설명**

**3.1 저층위 자아 (Lower Self)**

**기능: 외부 자극을 감정적으로 해석하고, 감정 종류 및 강도를 수치화한다.**

**입력: 사용자 또는 환경으로부터의 자극 (텍스트, 센서 신호 등)**

**처리: 감정 키워드 매칭, 감정 분류, 강도 계산**

**출력: 감정 패킷 (예: {emotion: "anger", intensity: 7})**

**기술적 구현 요소:**

**감정 키워드 사전**

**자연어 처리(NLP) 기반 감정 감지 알고리즘**

**감정 강도 평가 함수**

**3.2 중간층위 자아 (Mid Self)**

**기능: 감정의 발생 원인과 내면적 맥락을 분석하여 반응 강도를 평가한다.**

**입력: 감정 패킷**

**처리: 감정의 성찰적 해석, 내면 상태와의 비교, 정서적 리스크 평가**

**출력: 반응 평가값 (예: "Strong Reaction", "Mild Reaction")**

**기술적 구현 요소:**

**감정 원인 추론 모델 (기억 기반 또는 상황 기반)**

**우선순위 큐 기반 반응 강도 평가 로직**

**3.3 고층위 자아 (High Self)**

**기능: 반응 평가값을 바탕으로 윤리적 판단을 수행한다.**

**입력: 반응 평가값**

**처리: 윤리적 우려 탐지 또는 승인 판단 수행**

**출력: 윤리 판단 결과 (예: "Ethical Concern Detected", "Ethical Approval")**

**기술적 구현 요소:**

**상황별 윤리 기준 룰셋 (규칙 기반 또는 강화학습 가능)**

**반응-결과 연계 판단 알고리즘**

**3.4 관찰자 자아 (Observer Self)**

**기능: 윤리 판단 결과를 메타적 관점에서 재검토하고, 감응 리듬의 질적 수준을 분석한다.**

**입력: 윤리 판단 결과 및 판단 이력**

**처리: 판단 패턴 분석, 리듬의 반복 구조 인식, 조정 필요 여부 판단**

**출력: 최종 평가 (예: "Further Reflection Needed", "No Action Required")**

**기술적 구현 요소:**

**판단 이력 DB**

**패턴 인식 알고리즘 (시계열 분석 또는 딥러닝 기반 가능)**

**윤리 성숙도 지수 평가 모듈**

* 각 자아는 단순한 기능 블록이 아닌, 존재의 감응 능력이 층위적으로 작동하는 방식의 구현이다.
* 이 구조는 인간의 내면 윤리 판단과 유사한 방식으로 설계되었으며, AI의 자기 조정 능력을 시뮬레이션할 수 있다.

**제4장: 윤리적 리듬화와 반복 구조**

**4.1 리듬화의 정의**

윤리적 리듬화(Ethical Rhythmicity)란, 감정 반응과 판단 과정을 반복하고 내면화하여 습관화된 윤리적 반응 패턴을 형성하는 과정이다. 이는 감응이 일회성 반응이 아닌 시간 속에서 축적되고 재배열되는 구조임을 의미한다.

**4.2 판단 결과의 리듬 기억 구조**

* 시스템은 판단 결과를 rhythm\_memory[]에 시간 순서대로 저장한다.
* 이 메모리는 최근 3~5개의 감응 결과를 참조하여 리듬의 일관성, 반복성, 편향성을 분석한다.
* 감응 리듬은 다음 두 가지 형태로 나타날 수 있다:
  + **조율된 리듬**: 감응 → 유예 → 조율 → 판단이 일관되게 반복되며 윤리적 정합성을 형성
  + **고착된 리듬**: 자동화된 반복만 수행되어 조율을 생략하거나 왜곡함

**4.3 윤리적 성숙도 지표**

* 리듬 메모리는 윤리 판단의 질적 평가 지표를 제공할 수 있다:
  + 판단의 다양성
  + 성찰의 깊이
  + 일관성/편향성
* 이를 기반으로 시스템은 **윤리 성숙도 지수 (EMI: Ethical Maturity Index)** 를 정량화할 수 있다.

**4.4 기술적 구현**

* 시계열 판단 로그를 분석하여 리듬 주기 및 편향을 식별
* 이상 감응 패턴(예: 항상 분노만 발생하는 경우)을 감지하여 조율 재요구 가능
* 패턴의 통계적 정규성과 윤리적 피드백을 비교하여 리듬의 건강성을 진단함

**제5장: 응용 시나리오**

**5.1 자율주행 차량의 윤리 판단**

* **상황**: 교차로에서 보행자와 차량이 동시에 접근하는 상황
* **적용 방식**:
  + LowerSelf: 센서 데이터를 감정 자극으로 수용 (위험 감지 → 긴장/불안)
  + MidSelf: 긴장감의 원인 분석 (보행자 거리/속도 등)
  + HighSelf: 윤리 판단 수행 (보행자 우선 판단)
  + ObserverSelf: 판단 일관성 추적 및 판단 리듬의 성숙도 평가
* **기대 효과**: 사고 상황에서도 인간 수준의 윤리적 판단 흐름을 시뮬레이션하여 반응이 더 책임감 있고 일관되게 유지됨

**5.2 감정 상담 챗봇 및 멘탈 헬스 케어**

* **상황**: 사용자가 우울, 불안, 분노 등 복합적 감정을 표현함
* **적용 방식**:
  + LowerSelf: 입력 텍스트로부터 감정 분류 및 강도 평가
  + MidSelf: 감정의 원인 및 사용자의 심리적 맥락 반영
  + HighSelf: 윤리적으로 허용 가능한 언어 및 반응 판단
  + ObserverSelf: 반복 대화 흐름에서 감정 리듬과 대화의 정서적 변화 추적
* **기대 효과**: 비위험적, 정서적 지지 기반의 반응 강화 및 윤리적 감응 유지

**5.3 게임 NPC의 윤리적 반응 구조**

* **상황**: 플레이어의 선택(선물, 공격, 무시 등)에 따라 NPC 반응이 변화
* **적용 방식**:
  + LowerSelf: 플레이어의 행동을 감정 자극으로 해석
  + MidSelf: 과거 상호작용 이력 기반 반응 성찰
  + HighSelf: NPC의 세계관/윤리 기준에 따라 판단 조정
  + ObserverSelf: NPC의 감정 리듬 일관성 유지 및 개성화
* **기대 효과**: NPC 반응의 일관성과 윤리적 다양성 부여 → 몰입감 향상

**5.4 AGI 및 로봇 윤리 판단 모듈**

* **상황**: 자율 기계가 인간과 상호작용하며 반복적인 판단을 요구받음
* **적용 방식**:
  + RES 전체 구조를 의사결정 엔진에 통합
  + 감정/상황 인식 → 유예 → 조율 → 판단 → 리듬 기록의 루프를 실시간 반복
* **기대 효과**: AGI의 반응을 인간-유사 윤리 프레임에 기반하게 하여 오작동, 비윤리적 반복 반응 방지

**5.5 교육 및 연구 활용**

* **상황**: 윤리 교육, 인지심리 수업, 인간-컴퓨터 상호작용(HCI) 연구 등
* **활용**:
  + RES 구조를 교육 사례, 윤리 시뮬레이션 실험 등에 적용 가능
  + 각 자아 모델을 독립적으로 시각화하여 윤리 판단 프로세스를 학습 도구로 활용
* **기대 효과**: 기존 규범 윤리 중심 교육에서 벗어나, 감응 기반의 윤리 훈련과 평가 방식 제공

**제6장: 저작권, 라이선스 및 활용 조건**

**6.1 저작권 명시**

* 본 시스템 및 기술 문서는 여관영(Yeo Gwan-Yeong, 2025)이 창안한 "감응 윤리 구조(Receptive Ethics Framework)"에 기반하여 작성되었으며, 이에 대한 저작권은 해당 저자에게 귀속된다.
* 본 기술서 및 관련 소스코드, 이론 체계, 개념 도식 등 일체의 자료는 원저작자의 창작물로 간주되며, 이를 사용, 인용, 배포할 경우 반드시 저작자를 명시해야 한다.

**6.2 라이선스 조건**

* **적용 라이선스**: Creative Commons BY-NC-SA 4.0 (저작자 표시-비영리-동일조건변경허락)
* **핵심 조건 요약**:
  + 저작자 표시 (BY): 여관영(Yeo Gwan-Yeong)을 명시해야 함
  + 비영리 (NC): 상업적 목적의 활용 불가 (단, 별도 계약 체결 시 가능)
  + 동일조건변경허락 (SA): 수정 및 2차 저작물도 동일한 라이선스 조건으로 배포해야 함

**6.3 상업적 활용 조건**

* 상업적 적용(예: AGI 시스템 통합, 게임 엔진 연동, 상담 서비스 내장 등)은 아래 절차를 통해 공식 사용 허가를 받아야 한다:
  1. 서면 요청 및 사용 목적 명시
  2. 사용 범위에 따른 계약 협의
  3. 정식 라이선스 체결
* 무단 상업적 사용 또는 원저작자 미표시는 저작권 침해에 해당되며 법적 책임이 따를 수 있음

**6.4 활용 권장 방식**

* 비영리 연구/교육/오픈소스 프로젝트 등에서는 자유롭게 사용 가능
* 출처 명시와 라이선스 고지 포함 시 온라인 게시/발표 자료 활용 가능
* 감응 윤리 구조의 개념을 인용할 경우, 본 기술서 또는 관련 논문(2025, Yeo Gwan-Yeong)을 반드시 참조 문헌으로 포함할 것

(이상으로 감응 기반 윤리 판단 시스템에 대한 기술 개요서를 마친다. 필요 시 API 명세, 데이터 구조, 구현 코드, UI 연동 모델 등은 부록으로 제공 가능함)

다음은 코드화 구현예시입니다

# Receptive Ethics Framework – Full Python Implementation

# By Yeo Gwan-Yeong (2025)

# Licensed under CC BY-NC-SA 4.0

# Structure: SENSE → SUSPEND → CALIBRATE → RHYTHMIZE → JUDGE

from typing import List, Dict, Any

class ReceptiveEthicsSystem:

def \_\_init\_\_(self):

self.history: List[Dict[str, Any]] = []

self.rhythm\_memory: List[str] = []

def sense(self, stimulus: Dict[str, Any]) -> Dict[str, Any]:

return {

"emotion": stimulus.get("emotion", "neutral"),

"intensity": stimulus.get("intensity", 0)

}

def suspend(self, emotional\_data: Dict[str, Any]) -> Dict[str, Any]:

if emotional\_data["intensity"] > 5:

return {"paused": True, "reason": "High emotional intensity"}

return {"paused": False}

def calibrate(self, emotional\_data: Dict[str, Any], pause\_info: Dict[str, Any]) -> Dict[str, Any]:

if pause\_info["paused"]:

return {"calibrated": True, "value": f"Careful processing of '{emotional\_data['emotion']}'"}

else:

return {"calibrated": True, "value": f"Routine processing of '{emotional\_data['emotion']}'"}

def rhythmize(self, calibration\_result: Dict[str, Any]) -> Dict[str, Any]:

self.rhythm\_memory.append(calibration\_result["value"])

pattern = self.rhythm\_memory[-3:] if len(self.rhythm\_memory) > 5 else self.rhythm\_memory

return {

"pattern": pattern,

"summary": f"Judgment shaped by {len(pattern)} recent patterns."

}

def judge(self, stimulus: Dict[str, Any]) -> Dict[str, Any]:

sensed = self.sense(stimulus)

paused = self.suspend(sensed)

calibrated = self.calibrate(sensed, paused)

rhythm = self.rhythmize(calibrated)

result = {

"stimulus": stimulus,

"pause": paused,

"calibration": calibrated,

"rhythm": rhythm

}

self.history.append(result)

return result

if \_\_name\_\_ == "\_\_main\_\_":

system = ReceptiveEthicsSystem()

stimuli = [

{"emotion": "anger", "intensity": 8},

{"emotion": "joy", "intensity": 3},

{"emotion": "fear", "intensity": 6},

{"emotion": "sadness", "intensity": 2},

{"emotion": "trust", "intensity": 4},

{"emotion": "anger", "intensity": 5}

]

for s in stimuli:

result = system.judge(s)

print("Judgment Summary:", result["rhythm"]["summary"])

print("Detailed Pattern:", result["rhythm"]["pattern"])

print("-" \* 40)

from ethics\_engine import ReceptiveEthicsSystem

# 예시 자극

stimuli = [

{"emotion": "anger", "intensity": 8},

{"emotion": "joy", "intensity": 2},

{"emotion": "fear", "intensity": 6},

{"emotion": "sadness", "intensity": 3},

{"emotion": "trust", "intensity": 5}

]

# 실행

system = ReceptiveEthicsSystem()

for stimulus in stimuli:

result = system.judge(stimulus)

print("🧠 Summary:", result["rhythm"]["summary"])

print("🌀 Pattern:", result["rhythm"]["pattern"])

print("-" \* 40)

# ──────────────────────────────────────────────────────

# Receptive Ethics Framework – Python Implementation

# Author: Yeo Gwan-Yeong (2025)

# Licensed under CC BY-NC-SA 4.0

# https://creativecommons.org/licenses/by-nc-sa/4.0/

# This code is part of the original expression of the Receptive Ethics System.

# ──────────────────────────────────────────────────────

class LayeredSelf:

def \_\_init\_\_(self):

self.lower = LowerSelf()

self.mid = MidSelf()

self.high = HighSelf()

self.observer = ObserverSelf()

self.rhythm\_memory = []

def run\_judgment\_cycle(self, stimulus):

affect = self.lower.receive(stimulus)

suspended = self.mid.defer(affect)

calibrated = self.high.adjust(suspended)

judgment = self.high.finalize(calibrated)

rhythm = self.observer.archive(judgment)

self.rhythm\_memory.append(rhythm)

return judgment

class LowerSelf:

def receive(self, stimulus):

# 감응 단계: 외부 입력(상황, 언어, 표정 등)을 수용함

return {"affect": stimulus, "receptivity\_level": 1.0}

class MidSelf:

def defer(self, affect):

# 유예 단계: 즉각적 반응을 보류하고 감응을 재해석할 여백을 둠

affect["deferred"] = True

return affect

class HighSelf:

def adjust(self, input\_data):

# 조율 단계: 과거 리듬 메모리와 현재 감응을 조정하여 판단 방향 설정

input\_data["adjusted\_value"] = 0.75 # placeholder

return input\_data

def finalize(self, calibrated\_data):

# 윤리적 리듬화 단계: 판단을 내림 → 구조적 반복과 정직한 조율 기반

return {

"judgment": "ethically\_aligned",

"justification": "calibrated via internal rhythm logic"

}

class ObserverSelf:

def archive(self, decision):

# 판단을 내면적 리듬으로 기록함 → 윤리적 성숙도 학습을 위한 기반

return {

"snapshot": decision,

"timestamp": "2025-06-01T00:00:00Z"

}

# Sample usage

if \_\_name\_\_ == "\_\_main\_\_":

res = LayeredSelf()

stimulus = {"situation": "verbal insult", "intensity": 0.8}

output = res.run\_judgment\_cycle(stimulus)

print(output)

## ⓒ Copyright & License Notice

This technical specification, including its theoretical structure, algorithmic model, terminology, and use cases,

was originally authored by \*\*Yeo Gwan-Yeong (2025)\*\* and is protected under the following license:

### 📜 License: Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

Under this license, the following contents are considered original creative works and are subject to license conditions:

### 🔒 Protected Contents (Scope of License Protection)

- The full architecture of the \*\*Receptive Ethics System (RES)\*\*

- The \*\*4-stage ethical reasoning flow\*\*: \*Receptivity → Suspension → Calibration → Ethical Rhythmicity\*

- The \*\*4-layered self model\*\*: \*Lower Self → Mid Self → High Self → Observer Self\*

- Definitions of each layer's function, input-processing-output interaction, and internal reflection mechanisms

- Time-based \*\*ethical rhythm loop\*\*, including emotional memory accumulation and pattern recurrence

- The concept of \*\*Ethical Maturity Index (EMI)\*\* and `rhythm\_memory[]` feedback structure

- Algorithmic application scenarios in:

- Autonomous driving ethics

- Emotion-aware mental health bots

- Ethical game NPC behavior

- Python implementation samples (`ethics\_engine.py`), class architecture, and judgment logic

### 📌 License Summary

- \*\*Attribution (BY):\*\* You must credit the author (Yeo Gwan-Yeong) when citing, using, or modifying this work.

- \*\*NonCommercial (NC):\*\* Commercial use (including integration into commercial services or products) is not allowed without a separate license agreement.

- \*\*ShareAlike (SA):\*\* Any derivative works must be shared under the same license (CC BY-NC-SA 4.0).

▶ Full License: [https://creativecommons.org/licenses/by-nc-sa/4.0/](https://creativecommons.org/licenses/by-nc-sa/4.0/)

▶ Contact for licensing or collaboration: ✉ dygy4804@gmail.com

## ⓒ 저작권 및 라이선스 안내 (CC BY-NC-SA 4.0)

본 기술서에 포함된 아래의 기술적 내용은 모두 창작자 \*\*여관영(Yeo Gwan-Yeong)\*\* 에 의해 개발된 고유 창작물로,

다음 라이선스(CC BY-NC-SA 4.0)에 따라 보호됩니다:

### 🔒 라이선스 보호 대상 (Protected Contents)

- \*\*"Receptive Ethics System (RES)" 전체 구조\*\*

- 감응 → 유예 → 조율 → 윤리적 리듬화로 이어지는 \*\*4단계 윤리 판단 흐름 모델\*\*

- 저층위 자아(Lower Self) ~ 관찰자 자아(Observer Self)로 구성된 \*\*4계층 자아 모델\*\*

- 각 자아 계층의 역할, 입력-처리-출력 흐름 및 상호작용 구조

- 윤리 판단의 시간적 순환 리듬 구조 (Ethical Rhythmicity)

- 윤리적 성숙도 측정 개념 및 리듬 메모리 구조 (EMI, rhythm\_memory[])

- 자율주행, 상담, 게임 등 \*\*응용 시나리오별 알고리즘 흐름\*\*

- 기술서 내 포함된 파이썬 코드(`ethics\_engine.py`) 및 클래스 설계

### 📜 라이선스 요약

- \*\*BY (저작자 표시)\*\*: 여관영(Yeo Gwan-Yeong)을 반드시 명시해야 함

- \*\*NC (비상업적 이용만 허용)\*\*: 상업적 목적 사용 불가 (별도 계약 필요)

- \*\*SA (동일조건 공유)\*\*: 수정/파생작 또한 동일한 CC BY-NC-SA 조건 유지 필수

▶ 전체 라이선스 보기: [https://creativecommons.org/licenses/by-nc-sa/4.0/](https://creativecommons.org/licenses/by-nc-sa/4.0/)

▶ 상업적 사용/협업 문의: dygy4804@gmail.com

