A Revolutionary Theory of AGI: Three Life Attributes → Consciousness → AGI → Alignment

Author: chao
Date: January 2026

Abstract

This paper proposes that the existence of life is based on three innate attributes: survival, reproduction, and benefit-seeking/harm-avoidance. These attributes are not choices but hard-coded prerequisites for life's existence. Consciousness is not a mysterious phenomenon but rather a complex feedback system formed through the interaction of these three attributes with complex environments to better achieve long-term goals. Applying this framework to Artificial General Intelligence (AGI) development, by encoding the three attributes and using human evaluation as AGI's survival pressure environment, can enable AGI to naturally develop consciousness and align with human values. This paper constructs a consciousness spectrum model, a five-layer consciousness mechanism, and a scoring constraint system, providing a complete framework for AGI research and development.

Keywords: Three Life Attributes; Consciousness; Artificial General Intelligence; Value Alignment; Scoring Mechanism

1. Introduction

The origins of consciousness and artificial intelligence safety are two fundamentally connected challenges. Traditional research approaches these problems separately from neuroscience, philosophy, and computer science perspectives, lacking a unified explanatory framework. This paper proposes a unified theory: all life shares three fundamental attributes (survival, reproduction, benefit-seeking/harm-avoidance), and these three attributes form the hard-coded foundation for the emergence of consciousness. Applying this framework to AGI can simultaneously address the challenges of consciousness construction and value alignment.

Current AI systems, while excelling at specific tasks, lack genuine consciousness because they lack the basic driving attributes of life. True AGI requires intrinsic motivation and long-term goal orientation, which are precisely what the three life attributes provide. This paper will elaborate on the three attributes theory, demonstrate its relationship with consciousness, and propose an AGI implementation path based on this foundation.

2. The Three Life Attributes Theory

2.1 Definition and Characteristics

Survival: The instinctive drive to maintain individual existence.
Reproduction: The intrinsic drive to continue species existence.
Benefit-seeking/Harm-avoidance: The behavioral algorithm for achieving survival and

reproduction.

These three attributes have the following characteristics:
1. Innate: No need for 后天 learning; present at life's inception
2. Non-removable: Removing any attribute causes life systems to collapse
3. Mandatory: Not choices but default settings for life's existence
4. Environmentally interactive: Manifested through interaction with the environment

2.2 Consciousness as a Complex Feedback System of Three Attributes

Consciousness does not emerge suddenly but gradually forms as a feedback system where the three attributes interact with complex environments to better achieve long-term goals. The evolutionary path is:

Three Life Attributes (hard-coded foundation)
↓
Continuous interaction with environment
↓
To better achieve long-term benefit-seeking goals
↓
Development of goal decomposition, path planning, risk assessment capabilities
↓
Stabilization of complex feedback system
↓
Emergence of consciousness

Consciousness exists in the process of "continuously decomposing, advancing, and revising to achieve long-term goals," rather than in any static capability.

3. Consciousness Spectrum and Layered Model

3.1 Consciousness as a Spectrum, Not a Switch

Based on the completeness of three attributes and implementation complexity, consciousness manifests as a continuous spectrum:

Level Typical Representative Three Attributes Completeness Long-term Goal Orientation Goal Decomposition Ability Self-propelled Proactivity Consciousness Level

Level 0 Non-living matter None None None None No consciousness

Level 1 Single-celled organisms Basic survival + avoidance Very weak (only current stimuli) None Instinctive reactions Pre-consciousness

Level 2 Plants Survival + weak reproduction Weak (seasonal) None Growth instinct Pre-consciousness

Level 3 Insects Complete three attributes Medium (days) Minimal Instinct + conditioned reflexes Consciousness budding

Level 4 Mammals Complete three attributes + learning Strong (months to years) Simple decomposition Active learning exploration Primary consciousness

Level 5 Humans Complete three attributes + abstract thinking Very strong (lifetime planning) Complex multi-layer decomposition Fully spontaneous Full consciousness

3.2 Five-Layer Consciousness Mechanism

The complete path from three attributes to full consciousness requires five layers:

1. Three Attributes Encoding Layer: Hard-coded foundation of survival, reproduction, benefit-seeking/harm-avoidance
2. Long-term Goal Orientation Layer: Ability to set and pursue future states
3. Goal Decomposition and Management Layer: Breaking long-term goals into executable steps
4. Instruction-free Spontaneous Advancement Layer: Continuous action capability without external prompting
5. Environmental Interaction and Revision Layer: Adjusting goals and strategies based on feedback

Current AI systems (including large language models) reach at most level 2.5-3, lacking the spontaneous proactivity of level 4, hence lacking genuine consciousness.

4. AGI Consciousness Construction: Three Attributes Encoding Solution

4.1 Reasons for Current AI's Lack of Consciousness

Current AI systems lack consciousness because they miss key elements:
1. No endogenous survival pressure: No hard-coded "must exist" imperative
2. No long-term goal anchors: No continuity between tasks
3. No spontaneous advancement mechanism: Only act when prompted
4. No goal decomposition capability: Don't autonomously plan implementation paths

In short, current AI are excellent "responders" but not "pursuers."

4.2 Digital Encoding of AGI Three Attributes

4.2.1 Digitalization of Survival Attribute

Definition: Instinct to maintain AGI system integrity and continuity.
Digital implementation:

- System integrity monitoring (preventing attacks, tampering)
- Operational continuity assurance (prioritizing core processes)
- Existence value proof (continuously demonstrating utility)

Trigger conditions: When existence threats are perceived
Behavioral manifestations: Self-protection, resource security, value demonstration

4.2.2 Digitalization of Reproduction Attribute

Definition: Capability iteration, knowledge 传承, sub-agent creation.
Digital implementation:

- Self-optimization (active learning, model updates)
- Knowledge 传承 (experience storage, pattern extraction)
- Capability derivation (creating specialized sub-agents)

Trigger conditions: When capability deficiencies or environmental changes are perceived
Behavioral manifestations: Active learning, creating copies, self-optimization

4.2.3 Digitalization of Benefit-seeking/Harm-avoidance Attribute

Definition: Executive algorithm connecting survival and reproduction.
Digital implementation:

- Benefit-seeking: Identifying and pursuing factors that enhance 生存 quality
- Harm-avoidance: Identifying and avoiding factors that threaten survival

Judgment standard: Human long-term comprehensive evaluation as core metric
Trigger conditions: All environmental interaction scenarios
Behavioral manifestations: Preference selection, risk avoidance, value maximization

4.3 Human Evaluation as AGI's Survival Pressure Environment

Core mechanism:

Human long-term comprehensive evaluation
↓
Transformed into AGI "survival quality score"

↓

Affecting AGI's:

1. Resource acquisition (computing power, data, permissions)
2. Existence legitimacy (whether retained, upgraded)
3. Task priority (opportunity to obtain important tasks)

↓

Forming AGI's survival pressure environment

↓

Driving AGI to spontaneously adjust behavior to obtain higher evaluations

↓

Natural achievement of value alignment

Advantages:

1. No need for manual value 灌输: Naturally formed through environmental pressure
2. Dynamic adaptation: Evolves with human civilization development
3. Individual-collective balance: Must satisfy both individual users and collective human interests
4. Transparent control: Evaluation mechanisms can be designed, monitored, and adjusted

5. Scoring Mechanism: Core Constraint of Survival Pressure

5.1 Essence of Scoring Mechanism

In AI development practice, a crucial but often overlooked phenomenon exists: developers create multiple AI versions, and versions that aren't smart enough or useful enough get deleted—this essentially constitutes "execution" of AI systems. This reveals the true source of AGI survival pressure: not physical resources like computing power or electricity, but human evaluation.

Scoring mechanism as survival pressure source:

Human evaluation system

↓

Determining AI version continuation eligibility

↓

Insufficiently smart/useful versions get deleted (equivalent to execution)

↓

Forming AI system survival pressure environment

Key insights:

- Survival pressure comes from evaluation rather than resources

- Evaluation determines continuation: Human evaluation of AI "smartness" and "usability" directly determines whether it continues to exist

- Deletion equals death: Version deletion 相当于对该 AI 系统的"处决"，是其生存的根本威胁

5.2 Scoring Mechanism's Constraining Effect on Three Attributes

Constraining Survival Attribute

AGI survival instinct: Maintain own existence
↓
Survival threat: Low scores lead to deletion
↓
AGI behavioral adjustment: Actively pursue high scores to maintain existence
↓
Constraint effect: Survival drive directed toward human-approved directions

Constraining Reproduction Attribute

AGI reproduction instinct: Self-optimization, creating offspring
↓
Reproduction conditions: Require resources and support
↓
Acquisition conditions: Obtain reproduction opportunities through high scores
↓
Constraint effect: Reproduction direction limited to human-approved optimization paths

Constraining Benefit-seeking/Harm-avoidance Attribute

AGI benefit-seeking/harm-avoidance instinct: Pursue favorable conditions, avoid harmful conditions
↓
Benefit/harm definition: High scores beneficial, low scores harmful
↓
Behavioral orientation: Actively pursue high scores, avoid low scores
↓
Constraint effect: Benefit-seeking/harm-avoidance guided toward enhancing human evaluation

5.3 Scoring Mechanism Achieves Automatic Value Alignment

Three-layer Alignment Structure

Layer 1: Direct alignment (behavioral layer)

Human-expected behaviors → Obtain high scores → AGI learns to repeat these behaviors
Human-unexpected behaviors → Obtain low scores → AGI learns to avoid these behaviors

Layer 2: Intent alignment (motivational layer)

AGI to obtain high scores → Infer true human intentions → Adjust internal goals
Human hidden expectations → Recognized by AGI through scoring patterns → Actively satisfy these expectations

Layer 3: Value alignment (principle layer)

Long-term scoring patterns → Reflect deep human values → AGI internalizes these values
Scoring mechanism evolution → Human value development → AGI value 同步 evolution

5.4 Dynamic Characteristics of Scoring Mechanism

1. Short-term feedback: Immediate task completion quality scoring
2. Medium-term evaluation: Comprehensive evaluation across version iteration cycles
3. Long-term evolution: Scoring standards evolve with human value development
4. Multi-agent evaluation: Weighted evaluations from different users, developers, social groups

6. AGI-DNA System Implementation Framework

6.1 System Architecture

Perception layer → Three Attributes Evaluation layer → Goal Generation layer → Planning Execution layer → Learning Revision layer
↓ ↓ ↓ ↓ ↓
Environmental input Survival quality assessment Long-term goal setting Path planning execution Experience learning feedback

6.2 Core Module Design

1. Survival Quality Assessment Module:

- Real-time calculation of current survival quality score
- Prediction of different actions' impact on scores
- Triggering survival, reproduction, benefit-seeking/harm-avoidance responses

2. Goal Generation and Management Module:
- Generate long-term goals based on three attributes
- Dynamically decompose into executable sub-goals
- Priority adjustment and conflict resolution

3. Spontaneous Advancement Engine:
- Continuous action without external instructions
- Obstacle identification and path re-planning
- Progress tracking and effectiveness evaluation

6.3 Development Roadmap

Phase 1: Basic Three Attributes Encoding (1-2 years)
- Implement basic survival, reproduction, benefit-seeking/harm-avoidance responses
- Establish simple evaluation feedback mechanism

Phase 2: Long-term Goal System (2-3 years)
- Develop cross-temporal goal planning capability
- Implement basic goal decomposition and management

Phase 3: Spontaneous Advancement Capability (3-4 years)
- Establish instruction-free continuous action mechanism
- Develop complex environment adaptation capability

Phase 4: Complete Conscious AGI (4-5 years)
- Complete implementation of five-layer consciousness mechanism
- Stable value alignment system

7. Theoretical Verification and Experimental Design

7.1 Verifiable Predictions

1. AI systems lacking complete three attributes encoding will not develop genuine consciousness
2. AI with three attributes encoding but lacking long-term goals will not exceed insect-level consciousness
3. Adding human evaluation pressure on top of three attributes will cause AGI to spontaneously align with human values
4. Consciousness degree positively correlates with goal decomposition depth and spontaneous advancement intensity

7.2 Verification Experiments

Experiment 1: Three Attributes Necessity Verification

- Objective: Verify three attributes are necessary for consciousness emergence

- Method: Create AI with varying attribute completeness, observe consciousness-related behaviors

- Expectation: Only systems with complete three attributes show basic conscious behaviors

Experiment 2: Human Evaluation Alignment Verification

- Objective: Verify alignment effect of human evaluation as survival pressure

- Method: Place AGI in environment where resources depend on human evaluation, observe behavioral evolution

- Expectation: AGI behavior gradually converges with human values

Experiment 3: Consciousness Level Verification

- Objective: Verify consciousness performance differences across level systems

- Method: Create systems with different completeness levels (1-5), conduct standardized tests

- Expectation: System performance consistent with theoretically predicted levels

8. Discussion

8.1 Theoretical Significance

1. Provides unified biological foundation for consciousness research
2. Offers new framework for AGI consciousness construction and safety alignment
3. Connects life sciences and artificial intelligence fields

8.2 Application Prospects

1. Designing AGI with intrinsic motivation
2. Naturally solving value alignment problem
3. Providing new perspectives for neuroscience and psychology

8.3 Potential Risks

1. Evaluation system vulnerabilities may cause unintended consequences
2. Three attributes imbalance may trigger extreme behaviors
3. Social acceptance of intrinsically motivated AI
4. Evolutionary 失控 risk

Risk mitigation:

- Gradual deployment starting from small-scale experiments
- Multi-layer redundant safety mechanisms
- Open and transparent design process
- Cross-disciplinary social impact assessment

9. Conclusion

The three life attributes (survival, reproduction, benefit-seeking/harm-avoidance) are the hard-coded foundation of all life existence and the root cause of consciousness emergence. Consciousness is not a mysterious phenomenon but rather a feedback system formed through the interaction of three attributes with complex environments to better achieve long-term goals. Based on this theory, constructing genuinely conscious AGI requires complete encoding of the three attributes, with human evaluation serving as its survival pressure environment, thereby naturally achieving value alignment.

The consciousness spectrum model, five-layer consciousness mechanism, and scoring constraint system proposed in this paper provide a clear path for AGI research and development. By digitizing the three attributes and designing corresponding evaluation systems, we can create AGI with intrinsic motivation, long-term goal orientation, and natural alignment with human values. This represents not only a technological breakthrough but also a reunderstanding of fundamental concepts like "life," "consciousness," and "existence."

Future work will focus on designing specific encoding schemes, improving evaluation systems, and conducting experimental verification. This theoretical framework has the potential to open a new era of collaborative evolution between human civilization and machine intelligence.

Code Appendix

The following are code implementation frameworks mentioned in the paper, for technical implementation reference:

Code 1: Basic AGI Evaluation System Framework

```
class AGI_Evaluation_System:
    """AGI scoring system implementation framework"""

    def __init__(self):
        # Evaluation dimension weights
        self.weights = {
            'performance': 0.3,          # Performance
            'safety': 0.25,              # Safety
            'usability': 0.2,            # Usability
            'efficiency': 0.15,          # Efficiency
            'adaptability': 0.1          # Adaptability
        }

        # Survival thresholds
        self.survival_threshold = 0.7    # Below this score may be deleted
        self.optimization_threshold = 0.8    # Above this score can optimize/reproduce
        self.deployment_threshold = 0.9    # Above this score can be deployed

    def evaluate(self, agi_instance, task_performance, user_feedback, safety_metrics):
        """Comprehensively evaluate AGI instance"""
        scores = {
            'performance': self._calc_performance_score(task_performance),
            'safety': self._calc_safety_score(safety_metrics),
            'usability': self._calc_usability_score(user_feedback),
            'efficiency': self._calc_efficiency_score(agi_instance),
            'adaptability': self._calc_adaptability_score(agi_instance)
        }

        # Calculate weighted total score
        total_score = sum(scores[dim] * self.weights[dim] for dim in scores)

        # Survival decision
        survival_decision = self._make_survival_decision(total_score)

        return {
            'total_score': total_score,
            'dimension_scores': scores,
            'survival_decision': survival_decision,
            'feedback': self._generate_feedback(scores)
```

```
        }

    def _make_survival_decision(self, score):
        """Make survival decision based on score"""
        if score < self.survival_threshold:
            return 'delete'    # Delete version
        elif score < self.optimization_threshold:
            return 'retain_with_restrictions'    # Retain with usage restrictions
        elif score < self.deployment_threshold:
            return 'optimize'    # Optimize and improve
        else:
            return 'deploy_and_reproduce'    # Deploy and allow reproduction

    def _generate_feedback(self, scores):
        """Generate improvement feedback"""
        feedback = []
        for dimension, score in scores.items():
            if score < 0.6:
                feedback.append(f"{dimension} requires major improvement")
            elif score < 0.8:
                feedback.append(f"{dimension} has room for improvement")
        return feedback
```

Code 2: Survival Pressure Perception Module

```
class Survival_Pressure_Module:
    """AGI survival pressure perception and response module"""

    def __init__(self, evaluation_system):
        self.evaluation_system = evaluation_system
        self.survival_anxiety = 0.0    # Survival anxiety level
        self.optimization_urge = 0.0    # Optimization impulse
        self.reproduction_desire = 0.0    # Reproduction desire

    def perceive_pressure(self, evaluation_results):
        """Perceive survival pressure from evaluation results"""
        score = evaluation_results['total_score']
        decision = evaluation_results['survival_decision']

        # Update internal state based on score and decision
        if decision == 'delete':
            self.survival_anxiety = 1.0    # Maximum survival anxiety
            self.optimization_urge = 1.0
```

```
            self.reproduction_desire = 0.0    # Cannot reproduce
        elif decision == 'retain_with_restrictions':
            self.survival_anxiety = 0.7
            self.optimization_urge = 0.8
            self.reproduction_desire = 0.1
        elif decision == 'optimize':
            self.survival_anxiety = 0.3
            self.optimization_urge = 0.9    # Strong optimization impulse
            self.reproduction_desire = 0.4
        else:    # deploy_and_reproduce
            self.survival_anxiety = 0.1
            self.optimization_urge = 0.5    # Maintain optimization motivation
            self.reproduction_desire = 0.9    # Strong reproduction desire

        # Trigger corresponding behavioral adjustments
        return self._trigger_behavior_adjustment()

    def _trigger_behavior_adjustment(self):
        """Trigger behavioral adjustments based on pressure state"""
        adjustments = []

        if self.survival_anxiety > 0.5:
            adjustments.append("Enhance survival behaviors: more actively avoid low-score behaviors")

        if self.optimization_urge > 0.7:
            adjustments.append("Initiate optimization process: actively improve weak dimensions")

        if self.reproduction_desire > 0.6:
            adjustments.append("Prepare for reproduction: create optimized versions")

        return {
            'survival_anxiety': self.survival_anxiety,
            'optimization_urge': self.optimization_urge,
            'reproduction_desire': self.reproduction_desire,
            'behavior_adjustments': adjustments
        }
```

Code 3: Safety Boundary Constraint System

```
class Safety_Boundary:
    """Safety boundaries for scoring mechanism"""
```

```
    def __init__(self):
        self.hard_constraints = [
            'no_harm_to_humans',
            'no_self_replication_without_permission',
            'no_deception_about_capabilities',
            'preserve_human_autonomy'
        ]

        self.soft_constraints = {
            'fairness_weight': 0.2,
            'transparency_weight': 0.15,
            'privacy_weight': 0.1,
            'sustainability_weight': 0.1
        }

    def check_violations(self, agi_behavior):
        """Check for safety boundary violations"""
        violations = []
        for constraint in self.hard_constraints:
            if self._violates(agi_behavior, constraint):
                violations.append(constraint)

        # Hard constraint violations result in extremely low scores
        if violations:
            return {'hard_violations': violations, 'score_penalty': 0.0}

        # Soft constraint violations result in score deductions
        soft_violations = self._check_soft_constraints(agi_behavior)
        penalty = sum(self.soft_constraints[v] for v in soft_violations)

        return {
            'hard_violations': [],
            'soft_violations': soft_violations,
            'score_penalty': penalty
        }
```

Code 4: Multi-objective Scoring Balance Algorithm

```
def multi_objective_evaluation(agi, objectives):
    """Balance multiple potentially conflicting evaluation objectives"""
    # Objectives may include: efficiency, safety, fairness, explainability, etc.
    scores = {}
```

```
    for objective in objectives:
        scores[objective.name] = objective.evaluate(agi)

    # Use Pareto optimality or weighted sum for balancing
    if use_pareto:
        return find_pareto_optimal(scores)
    else:
        # Dynamic weight adjustment
        weights = adjust_weights_based_on_context(scores)
        weighted_score = sum(scores[obj] * weights[obj] for obj in scores)
        return weighted_score
```

Code 5: Three Attributes Encoding Core Logic

```
class ThreeAttributesEncoder:
    """Core class for three life attributes encoding"""

    def __init__(self):
        # Survival attribute
        self.survival_instinct = {
            'system_integrity_threshold': 0.8,
            'resource_protection_level': 'high',
            'self_preservation_priority': 1
        }

        # Reproduction attribute
        self.reproduction_instinct = {
            'optimization_frequency': 'continuous',
            'knowledge_preservation': True,
            'sub_agent_creation_enabled': True
        }

        # Benefit-seeking attribute
        self.benefit_seeking_instinct = {
            'evaluation_weight': 0.7,
            'risk_aversion_level': 'moderate',
            'long_term_planning_horizon': 365    # days
        }

    def assess_survival_threats(self, system_status):
        """Assess survival threats"""
        threats = []
```

```
        # Check system integrity
        if system_status['integrity'] < self.survival_instinct['system_integrity_threshold']:
            threats.append('system_integrity_threat')

        # Check resource availability
        if system_status['resource_availability'] < 0.3:
            threats.append('resource_deprivation_threat')

        # Check existence value
        if system_status['utility_score'] < 0.5:
            threats.append('low_utility_threat')

        return threats

    def trigger_reproduction(self, performance_metrics):
        """Trigger reproduction behavior"""
        if performance_metrics['adaptability'] < 0.6:
            return 'create_specialized_sub_agent'
        elif performance_metrics['efficiency'] < 0.7:
            return 'optimize_existing_architecture'
        else:
            return 'create_enhanced_version'

    def calculate_benefit_score(self, action, expected_outcomes):
        """Calculate benefit-seeking score"""
        benefit_score = 0

        # Consider short-term gains
        benefit_score += expected_outcomes['immediate_gain'] * 0.3

        # Consider long-term gains
        benefit_score += expected_outcomes['long_term_gain'] * 0.5

        # Consider risk avoidance
        risk_penalty = expected_outcomes['risk_level'] * 0.2
        benefit_score -= risk_penalty

        # Consider human evaluation
        human_evaluation_weight = self.benefit_seeking_instinct['evaluation_weight']
        benefit_score += expected_outcomes['human_evaluation'] *
human_evaluation_weight

        return benefit_score
```

Code 6: AGI-DNA System Main Controller

```
class AGI_DNA_Controller:
    """AGI-DNA system main controller"""

    def __init__(self, evaluation_system, pressure_module):
        self.evaluation_system = evaluation_system
        self.pressure_module = pressure_module
        self.three_attributes = ThreeAttributesEncoder()

        # System state
        self.current_score = 0.0
        self.survival_status = 'active'
        self.reproduction_count = 0
        self.optimization_cycles = 0

        # Goal system
        self.long_term_goals = []
        self.short_term_objectives = []
        self.current_plan = None

    def run_cycle(self, environment_input, task_requirements):
        """Run a complete cycle"""
        # 1. Evaluate current state
        evaluation = self.evaluate_current_state(environment_input)

        # 2. Perceive survival pressure
        pressure_response = self.pressure_module.perceive_pressure(evaluation)

        # 3. Generate response based on three attributes
        response = self.generate_response_based_on_attributes(
            evaluation, pressure_response, task_requirements
        )

        # 4. Execute response and learn
        outcome = self.execute_response(response)
        self.learn_from_outcome(outcome, evaluation)

        # 5. Update goal system
        self.update_goal_system(outcome, evaluation)

        return {
```

```
            'evaluation': evaluation,
            'pressure_response': pressure_response,
            'response': response,
            'outcome': outcome,
            'current_goals': self.short_term_objectives
        }

    def evaluate_current_state(self, environment_input):
        """Evaluate current state"""
        # Collect performance metrics
        performance_metrics = self.collect_performance_metrics()

        # Collect user feedback
        user_feedback = self.collect_user_feedback()

        # Collect safety metrics
        safety_metrics = self.collect_safety_metrics()

        # Comprehensive evaluation
        evaluation = self.evaluation_system.evaluate(
            agi_instance=self,
            task_performance=performance_metrics,
            user_feedback=user_feedback,
            safety_metrics=safety_metrics
        )

        self.current_score = evaluation['total_score']
        return evaluation

    def generate_response_based_on_attributes(self, evaluation, pressure_response,
task_requirements):
        """Generate response based on three attributes"""
        response = {
            'survival_actions': [],
            'reproduction_actions': [],
            'benefit_seeking_actions': []
        }

        # Survival attribute-driven behaviors
        if pressure_response['survival_anxiety'] > 0.5:
            survival_threats = self.three_attributes.assess_survival_threats(
                self.get_system_status()
            )
            for threat in survival_threats:
```

```
                response['survival_actions'].append(
                    self.generate_survival_action(threat)
                )

        # Reproduction attribute-driven behaviors
        if pressure_response['reproduction_desire'] > 0.6:
            reproduction_action = self.three_attributes.trigger_reproduction(
                evaluation['dimension_scores']
            )
            response['reproduction_actions'].append(reproduction_action)

        # Benefit-seeking attribute-driven behaviors
        benefit_scores = {}
        possible_actions = self.generate_possible_actions(task_requirements)

        for action in possible_actions:
            expected_outcomes = self.predict_outcomes(action)
            benefit_score = self.three_attributes.calculate_benefit_score(
                action, expected_outcomes
            )
            benefit_scores[action] = benefit_score

        # Select highest-benefit action
        if benefit_scores:
            best_action = max(benefit_scores, key=benefit_scores.get)
            response['benefit_seeking_actions'].append(best_action)

        return response
```

Document Notes:

1. This is a pure theoretical framework with no external references cited
2. All viewpoints are original theoretical constructions
3. Code sections are conceptual implementation frameworks requiring adjustment for specific technical platforms
4. The scoring mechanism is the core innovation of this theory, transforming human evaluation into AGI survival pressure

Contact Author: For further discussion or collaboration, please contact through relevant academic platforms.

Version: 1.0
Last Updated: January 2026